Oracle® Fusion Middleware

Administering Security for Oracle WebLogic Server

12*c* (12.1.2)

E28147-08

June 2015

Documentation for application architects, developers, and security administrators that explains how to configure WebLogic Server security, including settings for security realms, providers, identity and trust, single sign-on, and SSL.

ORACLE

Oracle Fusion Middleware Administering Security for Oracle WebLogic Server, 12*c* (12.1.2)

E28147-08

Primary Author: Oracle Corporation

Contents

5 Configuring Authentication Providers

8 Migrating Security Data

9 Managing the RDBMS Security Store

10 Managing the Embedded LDAP Server

11 Configuring Identity and Trust

12 Configuring SSL

16 Using Compatibility Security

17 Security Configuration MBeans

Preface

This preface describes the document accessibility features and conventions used in this guide—*Administering Security for Oracle WebLogic Server*.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction and Roadmap

This document explains how to configure WebLogic Server security, including settings for security realms, providers, identity and trust, SSL, and Compatibility security. See Section 1.3, "Related Information" for a description of other WebLogic security documentation.

The following sections describe the contents and organization of this guide, *Administering Security for Oracle WebLogic Server*, as well as new and changed security features in this release.

1.1 Document Scope and Audience

This document is intended for the following audiences:

- Application Architects—Architects who, in addition to setting security goals and designing the overall security architecture for their organizations, evaluate WebLogic Server security features and determine how to best implement them. Application Architects have in-depth knowledge of Java programming, Java security, and network security, as well as knowledge of security systems and leading-edge, security technologies and tools.
- Security Developers—Developers who define the system architecture and infrastructure for security products that integrate with WebLogic Server and who develop custom security providers for use with WebLogic Server. They work with Application Architects to ensure that the security architecture is implemented according to design and that no security holes are introduced, and work with Server Administrators to ensure that security is properly configured. Security Developers have a solid understanding of security concepts, including authentication, authorization, auditing (AAA), in-depth knowledge of Java (including Java Management eXtensions (JMX)), and working knowledge of WebLogic Server and security provider functionality.
- Application Developers—Java programmers who focus on developing client applications, adding security to Web applications and Enterprise JavaBeans (EJBs), and working with other engineering, quality assurance (QA), and database teams to implement security features. Application Developers have in-depth/working

knowledge of Java (including Java EE components such as servlets/JSPs and JSEE) and Java security.

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

1.2 Guide to This Document

This document is organized as follows:

- This chapter describes the audience, organization, and related information for this guide.
- Chapter 2, "Overview of Security Management" describes the default security configuration in WebLogic Server; lists the configuration steps for security, and describes Compatibility security.
- Chapter 3, "Customizing the Default Security Configuration" explains when to customize the default security configuration, the configuration requirements for a new security realm, and how to set a security realm as the default security realm.
- Chapter 4, "Configuring WebLogic Security Providers" describes the available configuration options for the security providers supplied by WebLogic Server and how to configure a custom security provider.
- Chapter 5, "Configuring Authentication Providers" describes the Authentication providers supplied by WebLogic Server, including information about how to configure them.
- Chapter 6, "Configuring Single Sign-On with Microsoft Clients" describes how to configure authentication between a WebLogic domain and .NET Web service clients or browser clients (for example, Internet Explorer) in a Microsoft domain, using Windows authentication based on the Simple and Protected Negotiate (SPNEGO) mechanism.
- Chapter 7, "Configuring Single Sign-On with Web Browsers and HTTP Clients" describes how to configure authentication between a WebLogic domain and Web browsers or other HTTP clients, using authentication based on the Security Assertion Markup Language (SAML).
- Chapter 8, "Migrating Security Data" provides information about exporting and importing security data between security realms and security providers.

- Chapter 10, "Managing the Embedded LDAP Server" describes the management tasks associated with the embedded LDAP server used by the WebLogic security providers.
- Chapter 9, "Managing the RDBMS Security Store" describes the steps required to configure the RDBMS security store, which enables you to store the security data managed by several security providers in an external RDBMS system rather than in the embedded LDAP server. The use of the RDBMS security store is required for SAML 2.0 services when configured on multiple servers in a domain, such as in a cluster.
- Chapter 11, "Configuring Identity and Trust" describes how to configure identity and trust for WebLogic Server.
- Chapter 12, "Configuring SSL" describes how to configure SSL for WebLogic Server.
- Chapter 13, "Configuring Oracle OPSS Keystore Service" describes how to configure the Oracle Platform Security Services (OPSS) Keystore Service for use with WebLogic Server.
- Chapter 14, "Configuring Security for a WebLogic Domain" describes how to set security configuration options for a WebLogic domain.
- Chapter 15, "Configuring JASPIC Security" describes how to configure the Java Authentication Service Provider Interface for Containers (JASPIC).
- Chapter 16, "Using Compatibility Security" describes how to use Compatibility security, a security configuration mode designed for backwards compatibility with security realms developed under WebLogic Server 6.x.
- Chapter 17, "Security Configuration MBeans" describes which WebLogic Security MBeans and MBean attributes are dynamic (can be changed without restarting the server) and which are non-dynamic (changes require a server restart).

1.3 Related Information

The following Oracle Oracle Fusion Middleware documents contain information that is relevant to the WebLogic Security Service:

- *Understanding Security for Oracle WebLogic Server*—Summarizes the features of the WebLogic Security Service, including an overview of its architecture and capabilities. It is the starting point for understanding WebLogic security.
- *Developing Security Providers for Oracle WebLogic Server*—Provides security vendors and application developers with the information needed to develop custom security providers that can be used with WebLogic Server.
- *Securing a Production Environment for Oracle WebLogic Server*—Highlights essential security hardening and lockdown measures for you to consider before you deploy WebLogic Server in a production environment.
- *Securing Resources Using Roles and Policies for Oracle WebLogic Server*—Introduces the various types of WebLogic resources, and provides information about how to secure these resources using WebLogic Server. This document focuses primarily on securing URL (Web) and Enterprise JavaBean (EJB) resources.
- *Developing Applications with the WebLogic Security Service*—Describes how to develop secure Web applications.
- *Securing WebLogic Web Services for Oracle WebLogic Server*—Describes how to develop and configure secure Web services.

- *Oracle WebLogic Server Administration Console Online Help*—Many security configuration tasks can be performed using the WebLogic Administration Console. The console's online help describes configuration procedures and provides a reference for configurable attributes.
- *Upgrading Oracle WebLogic Server*—Provides procedures and other information you need to upgrade from earlier versions of WebLogic Server to this release. It also provides information about moving applications from an earlier version of WebLogic Server to this release. For specific information about compatibility issues related to security and upgrading, see *Upgrading Oracle WebLogic Server*.
- *Java API Reference for Oracle WebLogic Server*—Provides reference documentation for the WebLogic security packages that are provided with and supported by this release of WebLogic Server.

1.4 Security Samples and Tutorials

In addition to the documents listed in Section 1.3, "Related Information", Oracle provides a variety of code samples for developers, some packaged with WebLogic Server and others available at the Oracle Technology Network (OTN) at http://www.oracle.com/technetwork/indexes/samplecode/weblogic-sample-522121.html.

1.4.1 Security Examples in the WebLogic Server Distribution

WebLogic Server optionally installs API code examples in *EXAMPLES_HOME*\wl_server\examples\src\examples\security, where *EXAMPLES_HOME* represents the directory in which the WebLogic Server code examples are configured. For more information about the WebLogic Server code examples, see "Sample Applications and Code Examples" in *Understanding Oracle WebLogic Server*.

The following examples are included to illustrate WebLogic security features:

- Java Authentication and Authorization Service
- Outbound and Two-way SSL

1.4.2 Additional Examples Available for Download

Additional WebLogic Server security examples are available for download at the Oracle Technology Network (OTN) at http://www.oracle.com/technetwork/indexes/samplecode/weblogic-sample-522121.html. These examples are distributed as .zip files that you can unzip into an existing WebLogic Server samples directory structure.

You build and run the downloadable examples in the same manner as you would an installed WebLogic Server example. See the download pages of individual examples for more information.

1.5 New and Changed Security Features

In this release, WebLogic Server includes support for the Oracle OPSS Keystore Service. For more information, see Chapter 13, "Configuring Oracle OPSS Keystore Service."

For a comprehensive listing of the new WebLogic Server features introduced in this release, see *What's New in Oracle WebLogic Server*.

2

Overview of Security Management

This chapter describes the basic features of the WebLogic Server security system. For a broader overview, see *Understanding Security for Oracle WebLogic Server*.

This chapter includes the following sections:

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

2.1 Security Realms 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. 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 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`.

You can 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 Section 2.4, "The Default Security Configuration in WebLogic Server".

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

For information about Compatibility security, see Chapter 16, "Using Compatibility Security".

2.2 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 by 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 Administration Console allows you to administer and manage all your security providers through one unified management interface.

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

- **Authentication**—Authentication is the process whereby the identity of users or system processes are proved or verified. Authentication also involves remembering, transporting, and making identity information available to various components of a system when that information is needed. Authentication providers supported by the WebLogic Security Service supply 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
- **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 a principal (an authenticated user, group, or system process).
- **Authorization**—Authorization is the process whereby the interactions between users and WebLogic resources are limited to ensure integrity, confidentiality, and availability. In other words, once a user's identity has been established by an authentication provider, authorization is responsible for determining whether access to WebLogic resources should be permitted for that user. An Authorization provider supplies these services.
- **Role Mapping**—You can assign one or more roles to multiple users and then specify access rights for users who hold particular roles. A Role Mapping provider 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)).
- **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 access decision.

- **Credential Mapping**—A credential map is a mapping of credentials used by WebLogic Server to credentials used in a legacy or remote system, which tell WebLogic Server how to connect to a given resource in that system. In other words, credential maps allow WebLogic Server to log into a remote system on behalf of a subject that has already been authenticated. Credential Mapping providers map credentials in this way.
- **Keystore**—A keystore is a mechanism for creating and managing password-protected stores of private keys and certificates for trusted certificate authorities. The keystore is available to applications that may need it for authentication or signing purposes. In the WebLogic Server security architecture, the WebLogic Keystore provider is used to access keystores.

  Note: The WebLogic Server Keystore provider is deprecated and is only supported for backward compatibility. Use keystores instead. For more information about configuring keystores, see Section 11.4.3.1, "Configuring Keystores for Production".

- **Certificate Lookup and Validation (CLV)**—X.509 certificates need to be located and validated for purposes of identity and trust. CLV providers receive certificates, certificate chains, or certificate references, complete the certificate path (if necessary), and validate all the certificates in the path. There are two types of CLV providers:
  - A CertPath Builder looks up and optionally completes the certificate path and validates the certificates.
  - A CertPath Validator looks up and optionally completes the certificate path, validates the certificates, and performs extra validation (for example, revocation checking).
- **Certificate Registry**—A certificate registry is a mechanism for adding certificate revocation checking to a security realm. The registry stores a list of valid certificates. Only registered certificates are valid. A certificate is revoked by removing it from the certificate registry. The registry is stored in the embedded LDAP server. The Certificate Registry is both a CertPath Builder and a CertPath Validator.
- **Auditing**—Auditing is the process whereby information about security requests and the outcome of those security requests is collected, stored, and distributed for the purpose of non-repudiation. In other words, auditing provides an electronic trail of computer activity. An Auditing provider supplies these services.

For information about the functionality provided by the WebLogic security providers, see Chapter 4, "Configuring WebLogic Security Providers" and Chapter 5, "Configuring Authentication Providers".

For information about the default security configuration, see Section 2.4, "The Default Security Configuration in WebLogic Server".

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

2.3 Security Policies and WebLogic Resources

WebLogic Server uses security policies (which replace the ACLs and permissions used in WebLogic Server 6.x) to protect WebLogic resources. 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 Resources Using Roles and Policies for Oracle WebLogic Server*. That document should be used in conjunction with Securing WebLogic Security to ensure security is completely configured for a WebLogic Server deployment.

2.3.1 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 WebLogic Scripting 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 the Oracle bidirectional COM-Java (jCOM) bridging tool.
- Enterprise Information System (EIS) resources that are designed as resource adapters, which allow the integration of Java applications with existing enterprise information systems. These resource adapters are also known as connectors.
- 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. Use the URL resource instead.

- Web services resources related to services that can be shared by and used as components of distributed, Web-based applications. This type of resource can be an entire Web service or individual components of a Web service (such as a

stateless session EJB, particular methods in that EJB, the Web application that contains the `web-services.xml` file, and so on).

- Remote resources.

2.3.2 Deployment Descriptors and the WebLogic Server Administration Console

WebLogic Server offers a choice of models for configuring security roles and policies. Under the standard Java Enterprise Edition model, you define role mappings and policies in the Web application or EJB deployment descriptors. The WebLogic Security Service can use information defined in deployment descriptors to grant security roles and define security policies for Web applications and EJBs. When WebLogic Server is booted for the first time, security role and security policy information stored in `web.xml`, `weblogic.xml`, `ejb-jar.xml`, or `weblogic-ejb-jar.xml` deployment descriptors is loaded into the Authorization and Role Mapping providers configured in the default security realm. You can then view the role and policy information from the Administration Console. (Optionally, you may configure the security realm to use a different security model that allows you to make changes to that information via the Administration Console as well.)

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

For more information, see "Options for Securing Web Application and EJB Resources" in *Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

2.4 The Default Security Configuration in WebLogic Server

To simplify the configuration and management of security, WebLogic Server provides a default security configuration. In the default security configuration, `myrealm` is set as the default security realm and the WebLogic Adjudication, Authentication, Identity Assertion, XACML Authorization, Credential Mapping, XACML Role Mapping, and CertPath providers are defined as the security providers. WebLogic Server's embedded LDAP server is used as the data store for these default 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.

> **Note:** WebLogic Server includes the WebLogic Authorization provider, which is referred to in the Administration Console and elsewhere as the Default Authorizer, and the WebLogic Role Mapping provider, which is referred to in the Administration Console and elsewhere as the Default RoleMapper. Beginning with WebLogic Server 9.1, these providers are no longer the default providers in newly-created security realms. Instead, the XACML Authorization provider and the XACML Role Mapping provider are the default providers.

For a description of the functionality provided by the WebLogic Security providers, see *Understanding Security for Oracle WebLogic Server*. If the WebLogic security providers do not fully meet your security requirements, you can supplement or replace them. See *Developing Security Providers for Oracle 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. See Chapter 3, "Customizing the Default Security Configuration".

2.5 Configuring WebLogic Security: Main Steps

Because WebLogic Server's 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, Oracle recommends the following procedure:

1. Determine whether or not to use the default security configuration by reading Section 3.1, "Why Customize the Default Security Configuration?"
   - 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. Configure additional 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 Section 3.1, "Why Customize the Default Security Configuration?" For information about creating custom security providers, see *Developing Security Providers for Oracle WebLogic Server*.

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

3. Optionally, configure the embedded LDAP server. WebLogic Server's embedded LDAP server is configured with default options. However, you may want to change those options to optimize the use of the embedded LDAP server in your environment. See Chapter 10, "Managing the Embedded LDAP Server".

4. Ensure that user accounts are properly secured. WebLogic Server provides a set of configuration options for protecting user accounts. By default, they are set for maximum security. However, during the development and deployment of WebLogic Server, you may need to weaken the restrictions on user accounts. Before moving to production, check that the options on user accounts are set for maximum protection. If you are creating a new security realm, you need to set the user lockout options. See Section 14.6, "How Passwords Are Protected in WebLogic Server" and Section 14.7, "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 Resources Using Roles and Policies for Oracle WebLogic Server*. *Administering Security for Oracle WebLogic Server* should be used in conjunction with *Securing Resources Using Roles and Policies for Oracle WebLogic Server* to ensure security is completely configured for a WebLogic Server deployment.
6. Configure identity and trust for WebLogic Server. (This step is optional but recommended.) See Chapter 11, "Configuring Identity and Trust".
7. Enable SSL for WebLogic Server. (This step is optional but recommended.) See Chapter 12, "Configuring SSL".
8. When you have moved to production, review and implement the additional security options described in *Securing a Production Environment for Oracle WebLogic Server*.

In addition, you can:

- Configure a connection filter. See Section 14.3, "Using Connection Filters".
- Enable interoperability between WebLogic domains. See Section 14.2.1, "Enabling Cross Domain Security Between WebLogic Server Domains".

2.6 Methods of Configuring Security

In many cases, this document describes how to configure WebLogic security by using the WebLogic Server Administration Console. Generally, any configuration task you can accomplish through the Console you can also accomplish by using the WebLogic Scripting Tool or the Java Management Extensions (JMX) APIs. For information about using WLST to manage WebLogic security, see "Managing Security Data (WLST Online)" in *Understanding the WebLogic Scripting Tool*. For information about using JMX APIs, see "Choosing an MBean Server to Manage Security Realms" in *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*.

When you manage security realms, you must use two different MBean servers depending on your task:

- To set the value of a security MBean attribute, you must use the Edit MBean Server.
- To add users, groups, roles, and policies, or to invoke other operations in a security provider MBean, you must use a Runtime MBean Server or the Domain Runtime MBean Server.

In addition, to prevent the possibility of incompatible changes, you cannot invoke operations in security provider MBeans if your client or another JMX client has an edit session currently active. The Administration Console automatically enforces this limitation and automatically accesses the proper MBean server. When you use the Administration Console, you can override this limitation by selecting the **Domain** > **Security** > **General** page and enabling **Allow Security Management Operations if**

Non-dynamic Changes have been Made. Setting this attribute to true permits users to perform security management operations without restarting the server. Note that this attribute is reset to false when a new MBean edit session begins.

For example, the value of the `MinimumPasswordLength` attribute in `DefaultAuthenticatorMBean` is stored in the domain's configuration document. Because all modifications to this document are controlled by WebLogic Server, to change the value of this attribute you must use the Edit MBean Server and acquire a lock on the domain's configuration. The `createUser` operation in `DefaultAuthenticatorMBean` adds data to an LDAP server, which is not controlled by WebLogic Server. To prevent incompatible changes between the `DefaultAuthenticatorMBean`'s configuration and the data that it uses in the LDAP server, you cannot invoke the createUser operation if you or other users are in the process of modifying the `MinimumPasswordLength` attribute. In addition, because changing this attribute requires you to restart WebLogic Server, you cannot invoke the `createUser` operation until you have restarted the server.

2.7 What Is Compatibility Security?

Compatibility security refers to the capability to run security configurations developed under WebLogic Server 6.x in this release of WebLogic Server. In Compatibility security, you manage 6.x security realms, users, groups, and ACLs, protect user accounts, and configure the Realm Adapter Auditing provider and optionally the Identity Assertion provider in the Realm Adapter Authentication provider.

The only security realm available in Compatibility security is the `CompatibilityRealm`. The Realm Adapter providers (Auditing, Adjudication, Authorization, and Authentication) in the Compatibility realm allow backward compatibility with the authentication, authorization, and auditing services in 6.x security realms. For more information, see Chapter 16, "Using Compatibility Security".

Note: Compatibility security is deprecated and will not be supported in future major releases. Oracle strongly recommends upgrading your WebLogic Server deployment to the security features in this release of WebLogic Server. You should only use Compatibility security pending such an upgrade.

2.7.1 Management Tasks Available in Compatibility Security

Because Compatibility security allows you to access only 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:

- Configure the Realm Adapter Auditing provider. For more information, see Section 16.6, "Configuring a Realm Adapter Auditing Provider".
- 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 Section 16.5, "Configuring the Identity Assertion Provider in the Realm Adapter Authentication Provider".

> **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 Section 5.5, "Configuring RDBMS Authentication Providers".

- Change the password of the `system` user to protect your WebLogic Server deployment.
- Manage the security realm in the `CompatibilityRealm`.
- Define additional users for the security realm in the `CompatibilityRealm`. Organize users further by implementing groups in the security realm.
- Manage ACLs and permissions for the resources in your WebLogic Server deployment.
- Create security roles and security policies for WebLogic resources you add to the `CompatibilityRealm`. For more information, see *Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

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

- Chapter 11, "Configuring Identity and Trust"
- Chapter 12, "Configuring SSL"
- Chapter 14, "Configuring Security for a WebLogic Domain"

3

Customizing the Default Security Configuration

This chapter describes how you can customize the default security configuration by creating a new security realm.

This chapter includes the following sections:

- Why Customize the Default Security Configuration?
- Before You Create a New Security Realm
- Creating and Configuring a New Security Realm: Main Steps

For information about configuring security providers, see Chapter 4, "Configuring WebLogic Security Providers" and Chapter 5, "Configuring Authentication Providers".

For information about migrating security data to a new security realm, see Chapter 8, "Migrating Security Data".

3.1 Why Customize the Default Security Configuration?

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

Customize the default security configuration if you want to do any of the following:

- Replace one of the security providers in the default realm with a different security provider.
- Configure additional security providers in the default security realm. (For example, if you want to use two Authentication providers, one that uses the embedded LDAP server and one that uses a Windows NT store of users and groups.)
- Use an Authentication provider that accesses an LDAP server other than WebLogic Server's embedded LDAP server.
- Use an existing store of users and groups (for example, a DBMS database) instead of defining users and groups in the WebLogic Authentication provider (also known as the DefaultAuthenticator).
- When performing authentication, use the GUID or DN attributes of principals, in addition to user names, specify that principal matching is case-insensitive.
- Add an Auditing provider to the default security realm.

- Use an Identity Assertion provider that handles SAML assertions or Kerberos tokens.
- Use the Certificate Registry to add certificate revocation to the security realm.
- Change the default configuration settings of the security providers.
- Use a custom Authorization or Role Mapping provider that does not support parallel security policy and role modification, respectively, in the security provider database.

For information about configuring different types of security providers in a security realm, see Chapter 4, "Configuring WebLogic Security Providers" and Chapter 5, "Configuring Authentication Providers".

The easiest way to customize the default security configuration is to add the security providers you want to the default security realm (`myrealm`). However, Oracle recommends instead that you customize the default security configuration by creating an entirely new security realm. This preserves your ability to revert more easily to the default security configuration. You configure security providers for the new realm; migrate any security data, such as users as groups, from the existing default realm; and then set the new security realm as the default realm. See Section 3.3, "Creating and Configuring a New Security Realm: Main Steps".

3.2 Before You Create a New Security Realm

Before creating a new security realm, you need to decide:

- Which security providers you want to use. WebLogic Server includes a wide variety of security providers and, in addition, allows you to create or obtain custom security providers. A valid security realm requires an Authentication provider, an Authorization provider, an Adjudication provider, a Credential Mapping provider, a Role Mapping provider, and a CertPathBuilder. In addition, a security realm can optionally include Identity Assertion, Auditing, and Certificate Registry providers. If your new security realm includes two or more providers of the same type (for example, more than one Authentication provider or more than one Authorization provider), you need to determine how these providers should interact with each other. See Section 5.2, "Using More Than One Authentication Provider".

  In addition, custom Authorization and Role Mapping providers may or may not support parallel security policy and role modification, respectively, in the security provider database. If your custom Authorization and Role Mapping security providers do not support parallel modification, the WebLogic Security framework can enforce a synchronization mechanism that results in each application and module being placed in a queue and deployed sequentially. To do this, set the **Deployable Provider Synchronization Enabled** and **Deployable Provider Synchronization Timeout** controls for the realm.
- What model to use to set security roles and security policies for Web application and EJB resources. These security roles and policies can be set through deployment descriptors or through the WebLogic Administration Console. See "Options for Securing Web Application and EJB Resources" in *Securing Resources Using Roles and Policies for Oracle WebLogic Server*.
- Whether or not to use the Web resource.

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

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

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

For more information, see "Configure new security realms" in the *Oracle WebLogic Server Administration Console Online Help*.

3.3 Creating and Configuring a New Security Realm: Main Steps

To create a new security realm:

1. Define a name and set the configuration options for the security realm. See Section 3.2, "Before You Create a New Security Realm" and "Configure new security realms" in the *Oracle WebLogic Server Administration Console Online Help*.
2. Configure the required security providers for the security realm. A valid security realm requires an Authentication provider, an Authorization provider, an Adjudication provider, a Credential Mapping provider, a Role Mapping provider, and a CertPathBuilder. See Chapter 4, "Configuring WebLogic Security Providers" and Chapter 5, "Configuring Authentication Providers".
3. Optionally, define Identity Assertion, Auditing, and Certificate Registry providers. See Chapter 4, "Configuring WebLogic Security Providers" and Chapter 5, "Configuring Authentication Providers".
4. If you configured the Default Authentication, Authorization, Credential Mapping or Role Mapping provider or the Certificate Registry in the new security realm, verify that the settings of the embedded LDAP server are appropriate. See Chapter 10, "Managing the Embedded LDAP Server".
5. Optionally, configure caches to improve the performance of the WebLogic or LDAP Authentication providers in the security realm. See Section 5.4.13, "Improving the Performance of WebLogic and LDAP Authentication Providers".
6. Protect WebLogic resources in the new security realm with security policies. Creating security policies is a multi-step process with many options. To fully understand this process, read *Securing Resources Using Roles and Policies for Oracle WebLogic Server* in conjunction with *Administering Security for Oracle WebLogic Server* to ensure security is completely configured for a WebLogic Server deployment.
7. If the security data (users and groups, roles and policies, and credential maps) defined in the existing security realm will also be valid in the new security realm, you can export the security data from the existing realm and import it into the new security realm. See Chapter 8, "Migrating Security Data".
8. Protect user accounts in the new security realm from dictionary attacks by setting lockout attributes. See Section 14.7, "Protecting User Accounts".
9. Set the new realm as the default security realm for the WebLogic domain. See "Change the default security realm" in the *Oracle WebLogic Server Administration Console Online Help*.

Note: You can also use the WebLogic Scripting Tool or Java Management Extensions (JMX) APIs to create a new security configuration. See *Understanding the WebLogic Scripting Tool*.

4

Configuring WebLogic Security Providers

This chapter describes how to configure the security providers supplied by WebLogic Server.

> **Note:** WebLogic Server includes so many Authentication providers and Identity Assertion providers that they are better handled in a separate section. See Chapter 5, "Configuring Authentication Providers".

This chapter includes the following sections:

4.1 When Do You Need to Configure a Security Provider?

By default, most WebLogic security providers are generally configured to run after you install WebLogic Server. However, the following circumstances require you to supply configuration information:

- Before using the WebLogic Identity Assertion provider, define the active token type. See Section 5.9, "Configuring Identity Assertion Providers".
- To map tokens to a user in a security realm, configure the user name mapper in the WebLogic Identity Assertion provider. See Section 4.8, "Configuring a WebLogic Credential Mapping Provider".

- To use auditing in the default (active) security realm, configure either the WebLogic Auditing provider or a custom Auditing provider. See Section 4.7, "Configuring the WebLogic Auditing Provider".
- To use HTTP and Kerberos-based authentication in conjunction with WebLogic Server. See Chapter 6, "Configuring Single Sign-On with Microsoft Clients".
- To use identity assertion based on SAML assertions. See Chapter 7, "Configuring Single Sign-On with Web Browsers and HTTP Clients".
- To use certificate revocation. See Section 4.12, "Configuring the Certificate Lookup and Validation Framework".
- To use an LDAP server other than the embedded LDAP server, configure one of the LDAP Authentication providers. An LDAP authentication provider can be used instead of or in addition to the WebLogic Authentication provider. See Section 5.4, "Configuring LDAP Authentication Providers".
- To access user, password, group, and group membership information stored in databases for authentication purposes. See Section 5.5, "Configuring RDBMS Authentication Providers". The RDBMS Authentication providers can be used to upgrade from the RDBMS security realm.
- To use Windows NT users and groups for authentication purposes. See Section 5.6, "Configuring a Windows NT Authentication Provider". The Windows NT Authentication provider is the upgrade path for the Window NT security realm.
- When you create a new security realm, configure security providers for that realm. See Section 3.3, "Creating and Configuring a New Security Realm: Main Steps".
- When you add a custom security provider to a security realm or replace a WebLogic security provider with a custom security provider, configure options for the custom security provider. When you create a custom security provider, you can implement options that are configurable through the Administration Console. However, those options are implementation-specific and are not addressed in this manual. See *Extending the Administration Console for Oracle WebLogic Server*.

You can use either the WebLogic-supplied security providers or a custom security provider in a security realm. To configure a custom security provider, see "Configure custom security providers" in the *Oracle WebLogic Server Administration Console Online Help*.

4.2 Reordering Security Providers

You can configure more than one security provider of a given type in a security realm. For example, you might use two or more different Role Mapping providers or Authorization providers. If you have more than one security provider of the same type in a security realm, the order in which these providers are called can affect the overall outcome of the security processes. By default, security providers are called in the order that they were added to the realm. You can use the Administration Console to change the order of the providers. See "Re-order security providers" in the *Oracle WebLogic Server Administration Console Online Help*.

4.3 Enabling Synchronization in Security Policy and Role Modification at Deployment

For the best performance, and by default, Weblogic Server supports parallel modification to security policy and roles during application and module deployment. For this reason, deployable Authorization and Role Mapping providers configured in

the security realm should support parallel calls. The WebLogic deployable XACML Authorization and Role Mapping providers meet this requirement.

However, custom deployable Authorization and Role Mapping providers may or may not support parallel calls. If your custom deployable Authorization or Role Mapping providers do not support parallel calls, you need to disable the parallel security policy and role modification and instead enforce a synchronization mechanism that results in each application and module being placed in a queue and deployed sequentially. Otherwise, if a provider does not support parallel calls, it generates a `java.util.ConcurrentModificationException` exception.

You can turn on this synchronization enforcement mechanism on in two ways:

> **Note:** Enabling the synchronization mechanism affects every deployable provider configured in the realm, including the WebLogic Server XACML providers. Enabling the synchronization mechanism may negatively impact the performance of these providers.

- From the WebLogic Server Administration Console. Set the **Deployable Provider Synchronization Enabled** and **Deployable Provider Synchronization Timeout** controls for the realm.

  The **Deployable Provider Synchronization Enabled** control enforces a synchronization mechanism that results in each application and module being placed in a queue and deployed sequentially.

  The **Deployable Provider Synchronization Timeout** control sets or returns the timeout value, in milliseconds, for the deployable security provider synchronization operation. This is the maximum time a deployment cycle wants to wait in the queue when the previous cycle is stuck.

- From the `DeployableProviderSynchronizationEnabled` and `DeployableProviderSynchronizationTimeout` attributes of the `RealmMBean`. From WLST, set the `DeployableProviderSynchronizationEnabled` and `DeployableProviderSynchronizationTimeout` attributes of the RealmMBean.

  See RealmMBean in *MBean Reference for Oracle WebLogic Server*.

4.4 Configuring an 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 should only need to configure an Authorization provider when you create a new security realm.

By default, security realms in newly created domains include the XACML Authorization provider. The XACML Authorization provider uses XACML, the eXtensible Access Control Markup Language. For information about using the XACML Authorization provider, see "Using XACML Documents to Secure WebLogic Resources" in *Securing Resources Using Roles and Policies for Oracle WebLogic Server*. WebLogic Server also includes the WebLogic Authorization provider, which uses a proprietary policy language. This provider is named DefaultAuthorizer, but is no longer the default authorization provider.

See Section 4.3, "Enabling Synchronization in Security Policy and Role Modification at Deployment" for information about how Authorization providers support parallel modification to security policy during application and module deployment.

See "Configure Authorization providers" in the *Oracle WebLogic Server Administration Console Online Help*.

> **Note:** The WebLogic Authorization provider improves performance by caching the roles, predicates, and resource data that it looks up. For information on configuring these caches, see "Best Practices: Configure Entitlements Caching When Using WebLogic Providers" in *Securing Resources Using Roles and Policies for Oracle WebLogic Server*. The XACML Authorization uses its own cache, but this cache is not configurable.

4.5 Configuring the WebLogic Adjudication Provider

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

Each security realm requires an Adjudication provider, and can have no more than one active Adjudication provider. By default, a WebLogic security realm is configured with the WebLogic Adjudication provider. You can use either the WebLogic Adjudication provider or a custom Adjudication provider in a security realm.

> **Note:** In the Administration Console, the WebLogic Adjudication provider is referred to as the Default Adjudicator.

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

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

4.6 Configuring a Role Mapping Provider

Role Mapping providers compute the set of roles granted to a subject for a given resource. Role Mapping providers supply Authorization providers with this role information so that the Authorization provider can answer the "is access allowed?" question for WebLogic resources. By default, a WebLogic security realm is configured with the XACML Role Mapping provider. The XACML Role Mapping provider uses XACML, the eXtensible Access Control Markup Language. For information about using the XACML Role Mapping provider, see "Using XACML Documents to Secure WebLogic Resources" in *Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

WebLogic Server also includes the WebLogic Role Mapping provider, which uses a proprietary policy language. This provider is named DefaultRoleMapper, but is no longer the default role mapping provider in newly-created security realms. You can also use a custom Role Mapping provider in your security realm.

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

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

See Section 4.3, "Enabling Synchronization in Security Policy and Role Modification at Deployment" for information about how Role Mapping providers support parallel modification to roles during application and module deployment.

For information about using, developing, and configuring Role Mapping providers, see:

- "Users, Groups, And Security Roles" in *Securing Resources Using Roles and Policies for Oracle WebLogic Server*
- "Role Mapping Providers" in *Developing Security Providers for Oracle WebLogic Server*
- "Configure Role Mapping providers" in the *Oracle WebLogic Server Administration Console Online Help*

> **Note:** The WebLogic Role Mapping provider improves performance by caching the roles, predicates, and resource data that it looks up. For information on configuring these caches, see "Best Practices: Configure Entitlements Caching When Using WebLogic Providers" in *Securing Resources Using Roles and Policies for Oracle WebLogic Server*. The XACML Role Mapping provider uses its own cache, but this cache is not configurable.

4.7 Configuring the 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. WebLogic Server includes a provider named the WebLogic Auditing provider (referred to as `DefaultAuditor` in the Administration Console). You can also develop custom Auditing providers, as described in "Auditing Providers" in *Developing Security Providers for Oracle WebLogic Server*.

The WebLogic Auditing provider can log the events described in Table 4–1. In addition, if you enable configuration auditing (as described in Section 4.7.2, "Configuration Auditing"), the WebLogic Auditing provider can log the events described in Table 4–4.

Table 4–1 WebLogic Auditing Provider Events

The following audit event . . .	. . . indicates
`AUTHENTICATE`	A simple authentication (username and password) occurred.

Table 4–1 (Cont.) WebLogic Auditing Provider Events

The following audit event . . .	. . . indicates
ASSERTIDENTITY	A perimeter authentication (based on tokens) occurred.
USERLOCKED	A user account is locked because of invalid login attempts.
USERUNLOCKED	The lock on a user account is cleared.
USERLOCKOUTEXPIRED	The lock on a user account expired.
ISAUTHORIZED	An authorization attempt occurred.
ROLEEVENT	A `getRoles` event occurred.
ROLEDEPLOY	A `deployRole` event occurred.
ROLEUNDEPLOY	An `undeployRole` event occurred.
POLICYDEPLOY	A `deploy`Policy event occurred.
POLICYUNDEPLOY	An `undeploy`Policy event occurred.
START_AUDIT	An Auditing provider has been started.
STOP_AUDIT	An Auditing provider has been stopped.

By default, most configuration options for the WebLogic Auditing provider are already defined and, once it is added to the active security realm, the WebLogic Auditing provider will begin to record audit events. However, you need to define the following settings, which you can do in the WebLogic Server Administration Console by selecting the **Configuration** > **Provider Specific** page for the provider. You can also use WebLogic Scripting tool or the Java Management Extensions (JMX) APIs to configure the Auditing provider:

- Rotation Minutes—Specifies how many minutes to wait before creating a new `DefaultAuditRecorder.log` file. At the specified time, the audit file is closed and a new one is created. A backup file named `DefaultAuditRecorder.`*`YYYYMMDDHHMM`*`.log` (for example, `DefaultAuditRecorder.200405130110.log`) is created in the same directory.
- Severity—Severity level appropriate for your WebLogic Server deployment. The WebLogic Auditing provider audits security events of the specified severity, as well as all events with a higher numeric severity rank. For example, if you set the severity level to ERROR, the WebLogic Auditing provider audits security events of severity level ERROR, SUCCESS, and FAILURE. You can also set the severity level to CUSTOM, and then enable the specific severity levels you want to audit, such as ERROR and FAILURE events only. Audit events include both the severity name and numeric rank; therefore, a custom Auditing provider can filter events by either the name or the numeric rank. Auditing can be initiated when the following levels of security events occur.

Event Severity	Rank
INFORMATION	1
WARNING	2
ERROR	3
SUCCESS	4
FAILURE	5

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

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

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

For more information, see "Security" in the *Command Reference for Oracle WebLogic Server*.

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

For more information, see "Configure Auditing providers" in the *Oracle WebLogic Server Administration Console Online Help*.

4.7.1 Auditing ContextHandler Elements

An Audit Event includes a ContextHandler that can hold a variety of information or objects. Set the WebLogic Auditing provider's Active ContextHandler Entries attribute to specify which ContextElement entries in the ContextHandler are recorded by the Auditing provider. By default, none of the ContextElements are audited. Objects in the ContextHandler are in most cases logged using the toString method. Table 4–2 lists the available ContextHandler entries.

Table 4–2 Context Handler Entries for Auditing

Context Element Name	Description and Type
com.bea.contextelement.servlet.HttpServletRequest	A servlet access request or SOAP message via HTTP javax.http.servlet.HttpServletRequest
com.bea.contextelement.servlet.HttpServletResponse	A servlet access response or SOAP message via HTTP javax.http.servlet.HttpServletResponse
com.bea.contextelement.wli.Message	An Oracle WebLogic Integration message. The message is streamed to the audit log. java.io.InputStream
com.bea.contextelement.channel.Port	Internal listen port of the network channel accepting or processing the request java.lang.Integer
com.bea.contextelement.channel.PublicPort	External listen port of the network channel accepting or processing the request java.lang.Integer
com.bea.contextelement.channel.RemotePort	Port of the remote end of the TCP/IP connection of the network channel accepting or processing the request java.lang.Integer
com.bea.contextelement.channel.Protocol	Protocol used to make the request of the network channel accepting or processing the request java.lang.String

Table 4–2 (Cont.) Context Handler Entries for Auditing

Context Element Name	Description and Type
com.bea.contextelement.channel.Address	The internal listen address of the network channel accepting or processing the request java.lang.String
com.bea.contextelement.channel.PublicAddress	The external listen address of the network channel accepting or processing the request java.lang.String
com.bea.contextelement.channel.RemoteAddress	Remote address of the TCP/IP connection of the network channel accepting or processing the request java.lang.String
com.bea.contextelement.channel.ChannelName	Name of the network channel accepting or processing the request java.lang.String
com.bea.contextelement.channel.Secure	Whether the network channel is accepting or processing the request using SSL java.lang.Boolean
com.bea.contextelement.ejb20.Parameter[1-N]	Object based on parameter
com.bea.contextelement.wsee.SOAPMessage	javax.xml.rpc.handler.MessageContext
com.bea.contextelement.entitlement.EAuxiliaryID	Used by a WebLogic Server internal process. weblogic.entitlement.expression.EAuxiliary
com.bea.contextelement.security.ChainPrevalidatedBySSL	SSL framework has validated the certificate chain, meaning that the certificates in the chain have signed each other properly; the chain terminates in a certificate that is one of the server's trusted CAs; the chain honors the basic constraints rules; and the certificates in the chain have not expired. java.lang.Boolean
com.bea.contextelement.xml.SecurityToken	Not used in this release of WebLogic Server. weblogic.xml.crypto.wss.provider.SecurityToken
com.bea.contextelement.xml.SecurityTokenAssertion	Not used in this release of WebLogic Server. java.util.Map
com.bea.contextelement.webservice.Integrity{id:XXXXX}	javax.security.auth.Subject
com.bea.contextelement.saml.SSLClientCertificateChain	SSL client certificate chain obtained from the SSL connection over which a sender-vouches SAML assertion was received. java.security.cert.X509Certificate[]
com.bea.contextelement.saml.MessageSignerCertificate	Certificate used to sign a Web services message. java.security.cert.X509Certificate
com.bea.contextelement.saml.subject.ConfirmationMethod	Type of SAML assertion: bearer, artifact, sender-vouches, or holder-of-key. java.lang.String

Table 4–2 (Cont.) Context Handler Entries for Auditing

Context Element Name	Description and Type
com.bea.contextelement.saml.subject.dom.KeyInfo	<ds:KeyInfo> element to be used for subject confirmation with holder-of-key SAML assertions. org.w3c.dom.Element

4.7.2 Configuration Auditing

You can configure the Administration Server to emit log messages and generate audit events when a user changes the configuration of any resource within a domain or invokes management operations on any resource within a domain. For example, if a user disables SSL on a Managed Server in a domain, the Administration Server emits log messages. If you have enabled the WebLogic Auditing provider, it writes the audit events to an additional security log. These messages and audit events provide an audit trail of changes within a domain's configuration (configuration auditing).

The Administration Server writes configuration auditing messages to its local log file. They are not written to the domain-wide message log by default.

Note that configuration audit information is contained in Authorization Events. As a result, another approach to configuration auditing is to consume Authorization Events. Note, however, that the information in an Authorization Event tells you whether access was allowed to perform a configuration change; it does not tell you whether the configuration change actually succeeded (for instance, it might have failed because it was invalid).

4.7.3 Enabling Configuration Auditing

Enable configuration auditing by one of these methods:

- Use the Administration Console. Select the **Configuration** > **General** page for your domain and set the **Configuration Audit Type**. See "Enable configuration auditing" in the *Oracle WebLogic Server Administration Console Online Help*.
- When you start the Administration Server, include one of the following Java options in the `weblogic.Server` command:
  - `-Dweblogic.domain.ConfigurationAuditType="audit"`

    Causes the domain to emit Audit Events only.
  - `-Dweblogic.domain.ConfigurationAuditType="log"`

    Causes the domain to write configuration auditing messages to the Administration Server log file only.
  - `-Dweblogic.domain.ConfigurationAuditType="logaudit"`

    Causes the domain to emit Audit Events and write configuration auditing messages to the Administration Server log file.

  See "weblogic.Server Command-Line Reference" in *Command Reference for Oracle WebLogic Server*.
- Use the WebLogic Scripting Tool to change the value of the `ConfigurationAuditType` attribute of the `DomainMBean`. See *Understanding the WebLogic Scripting Tool*.

4.7.4 Configuration Auditing Messages

Configuration auditing messages are of the following severities:

Table 4–3 Configuration Auditing Message Severities

Severity	Description
SUCCESS	A successful configuration change occurred.
FAILURE	An attempt to modify the configuration failed due to insufficient user credentials.
ERROR	An attempt to modify the configuration failed due to an internal error.

Configuration auditing messages are identified by message IDs that fall within the range of `159900-159910`.

The messages use MBean object names to identify resources. Object names for WebLogic Server MBeans reflect the location of the MBean within the hierarchical data model. To reflect the location, object names contain name/value pairs from the parent MBean. For example, the object name for a server's `LogMBean` is: `mydomain:Name=myserverlog,Type=Log,Server=myserver`. See "WebLogic Server MBean Data Model" in *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*.

Table 4–4 summarizes the messages.

Table 4–4 Summary of Configuration Auditing Messages

When This Event Occurs...	WebLogic Server Generates a Message With This ID...	And This Message Text...
Authorized user creates a resource.	`159900`	`USER username CREATED MBean-name` where *username* identifies the WebLogic Server user who logged in and created a resource.
Unauthorized user attempts to create a resource.	`159901`	`USER username CREATED MBean-name FAILED weblogic.management. NoAccessRuntimeException: exception-text stack-trace` where *username* identifies the unauthorized WebLogic Server user.
Authorized user deletes a resource.	`159902`	`USER username REMOVED MBean-name` where *username* identifies the WebLogic Server user who logged in and deleted a resource.
Unauthorized user attempts to delete a resource.	`159903`	`USER username REMOVE MBean-name FAILED weblogic.management. NoAccessRuntimeException: exception-text stack-trace` where *username* identifies the unauthorized WebLogic Server user.
Authorized user changes a resource's configuration.	`159904`	`USER username MODIFIED MBean-name ATTRIBUTE attribute-name FROM old-value TO new-value` where *username* identifies the WebLogic Server user who logged in and changed the resource's configuration.

Table 4–4 (Cont.) Summary of Configuration Auditing Messages

When This Event Occurs...	WebLogic Server Generates a Message With This ID...	And This Message Text...
Unauthorized user attempts to change a resource's configuration.	159905	`USER username MODIFY MBean-name ATTRIBUTE attribute-name FROM old-value TO new-value FAILED weblogic.management. NoAccessRuntimeException: exception-text stack-trace` where *username* identifies the unauthorized WebLogic Server user.
Authorized user invokes an operation on a resource. For example, a user deploys an application or starts a server instance.	159907	`USER username INVOKED ON MBean-name METHOD operation-name PARAMS specified-parameters` where *username* identifies the WebLogic Server user who logged in and invoked a resource operation.
Unauthorized user attempts to invoke an operation on a resource.	159908	`USER username INVOKED ON MBean-name METHOD operation-name PARAMS specified-parameters FAILED weblogic.management. NoAccessRuntimeException: exception-text stack-trace` where *username* identifies the unauthorized WebLogic Server user.
Authorized user enables configuration auditing.	159909	`USER username, Configuration Auditing is enabled` where *username* identifies the WebLogic Server user who enabled configuration auditing.
Authorized user disables configuration auditing.	159910	`USER username, Configuration Auditing is disabled` where *username* identifies the WebLogic Server user who disabled configuration auditing.

Note: Each time an authorized user adds, modifies, or deletes a resource, the Management subsystem also generates an Info message with the ID 140009 regardless of whether configuration auditing is enabled. For example:

```
<Sep 15, 2005 11:54:47 AM EDT> <Info> <Management> <140009>
<Configuration changes for domain saved to the repository.>
```

While the message informs you that the domain's configuration has changed, it does not provide the detailed information that configuration auditing messages provide. Nor does the Management subsystem generate this message when you invoke operations on resources.

Table 4–5 lists additional message attributes for configuration auditing messages. All configuration auditing messages specify the same values for these attributes.

Table 4–5 Common Message Attributes and Values

Message Attribute	Attribute Value
Severity	`Info`
Subsystem	`Configuration Audit`
User ID	`kernel identity` This value is always `kernel` identity, regardless of which user modified the resource or invoked the resource operation.
Server Name	*`AdminServerName`* Because the Administration Server maintains the configuration data for all resources in a domain, this value is always the name of the Administration Server.
Machine Name	*`AdminServerHostName`* Because the Administration Server maintains the configuration data for all resources in a domain, this value is always the name of the Administration Server's host machine.
Thread ID	*`execute-thread`* The value depends on the number of execute threads that are currently running on the Administration Server.
Timestamp	*`timeStamp`* at which the message is generated.

4.7.5 Audit Events and Auditing Providers

An audit event is an object that Auditing providers can read and process in specific ways. An Auditing provider is a pluggable component that the security realm uses to collect, store, and distribute information about operating requests and the outcome of those requests for the purposes of non-repudiation.

If you enable a domain to emit Audit Events, the domain emits the events described in Table 4–6. All Auditing providers that are configured for the domain can handle these events.

All of the events are of severity level `SUCCESS` and describe the security principal who initiated the action, whether permission was granted, and the object (MBean or MBean attribute) of the requested action.

Table 4–6 Summary of Audit Events for Configuration Auditing

When This Event Occurs...	WebLogic Server Generates This Audit Event Object...
A request to create a new configuration artifact has been allowed or prevented.	`weblogic.security.spi.AuditCreateConfigurationEvent`
A request to delete an existing configuration artifact has been allowed or prevented.	`weblogic.security.spi.AuditDeleteConfigurationEvent`
A request to modify an existing configuration artifact has been allowed or prevented.	`weblogic.security.spi.AuditInvokeConfigurationEvent`
A invoke an operation on an existing configuration artifact has been allowed or prevented.	`weblogic.security.spi.AuditSetAttributeConfigurationEvent`

If you enable the default WebLogic Server Auditing provider, it writes all Audit Events as log messages in its own log file.

Other Auditing providers that you create or purchase can filter these events and write them to output repositories such as an LDAP server, database, or a simple file. In addition, other types of security providers can request audit services from an Auditing provider. See "Auditing Providers" in *Developing Security Providers for Oracle WebLogic Server*.

4.8 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) obtain an appropriate set of credentials to authenticate remote users to a target WebLogic resource. The WebLogic Credential Mapping provider maps WebLogic Server subjects to the username/password pairs to be used when accessing such resources.

By default, most configuration options for the WebLogic Credential Mapping provider are defined.

Note: WebLogic Server provides the option of setting Credential Mapping Deployment Enabled, which specifies whether or not the Credential Mapping provider imports credential maps from a resource adapter's deployment descriptor (`weblogic-ra.xml` file) into the security realm. However, this option is now deprecated. Deploying credential maps from a `weblogic-ra.xml` file is no longer supported by WebLogic Server.

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

For more information:

- See "Credential Mapping Providers" in *Developing Security Providers for Oracle WebLogic Server*.
- See "Configure Credential Mapping Providers" and "Create outbound credential mappings" in the *Oracle WebLogic Server Administration Console Online Help*.
- For information about using credential maps, see *Developing Resource Adapters for Oracle WebLogic Server*.
- You can also use the WebLogic Scripting Tool or Java Management Extensions (JMX) APIs to create a new security configuration.
- For information about other credential mapping providers, see Section 4.9, "Configuring a PKI Credential Mapping Provider" and Section 4.10, "Configuring a SAML Credential Mapping Provider for SAML 1.1".

4.9 Configuring a PKI Credential Mapping Provider

The PKI (Public Key Infrastructure) Credential Mapping provider included in WebLogic Server maps (a) a WebLogic Server subject (the initiator) and target resource (and an optional credential action) to (b) a key pair or public certificate that can be used by applications when accessing the targeted resource. The PKI Credential

Mapping provider uses the subject and resource name to retrieve the corresponding credential from the keystore.

To use the PKI Credential Mapping provider, you need to:

1. Configure keystores with appropriate keys and distribute the keystores on all machines in a WebLogic Server cluster. Setting up keystores is not a WebLogic Server function. For information about setting up keystores, see the help for the Java keytool utility at `http://docs.oracle.com/javase/7/docs/technotes/tools/solaris/keytool.html`. See also Chapter 11, "Configuring Identity and Trust" for information about keystores and keys in WebLogic Server.
2. Configure a PKI Credential Mapping provider. A PKI Credential Mapping provider is not already configured in the default security realm (`myrealm`). See Section 4.9.1, "PKI Credential Mapper Attributes" and "Configure Credential Mapping providers" in the *Oracle WebLogic Server Administration Console Online Help*.
3. Create credential mappings. See "Create PKI Credential Mappings" in the *Oracle WebLogic Server Administration Console Online Help*.

4.9.1 PKI Credential Mapper Attributes

To configure the PKI Credential Mapping provider, set values for these attributes. See "Configure Credential Mapping Providers" in the *Oracle WebLogic Server Administration Console Online Help*.

- Keystore Provider—A keystore provider for the Java Security API. If no value is specified, the default provider class is used.
- Keystore Type— JKS (the default) or PKCS12.
- Keystore Pass Phrase—Password used to access the keystore
- Keystore File Name—Location of the keystore relative to the directory where the server was started.

In addition, two optional attributes determine how the PKI Credential Mapping provider locates credential mappings in cases where the exact resource or subject may not be available:

- Use Resource Hierarchy—A credential is located by traversing up the resource hierarchy for each type of resource. The search for all possible PKI credentials will start from the specific resource and will walk up the resource hierarchy to find all possible matches. This attribute is enabled by default.
- Use Initiator Group Names—When a subject is passed to the PKI Credential Mapper provider, a credential is located by examining the groups of which the initiator is a member. This is enabled by default.

4.9.2 Credential Actions

Optionally, you can label a credential mapping with a credential action. You can do this in the Administration Console when you create the credential mapping. The credential action is an arbitrary string that distinguishes credential mappings used in different circumstances. For example, one credential mapping could decrypt a message from a remote resource and another credential mapping could sign messages to be sent to the same resource. The subject initiator and the target resource are not sufficient to distinguish these two credential mappings. You can use the credential action to label one of these credential mappings something like `decrypt` and the other one `sign`.

Then, the container calling the PKI Credential Mapping provider can provide the desired credential action value in the `ContextHandler` that is passed to the provider.

For information about adding credential actions to PKI credential mappings, see "Create PKI Credential Mappings" in the *Oracle WebLogic Server Administration Console Online Help*.

4.10 Configuring a SAML Credential Mapping Provider for SAML 1.1

This release of WebLogic Server includes two SAML Credential Mapping providers. SAML Credential Mapping Provider Version 2 provides greatly enhanced configuration options and is recommended for new deployments. SAML Credential Mapping Provider Version 1 is deprecated in WebLogic Server 9.1. A security realm can have not more than one SAML Credential Mapping provider, and if the security realm has both a SAML Credential Mapping provider and a SAML Identity Assertion provider, both must be of the same version. Do not use a Version 1 SAML provider in the same security realm as a Version 2 SAML provider. For information about configuring the SAML Credential Mapping Provider Version 1, see "Configuring a SAML Credential Mapping Provider" in *Securing WebLogic Server* in the WebLogic Server 9.0 documentation at `http://docs.oracle.com/docs/cd/E13222_01/wls/docs90/secmanage/providers.html#SAML_cred`.

For general information about WebLogic Server's support for SAML, see "Security Assertion Markup Language (SAML)" and "Single Sign-On with the WebLogic Security Framework" in *Understanding Security for Oracle WebLogic Server*. For information about how to use the SAML Credential Mapping provider in a SAML single sign-on configuration, see Chapter 7, "Configuring Single Sign-On with Web Browsers and HTTP Clients".

4.10.1 Configuring Assertion Lifetime

A SAML Assertion's validity is typically time-limited. The default time-to-live for assertions generated by the SAML Credential Mapping provider is specified by the `DefaultTimeToLive` attribute. You can override the default time-to-live for assertions generated for different SAML Relying Parties.

Normally, an assertion is valid from the `NotBefore` time, which defaults to (roughly) the time the assertion was generated, until the `NotOnOrAfter` time, which is calculated as (`NotBefore` + `TimeToLive`). To allow the Credential Mapper to compensate for clock differences between the source and destination sites, you can configure the SAML Credential Mapping provider's `DefaultTimeToLiveDelta` attribute. This time-to-live offset value is a positive or negative integer indicating how many seconds before or after "now" the assertion's `NotBefore` value should be set to. If you set a value for `DefaultTimeToLiveDelta`, then the assertion lifetime is still calculated as (`NotBefore` + `TimeToLive`), but the `NotBefore` value is set to (`now` + `TimeToLiveDelta`). For example, given the following settings:

```
DefaultTimeToLive = 120
DefaultTimeToLiveDelta = -30
```

an assertion when generated would have a lifetime of two minutes (120 seconds), starting 30 seconds before it is generated.

4.10.2 Relying Party Registry

When you configure WebLogic Server to act as a source of SAML security assertions, you need to register the parties that may request SAML assertions to be generated. For

each SAML Relying Party, you can specify the SAML profile used, details about the Relying Party, and the attributes expected in assertions for the Relying Party. For information, see:

- Section 7.3.2.3, "Configure Relying Parties"
- "Configure a SAML 1.1 Relying Party" in the *Oracle WebLogic Server Administration Console Online Help*

4.11 Configuring a SAML 2.0 Credential Mapping Provider for SAML 2.0

The SAML 2.0 Credential Mapping provider included with WebLogic Server generates SAML 2.0 assertions that can be used to assert identity in the following use cases:

- SAML 2.0 Web SSO Profile
- WS-Security SAML Token Profile version 1.1

The SAML 2.0 Credential Mapping provider generates the assertion types listed and described in Table 4–7.

Table 4–7 Assertion Types Supported by the SAML 2.0 Credential Mapping Provider

Assertion Type	Description
`bearer`	The subject of the assertion is the bearer of the assertion, subject to optional constraints on confirmation using attributes that may be included in the `<SubjectConfirmationData>` element of the assertion. Used for all assertions generated for the SAML 2.0 Web Browser SSO Profile and with the Web Service Security SAML Token Profile 1.1.
`sender-vouches`	The Identity Provider (different from the subject) vouches for the verification of the subject. The receiver must have a trust relationship with the Identity Provider. Used with the Web Service Security SAML Token Profile 1.1 only.
`holder-of-key`	The subject represented in the assertion uses an X.509 certificate that may not be trusted by the receiver to protect the integrity of the request messages. Used with the Web Service Security SAML Token Profile 1.1 only.

For general information about WebLogic Server's support for SAML 2.0, see "Security Assertion Markup Language (SAML)" and "Single Sign-On with the WebLogic Security Framework" in *Understanding Security for Oracle WebLogic Server*. For information about how to use the SAML 2.0 Credential Mapping provider in a SAML 2.0 single sign-on configuration, see Chapter 7, "Configuring Single Sign-On with Web Browsers and HTTP Clients". For information about specifying the confirmation method for assertions generated for web service Service provider partners, see "Using Security Assertion Markup Language (SAML) Tokens For Identity" in *Securing WebLogic Web Services for Oracle WebLogic Server*.

4.11.1 SAML 2.0 Credential Mapping Provider Attributes

Configuration of the SAML 2.0 Credential Mapping provider is controlled by setting attributes on the `SAML2CredentialMapperMBean`. You can access the `SAML2CredentialMapperMBean` using the WebLogic Scripting Tool (WLST), or through the Administration Console by selecting the **Security Realms** > *RealmName* > **Providers** > **Credential Mapping** page and creating or selecting SAML2CredentialMapper. For details about these attributes, see the description of the "SAML2CredentialMapperMBean" in the *MBean Reference for Oracle WebLogic Server*.

To configure the SAML 2.0 Credential Mapping provider, set the following attributes:

- Issuer URI

  Name of this security provider. The value that you specify should match the Entity ID specified in the SAML 2.0 General page that configures the per-server SAML 2.0 properties.

- Name Qualifier

  Used by the Name Mapper class as the security or administrative domain that qualifies the name of the subject. This provides a means to federate names from disparate user stores while avoiding the possibility of subject name collision.

- Assertion life time

  Values that limit the life time of generated assertions during which they may be used. Expired assertions cannot be made available for use.

- Web service assertion signing key alias and passphrase

  Used for signing generated assertions.

- Custom name mapper class

  The custom Java class that overrides the default SAML 2.0 Credential Mapping provider name mapper class, which maps Subjects to identity information contained in the assertion.

- Generate attributes

  Specifies whether group membership information associated with the authenticated Subject is included in generated assertions.

4.11.2 Service Provider Partners

When you configure WebLogic Server to act as an Identity Provider, you need to create and configure the Service Provider partners for whom SAML 2.0 assertions are generated. When an Identity Provider site needs to generate an assertion, the SAML 2.0 Credential Mapping provider determines the Service Provider partner for whom the assertion must be generated, and generates it according to the configuration of that Service Provider partner.

The way in which you configure a Service Provider partner, and the specific information you configure for that partner, depends upon whether the partner is used for web single sign-on or web services. Configuring a web single sign-on Service Provider partner consists of importing that partner's metadata file and establishing additional basic information about that partner, such as the following:

- Determining whether SAML documents, such as authentication responses, SAML artifacts, and artifact requests, must be signed
- Certificates used for validating signed documents received from this partner
- The binding to be used for sending SAML artifacts to this partner
- The client user name and password used by this partner when connecting to the local site's binding

For details about configuring a Service Provider partner for web single sign-on, see:

- Section 7.4.3.3, "Create and Configure Web Single Sign-On Service Provider Partners"

- "Create a SAML 2.0 Web Single Sign-on Service Provider partner" in the *Oracle WebLogic Server Administration Console Online Help*

Configuring a Web service Service Provider partner does not use a metadata file, but does consist of establishing the following information about that partner:

- Audience URIs, which specify an audience restriction to be included in assertions generated for this partner

  In WebLogic Server, the Audience URI attribute is overloaded to also include the partner lookup string, which is required by the web service run time to discover the partner. See Section 4.11.2.1, "Partner Lookup Strings Required for Web Service Partners".

- Custom name mapper class that overrides the default name mapper and that is to be used specifically with this partner
- Values that specify the life span attributes of assertions generated for this partner
- Confirmation method for assertions received from this partner

For more information about configuring web service Service Provider partners, see "Create a SAML 2.0 Web service Service Provider partner" in the *Oracle WebLogic Server Administration Console Online Help*.

4.11.2.1 Partner Lookup Strings Required for Web Service Partners

For web service Service Provider partners, you also configure Audience URIs. In WebLogic Server, the Audience URI attribute is overloaded to perform two distinct functions:

- Specify an audience restriction that consists of the target service URL, per the OASIS SAML 2.0 specification.
- Contain a partner lookup string, which is required at run time by WebLogic Server to discover the Service Provider partner for which a SAML 2.0 assertion needs to be generated.

The partner lookup string specifies an endpoint URL, which is used for partner lookup and can optionally also serve as an Audience URI restriction that is included in the generated assertion. The ability to specify a partner lookup string that is also an Audience URI eliminates the need to specify a given target URL twice: once for lookup, and again for audience restriction.

Note: You must configure a partner lookup string for a Service Provider partner so that partner can be discovered at run time by the web service run time.

4.11.2.1.1 Lookup String Syntax The partner lookup string has the following syntax:

`[target:`*`char`*`:]<`*`endpoint-url`*`>`

In this syntax, `target:`*`char`*`:` is a prefix that designates the partner lookup string, where *`char`* represents one of three special characters: a hyphen, plus sign, or asterisk (-, +, or *). This prefix determines how partner lookup is performed, as described in Table 4–8.

Table 4–8 Service Provider Partner Lookup String Syntax

Lookup String	Description
`target:-:<endpoint-url>`	Specifies that partner lookup is conducted for an exact match of the URL, `<endpoint-url>`. For example, `target:-:http://www.avitek.com:7001/myserver/myservicecontext/myservice-endpoint` specifies the endpoint that can be matched to this Service Provider, for which an assertion should be generated. This form of partner lookup string excludes the endpoint URL from being added as an Audience URI in the generated assertion.
`target:+:<endpoint-url>`	Specifies that partner lookup is conducted for an exact match of the URL, `<endpoint-url>`. Using the plus sign (+) in the lookup string results in the endpoint URL being added as an Audience URI in the assertion generated for this Service Provider partner.
`target:*:<endpoint-url>`	Specifies that partner lookup is conducted for an initial-string pattern match of the URL, `<endpoint-url>`. For example, `target:*:http://www.avitek.com:7001/myserver` specifies that any endpoint URL beginning with `http://www.avitek.com:7001/myserver` can be matched to this Service Provider, such as: `http://www.avitek.com:7001/myserver/contextA/endpointA` and `http://www.avitek.com:7001/myserver/contextB/endpointB`. If more than one Service Provider partner is discovered that is a match for the initial string, the partner with the longest initial string match is selected. This form of partner lookup string excludes the endpoint URL from being added as an Audience URI in the generated assertion.

Note: Configuring one or more partner lookup strings for a Service Provider partner is required in order for that partner to be discovered at run time. If this partner cannot be discovered, no assertions for this partner can be generated.

If you configure an endpoint URL without using the target lookup prefix, it will be handled as a conventional Audience URI that must be contained in assertions generated for this Service Provider partner. (This also enables backwards-compatibility with existing Audience URIs that may be configured for this partner.)

4.11.2.1.2 Specifying Default Partners To support the need for a default Service Provider partner entry, one or more of the default partner's Audience URI entries may contain a wildcard match that works for all targets. The actual wildcard URI may depend on the specific format used by the web service run time. For example:

- `target:*:http://`
- `target:*:https://`

4.11.2.2 Management of Partner Certificates

The SAML 2.0 Credential Mapping provider manages a set of trusted certificates for each partner configured for web single sign-on. Whenever a signed authentication or artifact request is received during a message exchange with a partner, the trusted certificate is used to verify the partner's signature. Partner certificates are used for the following purposes:

- To validate trust when the SAML 2.0 Credential Mapping provider receives a signed authentication request or artifact request.
- To validate trust in a Service Provider partner that is retrieving a SAML artifact from the Artifact Resolution Service (ARS) via an SSL connection.

From the Administration Console, you can view a web single sign-on Service Provider partner's signing certificate and transport layer client certificate in the partner management pages of the configured SAML 2.0 Credential Mapping provider.

4.11.2.3 Java Interface for Configuring Service Provider Partner Attributes

For details about the available operations on web service partners, see the "com.bea.security.saml2.providers.registry.Partner Java" interface in the *Java API Reference for Oracle WebLogic Server*.

4.12 Configuring the Certificate Lookup and Validation Framework

WebLogic Server may receive digital certificates as part of Web services requests, two-way SSL, or other secure interactions. To validate these certificates, WebLogic Server includes a Certificate Lookup and Validation (CLV) framework, whose function is to look up and validate X.509 certificate chains. The key elements of the CLV framework are the CertPathBuilder and the CertPathValidators. The CLV framework requires one and only active CertPathBuilder which, given a reference to a certificate chain, finds the chain and validates it, and zero or more CertPathValidators which, given a certificate chain, validates it.

When WebLogic Server receives a certificate, the CLV framework uses the security provider configured as the CertPathBuilder to look up and validate the certificate chain. If the certificate chain is found and valid, the framework then calls each configured CertPathValidator, in the order the administrator configured them, to perform extra validation on the chain. The chain is only valid if the builder and all the validators successfully validate it.

A chain is valid only if all of the following are true:

- The certificates in the chain have signed each other properly.
- The chain terminates in a certificate that is one of the server's trusted CAs.
- The chain honors the basic constraints rules (for example, no certificate in the chain has been issued by a certificate that is not allowed to issue certificates).
- The certificates in the chain have not expired.

WebLogic Server includes two CLV security providers: the WebLogic CertPath provider (which acts as both a CertPathBuilder and a CertPathValidator), described in Section 4.12.1, "CertPath Provider". and the Certificate Registry, described in Section 4.12.2, "Certificate Registry". Use just the WebLogic CertPath provider if you want to use trusted CA-based validation of the full certificate chain. Use just the Certificate Registry if you want only to validate that certificates are registered. Use both, designating the Certificate Registry as the current builder, if you want to use both types of validation.

For more information about certificate lookup and validation, see Chapter 11, "Configuring Identity and Trust".

4.12.1 CertPath Provider

The default security realm in WebLogic Server is configured with the WebLogic CertPath provider. The CertPath provider serves two functions: CertPathBuilder and CertPathValidator. The CertPath provider receives an end certificate or a certificate chain. It uses the server's list of trusted CAs to complete the certificate chain, if necessary. After building the chain, the CertPath provider validates the chain, checking the signatures in the chain, ensuring that the chain has not expired, checking the chain's basic constraints, and verifying that the chain terminates in a certificate issued by one of the server's trusted CAs.

The WebLogic CertPath provider requires no configuration, other than its Current Builder attribute, which indicates whether the CertPath provider should be used as the active certificate chain builder.

4.12.2 Certificate Registry

The Certificate Registry is a security provider that allows you to explicitly register the list of trusted certificates that are allowed to access WebLogic Server. If you configure a Certificate Registry as part of your security realm, then only certificates that are registered in the Certificate Registry will be considered valid. The Certificate Registry provides an inexpensive mechanism for performing revocation checking. By removing a certificate from the Certificate Registry, you can invalidate a certificate immediately. The registry is stored in the embedded LDAP server.

The Certificate Registry is both a CertPath Builder and a CertPath Validator. In either case, the Certificate Registry ensures that the chain's end certificate is stored in the registry, but does no other validation. If you use the Certificate Registry as your security realm's CertPath Builder and you also want to use the WebLogic CertPath provider or another security provider to perform full chain validation, make sure that you register the intermediate and root CAs in each server's trust keystore, and the end certificates in the Certificate Registry.

The default security realm in WebLogic Server does not include a Certificate Registry. Once you configure a Certificate Registry, you can use the WebLogic Administration Console to add, remove, and view certificates in the registry. You can export a certificate from a keystore to a file, using the Java keytool utility. You can import a certificate that is a PEM or DER file in the file system into the Certificate Registry using the console. You can also use the Console to view the contents of a certificate, including its subject DN, issuer DN, serial number, valid dates, fingerprints, etc.

See "Configure Certification Path providers" in the *Oracle WebLogic Server Administration Console Online Help*.

4.13 Configuring a WebLogic Keystore Provider

Note: The WebLogic Keystore provider is deprecated. It is only supported for backward compatibility. Use Java KeyStores (JKS) instead. All of the functionality that was supported by the WebLogic Keystore provider is available through use of Java KeyStores.

For information about configuring the WebLogic Keystore provider, see "Configure keystores" in the *Oracle WebLogic Server Administration Console Online Help*.

5 Configuring Authentication Providers

This chapter describes how to configure the Authentication security providers supplied by WebLogic Server. Most of them work in similar fashion: given a username and password credential pair, the provider attempts to find a corresponding user in the provider's data store. These Authentication providers differ primarily in what they use as a data store: one of many available LDAP servers, a SQL database, or other data store. In addition to these username/password based security providers, WebLogic Server includes identity assertion Authentication providers, which use certificates or security tokens, rather than username/password pairs, as credentials.

This chapter includes the following sections:

5.1 Choosing an Authentication Provider

Authentication is the process whereby the identity of users and system processes are proved or verified. Authentication also involves remembering, transporting, and making identity information available to various components of a system when that information is needed.

The WebLogic Server security architecture supports: password-based and 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.

WebLogic Server offers the following types of Authentication providers:

- The *WebLogic Authentication provider*, also known as the *DefaultAuthenticator*, accesses user and group information in WebLogic Server's embedded LDAP server.

- The *Oracle Internet Directory Authentication provider* accesses users and groups in Oracle Internet Directory, an LDAP version 3 directory.
- The *Oracle Virtual Directory Authentication provider* accesses users and groups in Oracle Virtual Directory, an LDAP version 3 enabled service.
- *LDAP Authentication providers* access external LDAP stores. You can use an LDAP Authentication provider to access any LDAP server. WebLogic Server provides LDAP Authentication providers already configured for Open LDAP, Sun iPlanet, Microsoft Active Directory, and Novell NDS LDAP servers.
- *RDBMS Authentication providers* access external relational databases. WebLogic Server provides three RDBMS Authentication providers: SQL Authenticator, Read-only SQL Authenticator, and Custom RDBMS Authenticator.
- The *WebLogic Identity Assertion provider* validates X.509 and IIOP-CSIv2 tokens and optionally can use a user name mapper to map that token to a user in a WebLogic Server security realm.
- The *SAML Authentication provider*, which authenticates users based on Security Assertion Markup Language 1.1 (SAML) assertions.
- The *Negotiate Identity Assertion provider*, which uses Simple and Protected Negotiate (SPNEGO) tokens to obtain Kerberos tokens, validates the Kerberos tokens, and maps Kerberos tokens to WebLogic users.
- The *SAML Identity Assertion provider*, which acts as a consumer of SAML security assertions. This enables WebLogic Server to act as a SAML destination site and supports using SAML for single sign-on.

In addition, you can use:

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

5.2 Using More Than One Authentication Provider

Each security realm must have at least one Authentication provider configured. The WebLogic Security Framework supports 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.

How you configure multiple Authentication providers can affect the overall outcome of the authentication process. Configure the JAAS Control Flag for each Authentication provider to set up login dependencies between Authentication providers and allow single-sign on between providers. See Section 5.2.1, "Setting the JAAS Control Flag Option".

Authentication providers are called in the order in which they were configured in the security realm. Therefore, use caution when configuring Authentication providers. You can use the WebLogic Administration Console to re-order the configured Authentication providers, thus changing the order in which they are called. See Section 5.2.2, "Changing the Order of Authentication Providers".

5.2.1 Setting the JAAS Control Flag Option

When you configure multiple Authentication providers, use the JAAS Control Flag for each provider to control how the Authentication providers are used in the login sequence. You can set the JAAS Control Flag in the WebLogic Administration Console. See "Set the JAAS control flag" in the *Oracle WebLogic Server Administration Console Online Help*. You can also use the WebLogic Scripting Tool or Java Management Extensions (JMX) APIs to set the JAAS Control Flag for an Authentication provider.

JAAS Control Flag values are:

- REQUIRED—The Authentication provider is always called, and the user must pass its authentication test. However, regardless of whether authentication succeeds or fails, authentication still continues down the list of providers.
- REQUISITE—The Authentication provider is always called, and the user is required to pass its authentication test.
  - If authentication succeeds, subsequent providers are executed but can fail (except for REQUIRED Authentication providers).
  - If authentication fails, control is returned to the caller and no subsequent Authentication provider down the list is executed.
- SUFFICIENT—The user is not required to pass the authentication test of the Authentication provider.
  - If authentication succeeds, control is returned to the caller and no subsequent Authentication provider down the list is executed.
  - If authentication fails, authentication continues down the list of providers.

  Any REQUIRED or REQUISITE Authentication provider in the list must pass its own authentication test. If no REQUIRED or REQUISITE Authentication provider is in the list, then the authentication test of at least one OPTIONAL or SUFFICIENT Authentication provider must pass.
- OPTIONAL—The user is not required to pass the authentication test of the Authentication provider. Regardless of whether authentication succeeds or fails, authentication continues down the list of providers.

The overall authentication of the user succeeds only if all REQUIRED and REQUISITE Authentication providers configured in the realm succeed. Note also:

- If a SUFFICIENT Authentication provider is configured and succeeds, then only the REQUIRED and REQUISITE Authentication providers prior to that SUFFICIENT Authentication provider need to have succeeded for the overall authentication to succeed.
- If no REQUIRED or REQUISITE Authentication providers are configured in the security realm, then at least one SUFFICIENT or OPTIONAL Authentication provider must succeed.

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

Note: As part of the startup process, WebLogic Server must be able to initialize all security providers that are configured in the security realm, including any Authentication providers that have a JAAS Control Flag set to OPTIONAL. If the initialization process for any security provider cannot be completed, WebLogic Server fails to boot, and an error message similar to the following is displayed:

```
<BEA-090870> <The realm "myrealm" failed to be loaded:
```

5.2.2 Changing the Order of Authentication Providers

The order in which WebLogic Server calls multiple Authentication providers can affect the overall outcome of the authentication process. The Authentication Providers table lists the authentication providers in the order in which they will be called. By default, Authentication providers are called in the order in which they were configured. You can use the Administration Console to change the order of Authentication providers. Select the **Reorder** button on the **Security Realms** > *RealmName* > **Providers** > **Authentication** page in the Administration Console to change the order in which Authentication providers are called by WebLogic Server and listed in the console.

See "Re-order Authentication Providers" in the *Oracle WebLogic Server Administration Console Online Help*.

5.3 Configuring the WebLogic Authentication Provider

The WebLogic Authentication provider (also called the DefaultAuthenticator) uses WebLogic Server's embedded LDAP server to store user and group membership information and, optionally, a set of user attributes such as phone number, email address, and so on. This provider allows you to create, modify, list, and manage users and group membership in the WebLogic Server Administration Console. By default, most configuration options for the WebLogic Authentication provider are already defined. You should need to configure a WebLogic Authentication provider only when creating a new security realm. However, note the following:

- The WebLogic Authentication provider is configured in the default security realm with the name `DefaultAuthenticator`.
- User and group names in the WebLogic Authentication provider are case insensitive. For information about creating and managing users and groups in the WebLogic Server Administration Console, see "Manage users and groups" in the *Oracle WebLogic Server Administration Console Online Help*.
- Ensure that all user names are unique.
- Specify the minimum length of passwords defined for users that are stored in the embedded LDAP server, which you can by means of the **Minimum Password Length** option that is available on the **Configuration** > **Provider Specific** page for the WebLogic Authentication provider.
- Users in the WebLogic Authentication provider can be modified to include a set of attributes. See Section 5.3.1, "Setting User Attributes".
- If you are using multiple Authentication providers, set the JAAS Control Flag to determine how the WebLogic Authentication provider is used in the authentication process. See Section 5.2, "Using More Than One Authentication Provider".

5.3.1 Setting User Attributes

After you have defined a user in the WebLogic Authentication provider, you can set or modify for that user one more of the attributes listed in Table 5–1. These attributes conform to the user schema for representing individuals in the inetOrgPerson LDAP object class, described in RFC 2798.

Table 5–1 Attributes that Can Be Set for a User

Attribute	Description
`c`	Two-letter ISO 3166 country code
`departmentnumber`	Code for department to which the user belongs
`displayname`	Preferred name of the user
`employeenumber`	Numeric or alphanumeric identifier assigned to the user
`employeetype`	Type of employment, which represents the employer to employee relationship
`facsimiletelephonenumber`	Facsimile (fax) telephone number
`givenname`	First name; that is, not surname (last name) or middle name
`homephone`	Home telephone number
`homepostaladdress`	Home postal address
`l`	Name of a locality, such as a city, county or other geographic region
`mail`	Electronic address of user (email)
`mobile`	Mobile telephone number
`pager`	Pager telephone number
`postaladdress`	Postal address at location of employment
`postofficebox`	Post office box
`preferredlanguage`	User's preferred written or spoken language
`st`	Full name of state or province
`street`	Physical location of user
`telephonenumber`	User's telephone number in organization
`title`	Title representing user's job function

When you set a value for an attribute, the attribute is added for the user. Likewise, if you subsequently delete the value of an attribute, the attribute is removed for the user. The set of available attributes is limited to the preceding list, however. The attribute names cannot be customized.

These attributes can be managed for a user by operations on the `UserAttributeEditorMBean`, or viewed via operations on the `UserAttributeReaderMBean`.

For more information about setting, modifying, or viewing the attributes for a user created in the WebLogic Authentication provider, see "Manage values for user attributes" in *Oracle WebLogic Server Administration Console Online Help*.

5.4 Configuring LDAP Authentication Providers

This section includes the following topics:

5.4.1 LDAP Authentication Providers Included in WebLogic Server

WebLogic Server includes the following LDAP Authentication providers:

- Oracle Internet Directory Authentication provider
- Oracle Virtual Directory Authentication provider
- iPlanet Authentication provider
- Active Directory Authentication provider
- Open LDAP Authentication provider
- Novell Authentication provider
- generic LDAP Authentication provider

Each LDAP Authentication provider stores user and group information in an external LDAP server. They differ primarily in how they are configured by default to match typical directory schemas for their corresponding LDAP server. For information about configuring the Oracle Internet Directory and Oracle Virtual Directory Authentication providers to match the LDAP schema for user and group attributes, see Section 5.4.8, "Configuring Users and Groups in the Oracle Internet Directory and Oracle Virtual Directory Authentication Providers".

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:

- Oracle Internet Directory
- Oracle Virtual Directory
- Oracle Unified Directory
- Sun iPlanet (also known as Oracle Directory Server Enterprise Edition)

- Active Directory shipped as part of the Microsoft Windows platform
- Open LDAP
- Novell Directory Service (NDS)

An LDAP Authentication provider can also be used to access other LDAP servers. However, you must either use the LDAP Authentication provider (`LDAPAuthenticator`) or choose a pre-defined LDAP provider and customize it. See Section 5.4.4, "Accessing Other LDAP Servers".

> **Note:** The Active Directory Authentication provider also supports Microsoft Active Directory Application Mode (ADAM) as a standalone directory server.

5.4.2 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, if one does not already exist. Make sure the LDAP user who will boot WebLogic Server is included in the group.

  The Active Directory LDAP directory has a default group called `Administrators`. Add the user who will be booting WebLogic Server to the `Administrators` group and define Group Base Distinguished Name (DN) so that the Administrators group is found.
- If you do not want to create an `Administrators` group in the LDAP directory (for example, because the LDAP directory uses the `Administrators` group for a different purpose), create a new group (or use an existing group) in the LDAP directory and include the user from which you want to boot WebLogic Server in that group. In the WebLogic Administration Console, assign that group the `Admin` role.

> **Note:** If the LDAP user who boots WebLogic Server is not properly added to a group that is assigned to the `Admin` role, and the LDAP authentication provider is the only authentication provider with which the security realm is configured, WebLogic Server cannot be booted.

5.4.3 Configuring an LDAP Authentication Provider: Main Steps

To configure an LDAP Authentication provider:

1. Choose an LDAP Authentication provider that matches your LDAP server and create an instance of the provider in your security realm. For information, see the following topics:
   - If you are using the WebLogic Server Administration Console, see "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*.
   - If you are using the WebLogic Scripting Tool (WLST), see "Managing Security Data (WLST Online)" in *Understanding the WebLogic Scripting Tool*. This section

also explains how to use WLST to switch from one LDAP authentication provider to another.

2. Configure the provider-specific attributes of the LDAP Authentication provider, which you can do through the Administration Console. For each LDAP Authentication provider, attributes are available to:

   a. Enable communication between the LDAP server and the LDAP Authentication provider. For a more secure deployment, Oracle recommends using the SSL protocol to protect communications between the LDAP server and WebLogic Server. Enable SSL with the `SSLEnabled` attribute.

   b. Configure options that control how the LDAP Authentication provider searches the LDAP directory.

      Note: The value you enter for `principal` must be an LDAP administrator who has the privilege to search users and groups in the corresponding LDAP server. If the LDAP administrator does not have privileges to search the LDAP server, an LDAP exception with error code 50 is generated.

   c. Specify where in the LDAP directory structure users are located.

   d. Specify where in the LDAP directory structure groups are located.

      Note: When specifying an LDAP search filter for users or groups using the following LDAPAuthenticatorMBean attributes, wildcards are accepted but they can have a negative performance impact on the LDAP server, particularly if you use a combination of them:

      - `AllUsersFilter`
      - `UserFromNameFilter`
      - `AllGroupsFilter`
      - `GroupFromNameFilter`

      For example, the following filter expression combines five wildcarded conditions, each condition using two asterisk wildcard characters:

      ```
      (|(cn=*wall*)(givenname=*wall*)(sn=*wall*)(cn=*wall*)(mail=*wall*))
      ```

      The preceding example filter would likely cause an unacceptable overhead on the corresponding LDAP server.

      Additionally, group names must not contain any trailing space characters.

   e. Define how members of a group are located.

   f. Set the name of the global universal identifier (GUID) attribute defined in the LDAP server.

Note: If you are configuring either the Oracle Internet Directory or Oracle Virtual Directory Authentication provider, see Section 5.4.8, "Configuring Users and Groups in the Oracle Internet Directory and Oracle Virtual Directory Authentication Providers". This section explains how to match the authentication provider attributes for users and groups to the LDAP directory structure.

g. Set timeout values for the connection to the LDAP server. You can specify two timeout values: a connection timeout, and a socket timeout.

The connection timeout, specified in the `LDAPServerMBean.ConnectTimeout` attribute for all LDAP Authentication providers, has a default value of zero. This default setting specifies no timeout limit, and can result in a slowdown in WebLogic Server execution if the LDAP servers configured for an LDAP Authentication provider are unavailable. In addition, if WebLogic Server has multiple LDAP Authentication providers configured, the failure to connect to one LDAP server may block the use of the other LDAP Authentication providers.

Oracle recommends that you set the `LDAPServerMBean.ConnectTimeout` attribute on the LDAP Authentication provider to a non-zero value; for example, 60 seconds. You can set this value via either the WebLogic Server Administration Console or WLST. You can also set this value in the `config.xml` file by adding the following configuration parameter for the LDAP Authentication provider:

```
<wls:connect-time>60</wls:connect-time>
```

Note: Oracle recommends that you do not edit the `config.xml` file directly.

The socket timeout, specified in the `-Dweblogic.security.providers.authentication.ldap.socketTimeout` JVM configuration option, sets the timeout in seconds for connecting to any one LDAP server specified in the `LDAPServerMBean.Host` attribute. The default value of the socket timeout is 0, which sets no socket timeout on the connection.

For information about the appropriate values to set for the connection timeout and socket timeout values for an LDAP Authentication provider, see Section 5.4.10, "Configuring Failover for LDAP Authentication Providers".

3. Configure performance options that control the cache for the LDAP server. Use the **Configuration** > **Provider Specific** and **Performance** pages for the provider in the Administration Console to configure the cache. See Section 5.4.13, "Improving the Performance of WebLogic and LDAP Authentication Providers".

Note: If the LDAP Authentication provider fails to connect to the LDAP server, or throws an exception, check the configuration of the LDAP Authentication provider to make sure it matches the corresponding settings in the LDAP server.

For more information, see the following sections:

5.4.4 Accessing Other LDAP Servers

The LDAP Authentication providers in this release of WebLogic Server are configured to work readily with the Oracle Internet Directory, Oracle Virtual Directory, SunONE (iPlanet), Active Directory, Open LDAP, and Novell NDS LDAP servers. You can use an LDAP Authentication provider to access other types of LDAP servers. Choose either the LDAP Authentication provider (`LDAPAuthenticator`) or the existing LDAP provider that most closely matches the new LDAP server and customize the existing configuration to match the directory schema and other attributes for your LDAP server.

If you are using Oracle Unified Directory, see Section 5.4.11, "Configuring an Authentication Provider for Oracle Unified Directory".

If you are using Active Directory, see Section 5.4.12, "Following Referrals in the Active Directory Authentication Provider".

5.4.5 Enabling an LDAP Authentication Provider for SSL

If you want to secure the connection between WebLogic Server and the LDAP server — for example, because the LDAP server requires it — you must do the following:

- Create and configure a custom trust keystore for use with the LDAP server
- Specify that the SSL protocol should be used by the LDAP Authentication provider when connecting to the LDAP server

To do this, complete the following steps:

1. Configure the LDAP Authentication provider. Make sure you select **SSLEnabled** on the **Configuration** > **Provider Specific** page.
2. Obtain the root certificate authority (CA) certificate for the LDAP server.
3. Create a trust keystore using the preceding certificate. For example, the following example shows using the `keytool` command to create the keystore `ldapTrustKS` with the root CA certificate `rootca.pem`.:

```
keytool -import -keystore ./ldapTrustKS -trustcacerts -alias oidtrust -file
```

```
rootca.pem -storepass TrustKeystorePwd -noprompt
```

For more information about creating a trust keystore, see Chapter 11, "Configuring Identity and Trust".

4. Copy the keystore to a location from which WebLogic Server has access.
5. Start the WebLogic Server Administration Console and navigate to the *server-name* > **Configuration** > **Keystores** page, where *server-name* is the WebLogic Server instance for which you are configuring this keystore.
6. If necessary, in the **Keystores** field, click **Change** to select the **Custom Identity and Custom Trust** configuration rules.
7. If the communication with the LDAP server uses 2-way SSL, configure the custom identity keystore, keystore type, and passphrase.
8. In **Custom Trust Keystore**, enter the path and file name of the trust keystore created in step 2.
9. In **Custom Trust Keystore Type**, enter `jks`.
10. In **Custom Trust Keystore Passphrase**, enter the password used when creating the keystore.
11. Reboot the WebLogic Server instance for changes to take effect.

For more information, see Chapter 12, "Configuring SSL". For more information about using the WebLogic Server Administration Console to configure keystores and enable SSL, see the following topics in the *Oracle WebLogic Server Administration Console Online Help*:

- "Configure identity and trust"
- "Set up SSL"
- "Configure two-way SSL"

5.4.6 Dynamic Groups and WebLogic Server

Many LDAP servers have a concept of dynamic groups or virtual groups. These are groups that, rather than consisting of a list of users and groups, contain some policy statements, queries, or code that define the set of users that belong to the group. Even if a group is marked dynamic, users must log out and log back in before any changes in their group memberships take effect. The term *dynamic* describes the means of defining the group and not any runtime semantics of the group within WebLogic Server.

5.4.7 Use of GUID and LDAP DN Data in WebLogic Principals

When a user is authenticated into WebLogic Server, an authentication provider creates a Subject with a set of user and group principals, which include the user and group names, respectively. The LDAP Authentication providers included in WebLogic Server also store the global universal identifier (GUID) and LDAP distinguished name (DN) data of users and groups as attributes of those principals. By default, WebLogic Server does not use the GUID or DN data in WebLogic principals. However, if the WebLogic domain is configured to use JAAS authorization, the GUID and DN data can be used in principal comparison operations that occur with Java policy decisions.

When configuring an LDAP Authentication provider, make sure that the name of the GUID attribute defined in the LDAP server is specified correctly for that provider. The default GUID attribute name for each LDAP Authentication provider included in

WebLogic Server is listed in Table 5–2.

Table 5–2 Name of GUID Attribute for LDAP Authentication Providers in WebLogic Server

Provider	Default GUID Attribute Name
WebLogic Authentication provider	`orclguid`[1]
Oracle Internet Directory Authentication provider	`orclguid`
Oracle Virtual Directory Authentication provider	`orclguid`[2]
Active Directory Authentication provider	`objectguid`[3]
iPlanet Authentication provider	`nsuniqueid`
Novell Authentication provider	`guid`
Open LDAP Authentication provider	`entryuuid`

[1] Note that the GUID attribute name for the embedded LDAP server cannot be modified, so the WebLogic Authentication provider does not have a corresponding attribute that is configurable.

[2] The GUID attribute name you configure for the Oracle Virtual Directory Authentication provider depends on whether Oracle Virtual Directory has a mapping of this attribute name. The mapping specifies a name for this attribute that is renamed from the one defined in the LDAP server with which Oracle Virtual Directory is connected. If a mapping exists, specify the name that is defined in the mapping. For example, if the GUID attribute is renamed in the mapping to `OVDguid`, configure the Oracle Virtual Directory Authentication provider to use `OVDguid` as the GUID attribute name. If a mapping does *not* exist, specify the name that is defined in the LDAP server. For example, if the LDAP server is Sun iPlanet Directory, and Sun iPlanet Directory defines the GUID attribute name as `nsuniqueid`, configure the Oracle Virtual Directory Authentication provider to use `nsuniqueid`. For more information, see "Understanding Oracle Virtual Directory Mapping" in *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

[3] The Active Directory Authentication provider also supports Microsoft Active Directory Application Mode (ADAM) as a standalone directory server.

For more information about how GUID and DN data in principal objects may be used, see Section 14.8, "Configuring a Domain to Use JAAS Authorization".

5.4.8 Configuring Users and Groups in the Oracle Internet Directory and Oracle Virtual Directory Authentication Providers

The following sections explain how to change default values in the Oracle Internet Directory and Oracle Virtual Directory Authentication providers that specify how users and groups are located in the LDAP server:

- Section 5.4.8.1, "Configuring User and Group Name Types"
- Section 5.4.8.2, "Configuring Static Groups"

5.4.8.1 Configuring User and Group Name Types

By default, the Oracle Internet Directory and Oracle Virtual Directory Authentication providers are configured to search users and groups in the LDAP directory using the class attribute types identified in Table 5–3:

Table 5–3 Default User Name and Group Name Attribute Types

Class	Attribute	Type
User object class	user name	`cn`
Group object class	group name	`cn`

If the user name attribute type, or group name attribute type, defined in the LDAP directory structure differs from the default settings for the Authentication provider you are using, you must change those provider settings. The following sections explain how to make those changes.

Note: Neither the Oracle Internet Directory Authentication provider nor Oracle Virtual Directory Authentication provider can read the name of a user or group from the LDAP server if the name contains an invalid character. Invalid characters are:

- Comma (,)
- Plus sign (+)
- Quotes (")
- Backslash (\)
- Angle brackets (< or >)
- Semicolon (;)

If either of these providers encounters a group or user name containing an invalid character, the name is ignored. (WebLogic Server in general does not support group names containing any of these invalid characters. See "Create groups" in the *Oracle WebLogic Server Administration Console Online Help*.)

5.4.8.1.1 Changing the User Name Attribute Type By default, the Oracle Internet Directory and Oracle Virtual Directory Authentication providers are configured with the user name attribute set to type `cn`. If the user name attribute type in the LDAP directory structure uses a different type — for example, `uid` — you must change the following Authentication provider attributes:

- `AllUsersFilter`
- `UserFromNameFilter`
- `UserNameAttribute`

For example, if the LDAP directory structure has the user name attribute type `uid`, the preceding Authentication provider attributes must be changed as shown in Table 5–4. The required changes are shown in **bold**.

Table 5–4 Changing the User Name Attribute Type for the User Object Class

Attribute Name	Default Setting	Required New Setting
`UserNameAttribute`	`cn`	**`uid`**
`AllUsersFilter`[1]	`(&(cn=*)(objectclass=person))`	`(&(`**`uid`**`=*)(objectclass=person))`
`UserFromNameFilter`[1]	`(&(cn=%u)(objectclass=person))`	`(&(`**`uid`**`=%u)(objectclass=person))`

[1] When specifying an LDAP search filter for users or groups, wildcards are accepted. However, using multiple asterisk wildcards, particularly for a user or group name attribute, have a negative performance impact on the LDAP server.

For information about configuring the user name attribute type, see the following topics in the *Oracle WebLogic Server Administration Console Online Help*:

- "Configure the Oracle Internet Directory Authentication provider"
- "Configure the Oracle Virtual Directory Authentication provider"

5.4.8.1.2 Changing the Group Name Attribute Type

By default, the Oracle Internet Directory and Oracle Virtual Directory Authentication providers are configured with the group name attribute type of `cn` for the static group object class and dynamic group object class. If the group name attribute type in the LDAP directory structure is different — for example, type `uid` is used — you must change the following Authention provider attributes:

- `AllGroupsFilter`
- `GroupFromNameFilter`
- `StaticGroupNameAttribute` (for static groups)
- `DynamicGroupNameAttribute` (for dynamic groups)

For example, if the LDAP directory structure of the group object class uses a group name attribute of type `uid`, you must change the Authentication provider attributes as shown in Table 5–5. The required changes are shown in **bold**.

Table 5–5 Required Changes for the Group Name Attribute Type

Attribute Name	Default Setting	Required Changes
`StaticGroupNameAttribute`	`cn`	**`uid`**
`DynamicGroupNameAttribute`	`cn`	**`uid`**
`AllGroupsFilter`[1]	`(&(cn=*)(|(objectclass=groupofUniqueNames)(objectclass=orcldynamicgroup)))`	`(&(`**`uid`**`=*)(|(objectclass=groupofUniqueNames)(objectclass=orcldynamicgroup)))`
`GroupFromNameFilter`[1]	`(|(&(cn=%g)(objectclass=groupofUniqueNames))(&(cn=%g)(objectclass=orcldynamicgroup)))`	`(|(&(`**`uid`**`=%g)(objectclass=groupofUniqueNames))(&(`**`uid`**`=%g)(objectclass=orcldynamicgroup)))`

[1] When specifying an LDAP search filter for users or groups, wildcards are accepted. However, using multiple asterisk wildcards, particularly for a user or group name attribute, have a negative performance impact on the LDAP server.

For more information about configuring group name attributes, see the following topics in the *Oracle WebLogic Server Administration Console Online Help*

- "Configure the Oracle Internet Directory Authentication provider"
- "Configure the Oracle Virtual Directory Authentication provider"

5.4.8.2 Configuring Static Groups

The Oracle Internet Directory and Oracle Virtual Directory Authentication providers are configured by default with the following settings for static groups:

- Static group object class name of `groupofuniquenames`
- Static member DN attribute of type `uniquemember`

However, the directory structure of the Oracle Internet Directory or Oracle Virtual Directory LDAP server with which you are configuring either of these Authentication providers may instead define the following for static groups:

- Static group object class name of `groupofnames`
- Static member DN attribute of type `member`

If the LDAP database schema contains the static group object class name of `groupofnames`, and the static member DN attribute of type `member`, you need to change the Oracle Internet Directory or Oracle Virtual Directory Authentication provider attribute settings as shown in Table 5–6. The required changes are shown in **bold**.

Table 5–6 Attribute Settings for Static Groups in the Oracle Internet Directory and Oracle Virtual Directory Authentication Providers

Attribute	Default Setting	Required Changes
`StaticGroupObjectClass`	`groupofuniquenames`	**`groupofnames`**
`StaticMemberDNAttribute`	`uniquemember`	**`member`**
`AllGroupsFilter`[1]	`(&(cn=*)(|(objectclass=groupofUniqueNames)(objectclass=orcldynamicgroup)))`	`(&(cn=*)(|(objectclass=`**`groupofnames`**`)(objectclass=orcldynamicgroup)))`
`GroupFromNameFilter`[1]	`(|(&(cn=%g)(objectclass=groupofUniqueNames))(&(cn=%g)(objectclass=orcldynamicgroup)))`	`(|(&(cn=%g)(objectclass=`**`groupofnames`**`))(&(cn=%g)(objectclass=orcldynamicgroup)))`

[1] When specifying an LDAP search filter for users or groups, wildcards are accepted. However, using multiple asterisk wildcards, particularly for a user or group name attribute, have a negative performance impact on the LDAP server.

For more information about configuring static groups, see the following topics in the *Oracle WebLogic Server Administration Console Online Help*:

- "Configure the Oracle Internet Directory Authentication provider"
- "Configure the Oracle Virtual Directory Authentication provider"

5.4.9 Example of Configuring the Oracle Internet Directory Authentication Provider

This section walks you through the process of configuring a sample Oracle Internet Directory Authentication provider and describes a quick method to verify the configuration.

The example does not describe how to configure your Oracle Internet Directory instance, and instead assumes that you are already knowledgeable about its configuration and are familiar with the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

For example, the default value for the **Dynamic Group Object Class** attribute is `orcldynamicgroup`. This object class is described in detail in the section "Dynamic Groups" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

Perform the following steps to configure this provider:

1. Create a new Oracle Internet Directory Authentication provider from the Administration Console:
   a. In the Console, navigate to the **Security Realms** > *RealmName* > **Providers** > **Authentication** page.
   b. Click **New** to add a new Authentication provider.
   c. Enter a name of your choice and choose `OracleInternetDirectoryAuthenticator` as the type.
   d. Click **OK**.
2. Configure the new Oracle Internet Directory Authentication provider:
   a. Click the name of the new provider you just created.
   b. On the **Common** page, set the **Control Flag** as needed (REQUIRED, REQUISITE, OPTIONAL or SUFFICIENT), as described in Section 5.2.1, "Setting the JAAS Control Flag Option".
   c. Navigate to the **Provider Specific** page.

d. Configure the **Connection** section with the Oracle Internet Directory server values you want to use. The port must be the Oracle Internet Directory LDAP port. For the purpose of this example, assume the following values:

```
Host: hostname.com
Port: 3060
Principal: cn=orcladmin
Credential: password
SSLEnabled is unchecked
```

e. Configure the **Users** section as per your Oracle Internet Directory configuration.

As described in Section 5.4.8.1.1, "Changing the User Name Attribute Type", pay particular attention to the fields **All Users Filter** and **User From Name Filter**. They **must** reflect the value of the **User Name Attribute** field.

The default value for **User Name Attribute** is `cn` and therefore the default values for the filter fields include `(&(cn=)...)` and `(&(*cn=%u)...)`, respectively. If you change the **User Name Attribute** value, you **must** replace it accordingly in the filter fields as well.

Note: If there are any leading or trailing white spaces in these filter field values, the users list may not be properly fetched from Oracle Internet Directory and you may not be able to authenticate using the Oracle Internet Directory Authentication provider.

For the purpose of example, assume the following values. Key changes are marked in **bold**.

```
User Base DN: cn=Users,dc=us,dc=oracle,dc=com
All Users Filter: (&(uid=*)(objectclass=person))
User From Name Filter: (&(uid=%u)(objectclass=person))
User Seearch Scope: subtree
User Name Attribute: uid
User Object Class: person
```

f. Configure the **Groups** section as per your Oracle Internet Directory configuration.

As described in Section 5.4.8.1.2, "Changing the Group Name Attribute Type", by default the Oracle Internet Directory Authentication provider is configured with the group name attribute type of `cn` for the static group object class and dynamic group object class. If the group name attribute type in the LDAP directory structure is different, you must change other Authentication provider attributes to match.

In addition, as described in Section 5.4.8.2, "Configuring Static Groups", the Oracle Internet Directory Authentication provider is configured by default with a Static group object class name of `groupofuniquenames` and a Static member DN attribute of type `uniquemember`. (This configuration is described in the section "Schema Elements for Creating Static Groups" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.)

If the LDAP database schema instead contains the static group object class name of `groupofnames`, and the static member DN attribute of type `member`, you need to change the attribute settings as shown in Table 5–6.

For the purpose of example, assume the following values. Key values that must match are marked in **bold**.

```
Group Base DN: cn=Groups,dc=us,dc=oracle,dc=com
All Groups Filter: (&(cn=*)(objectclass=groupofUniqueNames))

Static Group Name Attribute: cn
Static Group Object Class: groupofuniquenames
Static Member DN attribute: uniquemember
Static Group DNs from Member DN Filter:
(&(uniquemember=%M)(objectclass=groupofuniquenames))

Dynamic Group Name Attribute: (empty)
Dynamic Group Object Class:   (empty)
Dynamic Member URL Attribute: (empty)
User Dynamic Group DN Attribute:  (empty)
```

g. Configure all other sections as needed, using Section 5.4.3, "Configuring an LDAP Authentication Provider: Main Steps" for guidance. In this example, all of the default values are appropriate.

h. Save your changes.

3. If needed, order the providers to make the Oracle Internet Directory Authentication provider first in the list.

4. Restart the WebLogic Server to complete the changes.

5. Verify the setup.

   In the Administration Console, navigate to the **Security Realms** > *RealmName* > **Users and Groups** page. You should be able to see all users and groups that exist in the Oracle Internet Directory LDAP structure.

5.4.10 Configuring Failover for LDAP Authentication Providers

You can configure an LDAP provider to work with multiple LDAP servers and enable failover if one LDAP server is not available. Use the Host attribute (found in the Administration Console on the **Configuration** > **Provider Specific** page for the LDAP Authentication provider) to specify the names of the additional LDAP servers. Each host name may include a trailing space character and a port number. In addition, set the Parallel Connect Delay and Connection Timeout attributes for the LDAP Authentication provider:

- Parallel Connect Delay—Specifies the number of seconds to delay when making concurrent attempts to connect to multiple servers. An attempt is made to connect to the first server in the list. The next entry in the list is tried only if the attempt to connect to the current host fails. This setting might cause your application to block for an unacceptably long time if a host is down. If the value is greater than 0, another connection setup thread is started after the specified number of delay seconds has passed. If the value is 0, connection attempts are serialized.

- Connection Timeout—Specifies the maximum number of seconds to wait for the connection to the LDAP server(s) to be established. If the value is set to 0, the default, there is no maximum time limit and WebLogic Server waits until the TCP/IP layer times out to return a connection failure.

  If multiple hosts are set in the Host attribute, the connection timeout controls the total timeout value for attempts to connect to *all* the specified hosts.

Oracle recommends setting the connection timeout to a value of at least 60 seconds, depending upon the configuration of TCP/IP.

- Socket Timeout—Specifies the maximum number of seconds to wait for the connection to any *one* host specified in the Host attribute. The socket timeout is specified only using the `-Dweblogic.security.providers.authentication.ldap.socketTimeout=`*`seconds`* security parameter for the JVM in which WebLogic Server runs. The default value of the socket timeout is 0, which sets no socket timeout.

  Note that setting the socket timeout is not available in the WebLogic Server Administration Console. For information about the options for configuring WebLogic Server security parameters, see "Security" in *Command Reference for Oracle WebLogic Server*.

The following examples present scenarios that occur when an LDAP Authentication provider is configured for LDAP failover.

5.4.10.1 LDAP Failover Example 1

In the following scenario, an LDAP Authentication provider is configured with three servers in its Host attribute: `directory.knowledge.com:1050`, `people.catalog.com`, and `199.254.1.2`. The status of the LDAP servers is as follows:

- `directory.knowledge.com:1050` is down
- `people.catalog.com` is up
- `199.254.1.2` is up

WebLogic Server attempts to connect to `directory.knowledge.com`. After three seconds, or the socket connection throws an exception, the connect attempt times out and WebLogic Server attempts to connect to the next specified host (`people.catalog.com`). WebLogic Server then uses `people.catalog.com` as the LDAP Server for this connection. Otherwise, after another three seconds, WebLogic Server tries to connect to `199.254.1.2`. This process continues, but will fail if the overall LDAP server connection process exceeds 10 seconds.

Table 5–7 LDAP Configuration Example 1

LDAP Option	Value
Host	`directory.knowledge.com:1050 people.catalog.com 199.254.1.2`
Parallel Connect Delay	`0`
Connect Timeout	`10`
Socket Timeout	`3`

5.4.10.2 LDAP Failover Example 2

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

Table 5–8 LDAP Configuration Example 2

LDAP Option	Value
Host	`directory.knowledge.com:1050 people.catalog.com 199.254.1.2`
Parallel Connect Delay	`1`
Connect Timeout	`10`
Socket Timeout	`3`

5.4.11 Configuring an Authentication Provider for Oracle Unified Directory

To configure an Authentication provider for Oracle Unified Directory, complete the following steps:

1. In the Administration Console, navigate to the **Security Realms** > *RealmName* > **Providers** > **Authentication** page.
2. Click **New** to add a new Authentication provider.
3. Enter a name for the Authentication provider and choose `IPlanetAuthenticator` as the type.
4. Click **OK**.
5. In the **Security Realms** > *RealmName* > **Providers** > **Authentication** page, click the name of the iPlanet Authentication provider you created, and select the **Configuration** > **Provider Specific** page.
6. Configure the connection attributes for Oracle Unified Directory, as well as any other attributes as appropriate.
7. In the field labeled **GUID Attribute**, located at the bottom of the page, update the value of `nsuniqueid` to `entryuuid`.
8. Click **Save**.

5.4.12 Following Referrals in the Active Directory Authentication Provider

If Active Directory uses LDAP referrals, you must configure the Active Directory Authentication provider to follow those referrals by making sure that the `LDAPServerMBean.FollowReferrals` attribute is enabled. This attribute is enabled by default, but Oracle recommends that you make sure it is specifically enabled.

You can enable this attribute using WLST or the WebLogic Server Administration Console. If you are using the Administration Console, this attribute is available from the **Configuration** > **Provider Specific** page for the Active Directory Authentication provider.

5.4.13 Improving the Performance of WebLogic and LDAP Authentication Providers

To improve the performance of WebLogic and LDAP Authentication providers:

- Optimize the group membership caches used by the WebLogic and LDAP Authentication providers. See Section 5.4.13.1, "Optimizing the Group Membership Caches".
- Optimize the connection pool size and user cache. See Section 5.4.13.2, "Optimizing the Connection Pool Size and User Cache".
- Expose the Principal Validator cache for the security realm and increase its thresholds. See Section 5.4.13.4, "Optimizing the Principal Validator Cache".

- If you are using the Active Directory Authentication provider, configure it to perform group membership lookups using the `tokenGroups` option. The `tokenGroups` option holds the entire flattened group membership for a user as an array of system ID (SID) values. The SID values are specially indexed in the Active Directory and yield extremely fast lookup response. See Section 5.4.13.5, "Configuring the Active Directory Authentication Provider to Improve Performance."

5.4.13.1 Optimizing the Group Membership Caches

To optimize the group membership caches for WebLogic and LDAP Authentication providers, set the following attributes (found in the Administration Console on the LDAP Authentication provider's **Configuration** > **Provider Specific** and **Performance** pages):

- Group Membership Searching—Available from the Provider Specific page, this attribute controls whether group searches are limited or unlimited in depth. This option controls how deeply to search into nested groups. For configurations that use only the first level of nested group hierarchy, this option allows improved performance during user searches by limiting the search to the first level of the group.
  - If a limited search is defined, Max Group Membership Search Level must be defined.
  - If an unlimited search is defined, Max Group Membership Search Level is ignored.
- Max Group Membership Search Level—Available from the Provider Specific page, this attribute controls the depth of a group membership search if Group Membership Searching is defined. Possible values are:
  - 0—Indicates only direct groups will be found. That is, when searching for membership in Group A, only direct members of Group A will be found. If Group B is a member of Group A, the members will not be found by this search.
  - Any positive number—Indicates the number of levels to search. For example, if this option is set to 1, a search for membership in Group A will return direct members of Group A. If Group B is a member of Group A, the members of Group B will also be found by this search. However, if Group C is a member of Group B, the members of Group C will not be found by this search.
- Enable Group Membership Lookup Hierarchy Caching— Available from the Performance page, this attribute indicates whether group membership hierarchies found during recursive membership lookup are cached. Each subtree found will be cached. The cache holds the groups to which a group is a member. This setting only applies if Group Membership is enabled. By default, it is disabled.
- Max Group Hierarchies in Cache—Available from the Performance page, this attribute specifies the maximum size of the Least Recently Used (LRU) cache that holds group membership hierarchies. A value of 1024 is recommended. This setting only applies if Enable Group Membership Lookup Hierarchy Caching is enabled.
- Group Hierarchy Cache TTL—Available from the Performance page, this attribute specifies the number of seconds cached entries stay in the cache. The default is 60 seconds. A value of 6000 is recommended.

In planning your cache settings, bear in mind the following considerations:

- Enabling a cache involves a trade-off of performance and accuracy. Using a cache means that data is retrieved faster, but runs the risk that the data may not be the latest available.
- The time-to-live (TTL) setting how long you are willing to accept potentially stale data. This depends a lot on your particular business needs. If you frequently changes group memberships for users, then a long TTL could mean that group related changes won't show up for a while, and you may want a short TTL. If group memberships almost never change after a user is added, a longer TTL may be fine.
- The cache size is related to the amount of memory you have available, as well as the cache TTL. Consider the number of entries that might be loaded in the span of the TTL, and size the cache in relation to that number. A longer TTL will tend to require a larger cache size.

5.4.13.2 Optimizing the Connection Pool Size and User Cache

When configuring any of the LDAP Authentication providers, you can improve the performance of the connection between WebLogic Server and the LDAP server by optimizing the size of the LDAP connection pool and user cache. To make these optimizations, complete the following steps:

1. Set the LDAP connection pool size to 100 by using either of the following methods:
   - Define the following system property in the `setDomainEnv` script, which is located in the `bin` directory of the WebLogic domain:

     ```
     -Dweblogic.security.providers.authentication.LDAPDelegatePoolSize=100
     ```

   - In the WebLogic Server Administration Console, select the Provider Specific page for the LDAP authentication provider you are configuring (**Security Realms** > *myrealm* > **Providers** > **Authentication** > *your LDAP Authentication provider* > **Provider Specific**), and specify 100 in the field labeled **Connection Pool Size**.
2. Enable and enlarge the cache used with the LDAP server by completing the following steps in the WebLogic Server Administration Console:
   a. Select the Provider Specific page for the LDAP Authentication provider (**Security Realms** > *myrealm* > **Providers** > **Authentication** > *your LDAP Authentication provider* > **Provider Specific**).
   b. Scroll towards the bottom and make sure that **Cache Enabled** is checked.
   c. In the field labeled **Cache Size**, specify a value of 3200 KB.
   d. In the field labeled **Cache TTL**, specify a time-to-live value that matches the **Group Hierarchy Cache TTL** value (see Section 5.4.13.1, "Optimizing the Group Membership Caches"). A value of 6000 is recommended).
   e. Set the results timeout value for the LDAP server. On the current Provider Specific configuration page, specify a value of 1000 ms in the field labeled **Results Time Limit**.
3. Restart WebLogic Server for the changes to take effect.

5.4.13.3 Configuring Dynamic Groups in the iPlanet Authentication Provider to Improve Performance

Dynamic groups do not list the names of their members. Instead, the membership of the dynamic group is constructed by matching user attributes. Because group

membership needs to be computed dynamically for dynamic groups, there is a risk of performance problems for large groups. Configuring the iPlanet Authentication provider appropriately can improve performance where dynamic groups are involved.

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

```
UserDynamicGroupDNAttribute="wlsMemberOf"
DynamicGroupNameAttribute="cn"
DynamicGroupObjectClass=""
DynamicMemberURLAttribute=""
```

To set these attributes in the Administration Console:

1. Expand **Security Realms** > *RealmName* > **Providers** > **Authentication**.
2. On the Provider Specific tab for your iPlanet Authentication provider, set User Dynamic Group DN Attribute. Set Dynamic Group Object Class and Dynamic Member URL Attribute to null (delete anything in the fields) and leave Dynamic Group Name Attribute set to cn.

5.4.13.4 Optimizing the Principal Validator Cache

To improve the performance of a WebLogic or LDAP Authentication provider, the settings of the cache used by the WebLogic Principal Validation provider can be increased as appropriate. The Principal Validator cache used by the WebLogic Principal Validation provider caches signed WLSAbstractPrincipals. To optimize the performance of the Principal Validator cache, set these attributes for your security realm (found in the Administration Console on the **Configuration** > **Performance** page for the security realm):

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

5.4.13.5 Configuring the Active Directory Authentication Provider to Improve Performance

To configure an Active Directory Authentication provider to use the `tokenGroups` option, set the following attributes (found in the Administration Console on the Active Directory Authentication provider's **Configuration** > **Provider Specific** page):

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

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

- Enable SID to Group Lookup Caching—Indicates whether or not SID-to-group name lookup results are cached. This setting only applies if the Use Token Groups for Group Membership Lookup option is enabled.
- Max SID To Group Lookups In Cache—The maximum size of the Least Recently Used (LRU) cache for holding SID to group lookups. This setting applies only if both the Use Token Groups for Group Membership Lookup and Enable SID to Group Lookup Caching options are enabled.

5.5 Configuring RDBMS Authentication Providers

In WebLogic Server, an RDBMS Authentication provider is a username/password based Authentication provider that uses a relational database (rather than an LDAP directory) as its data store for user, password, and group information. WebLogic Server includes these RDBMS Authentication providers:

- SQL Authenticator—Uses a SQL database and allows both read and write access to the database. This Authentication provider is configured by default with a typical SQL database schema, which you can configure to match your database's schema. See Section 5.5.2, "Configuring the SQL Authentication Provider".
- Read-only SQL Authenticator—Uses a SQL database and allows only read access to the database. For write access, you use the SQL database's own interface, not the WebLogic security provider. See Section 5.5.3, "Configuring the Read-Only SQL Authenticator".
- Custom RDBMS Authenticator—Requires you to write a plug-in class. This may be a better choice if you want to use a relational database for your authentication data store, but the SQL Authenticator's schema configuration is not a good match for your existing database schema. See Section 5.5.4, "Configuring the Custom DBMS Authenticator".

For information about adding an RDBMS Authentication provider to your security realm, see "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*. Once you have created an instance of the RDBMS Authentication provider, configure it on the RDBMS Authentication provider's **Configuration** > **Provider Specific** page in the Administration Console.

5.5.1 Common RDBMS Authentication Provider Attributes

All three RDBMS Authentication providers include these configuration options.

5.5.1.1 Data Source Attribute

The Data Source Name specifies the WebLogic Server data source to use to connect to the database.

5.5.1.2 Group Searching Attributes

The Group Membership Searching and Max Group Membership Search Level attributes specify whether recursive group membership searching is unlimited or limited, and if limited, how many levels of group membership can be searched. For

example, if you specify that Group Membership Searching is LIMITED, and the Max Group Membership Search Level is 0, then the RDBMS Authentication providers will find only groups that the user is a direct member of. Specifying a maximum group membership search level can greatly increase authentication performance in certain scenarios, since it may reduce the number of DBMS queries executed during authentication. However, you should only limit group membership search if you can be certain that the group memberships you require are within the search level limits you specify.

Note: If the RDBMS contains cyclic groups, or groups that are defined to contain themselves, the RDBMS Authentication provider may be unable to complete the authentication process. Setting the Group Membership Searching and Max Group Membership Search Level attributes can help limit recursive group name lookups. However, the use of RDBMS Authentication providers with cyclic groups is not supported and must be avoided.

5.5.1.3 Group Caching Attributes

You can improve the performance of RDBMS Authentication providers by caching the results of group hierarchy lookups. Use of this cache can reduce the frequency with which the RDBMS Authentication provider needs to access the database. In the Administration Console, you can use the Performance page for your Authentication provider to configure the use, size, and duration of this cache. See "Security Realms: Security Providers: SQL Authenticator: Performance" in the *Oracle WebLogic Server Administration Console Online Help*.

5.5.2 Configuring the SQL Authentication Provider

For detailed information about configuring a SQL Authentication provider, see "Security Realms: Security Providers: SQL Authenticator: Provider Specific" in the *Oracle WebLogic Server Administration Console Online Help*. In addition to the attributes described in Section 5.5.1, "Common RDBMS Authentication Provider Attributes", the SQL Authentication provider has the following configurable attributes.

5.5.2.1 Password Attributes

The following attributes govern how the RDBMS Authentication provider and its underlying database handle user passwords:

- Plaintext Passwords Enabled
- Password Style Retained
- Password Style
- Password Algorithm

5.5.2.2 SQL Statement Attributes

SQL statement attributes specify the SQL statements used by the provider to access and edit the username, password, and group information in the database. With the default values in the SQL statement attributes, it is assumed that the database schema includes the following tables:

- users (username, password, [description])
- groupmembers (group name, group member)

- groups (group name, group description)

> **Note:** The tables referenced by the SQL statements must exist in the database; the provider will not create them. You can modify these attributes as needed to match the schema of your database. However, if your database schema is radically different from this default schema, you may need to use a Custom DBMS Authentication provider instead.

5.5.3 Configuring the Read-Only SQL Authenticator

For detailed information about configuring a Read-Only SQL Authentication provider, see "Security Realms: Security Providers: Read-Only SQL Authenticator: Provider Specific" in the *Oracle WebLogic Server Administration Console Online Help*. In addition to the attributes described in Section 5.5.1, "Common RDBMS Authentication Provider Attributes", the Read-Only SQL Authentication provider's configurable attributes include attributes that specify the SQL statements used by the provider to list the username, password, and group information in the database. You can modify these attributes as needed to match the schema of your database.

5.5.4 Configuring the Custom DBMS Authenticator

The Custom DBMS Authentication provider, like the other RDBMS Authentication providers, uses a relational database as its data store for user, password, and group information. Use this provider if your database schema does not map well to the SQL schema expected by the SQL Authenticator. In addition to the attributes described in Section 5.5.1, "Common RDBMS Authentication Provider Attributes", the Custom DBMS Authentication provider's configurable attributes include the following.

5.5.4.1 Plug-In Class Attributes

A Custom DBMS Authentication provider requires that you write a plug-in class that implements the `weblogic.security.providers.authentication.CustomDBMSAuthenticatorPlugin` interface. The class must exist in the system classpath and must be specified in the Plug-in Class Name attribute for the Custom DBMS Authentication provider. Optionally, you can use the Plugin Properties attribute to specify values for properties defined by your plug-in class.

5.6 Configuring a Windows NT Authentication Provider

The Windows NT Authentication provider uses account information defined for a Windows NT domain to authenticate users and groups and to permit Windows NT users and groups to be listed in the WebLogic Server Administration Console.

To use the Windows NT Authentication provider, create the provider in the Administration Console. In most cases, you should not need to do anything more to configure this Authentication provider. Depending on how your Windows NT domains are configured, you may want to set the Domain Controllers and Domain Controller List attributes, which control how the Windows NT Authentication provider interacts with the Windows NT domain.

Note: The Windows NT Authentication provider is deprecated as of WebLogic Server 10.0. Use one or more other supported authentication providers instead.

5.6.1 Domain Controller Settings

Usernames in a Windows NT domain can take several different forms. You may need to configure the Windows NT Authentication provider to match the form of usernames you expect your users to sign on with. A simple username is one that gives no indication of the domain, such as `smith`. Compound usernames combine a username with a domain name and may take a form like `domain\smith` or `smith@domain`.

If the local machine is not part of a Microsoft domain, then no changes to the Domain Controllers and Domain Controller List attributes are needed. On a stand-alone machine, the users and groups to be authenticated are defined only on that machine.

If the local machine is part of a Microsoft domain and is the domain controller for the local domain, then no changes are needed to the Domain Controller List attribute. Users defined on the local machine and the domain are the same in this case, so you can use the default Domain Controllers setting.

If the local machine is part of a Microsoft domain, but is not the domain controller for the local domain, then a simple username might be found on either the local machine or in the domain. In this case, consider the following:

- Do you want to prevent the users and groups from the local machine from being displayed in the Console when the local machine is part of a Microsoft domain?
- Do you want users from the local machine to be found and authenticated when a simple username is entered?

If the answer to either question is yes, then set the Domain Controller attribute to `DOMAIN`.

If you have multiple trusted domains, you may need to set the Domain Controller attribute to `LIST` and specify a Domain Controller List. Do this if:

- You require the users and groups for other trusted domains to be visible in the Console, or
- You expect that your users will be entering simple usernames and expect them to be located in the trusted domains (that is, users will sign on with a simple username like `smith`, not `smith@domain` or `domain\Smith`).

If either of these situations is the case, then set the Domain Controllers attribute to `LIST` and specify the names of the domain controllers in the Domain Controller List attribute for the trusted domains that you want to be used. Consider also whether to use explicit names for the local machine and local domain controller or if you want to use placeholders in the list for those. You can use the following placeholders in the Domain Controller List attribute:

- [Local]
- [LocalAndDomain]
- [Domain]

5.6.2 LogonType Setting

The proper value of the `LogonType` attribute in the Windows NT Authentication provider depends on the Windows NT logon rights of the users that you want to be able to authenticate:

- If users have the "logon locally" right assigned to them on the machines that will run WebLogic Server, then use the default value, `interactive`.
- If users have the "Access this computer from the Network" right assigned to them, then change the LogonType attribute to `network`.

You must assign one of these rights to users in the Windows NT domain or else the Windows NT Authentication provider will not be able to authenticate any users.

5.6.3 UPN Names Settings

UPN style usernames can take the form `user@domain`. You can configure how the Windows NT Authentication provider handles usernames that include the @ character, but which may not be UPN names, by setting the `mapUPNNames` attribute in the Windows NT Authentication provider.

If none of your Windows NT domains or local machines have usernames that contain the @ character other than UPN usernames, then you can use the default value of the `mapUPNNames` attribute, `FIRST`. However, you may want to consider changing the setting to `ALWAYS` in order to reduce the amount of time it takes to detect authentication failures. This is especially true if you have specified a long domain controller list.

If your Windows NT domains do permit non-UPN usernames with the @ character in them, then:

- if a username with the @ character is more likely to be a UPN username than a simple username, set the `mapUPNNames` attribute to `FIRST`.
- if a username with the @ character is more likely to be a simple username than a UPN username, set the `mapUPNNames` attribute to `LAST`.
- if a username is never in UPN format, set the `mapUPNNames` attribute to `NEVER`.

5.7 Configuring the SAML Authentication Provider

The SAML Authentication provider may be used in conjunction with the SAML 1.1 or SAML 2.0 Identity Assertion provider to do the following:

- Allow virtual users to log in via SAML

  If true, the SAML Identity Asserter will create user/group principals, with the possible result that the user is logged in as a virtual user — a user that does not correspond to any locally-known user.
- If the SAML Authentication provider is configured to run before other authentication providers, and has a JAAS Control Flag set to SUFFICIENT, this provider creates an authenticated subject using the user name and groups retrieved from a SAML assertion by the SAML Identity Assertion provider V2 or the SAML 2.0 Identity Assertion provider.

If the SAML Authentication provider is not configured, or if another authentication provider (e.g., the default LDAP Authentication provider) is configured before it and its JAAS Control Flag set is set to SUFFICIENT, then the user name returned by the SAML Identity Assertion provider is validated by the other authentication provider. In

the case of the default LDAP Authentication provider, authentication fails if the user does not exist in the identity directory.

Important: If you configure the SAML Authentication provider to allow virtual users to log in and gain access to a resource, make note of the following:

1. The resource must be configured with a security policy to control access. If the resource is unprotected, the subject created for the virtual user has no principals, which prevents access from being granted.
2. The protected resource must also use the default cookie `JSESSIONID`. If the resource uses a cookie name other than `JSESSIONID`, the subject's identity is not propagated to the resource.

For information about configuring security policies, see *Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

If you want groups from a SAML assertion, you must configure the SAML Authentication provider even if you want the LDAP Authentication provider to verify the user's existence. Otherwise, the groups with which the user is associated is derived from the LDAP directory and not with the groups in the assertion.

The SAML Authentication provider creates a subject only for users whose identities are asserted by either the SAML Identity Assertion provider V2 or SAML 2.0 Identity Assertion provider. The SAML Authentication provider ignores all other authentication or identity assertion requests.

5.8 Configuring the Password Validation Provider

WebLogic Server includes a Password Validation provider, which is configured by default in each security realm. The Password Validation provider manages and enforces a set of configurable password composition rules, and is automatically invoked by a supported authentication provider whenever a password is created or updated for a user in the realm. When invoked, the Password Validation provider performs a check to determine whether the password meets the criteria established by the composition rules. The password is then accepted or rejected as appropriate.

The following authentication providers can be used with the Password Validation provider:

- WebLogic Authentication provider
- SQL Authenticator provider
- LDAP Authentication provider
- Oracle Internet Directory Authentication Provider
- Oracle Virtual Directory Authentication Provider
- Active Directory Authentication provider
- iPlanet Authentication provider
- Novell Authentication provider
- Open LDAP Authentication provider

The following sections describe the composition rules that may be configured and explain how to create and configure an instance of the Password Validation provider in a security realm:

- Section 5.8.1, "Password Composition Rules for the Password Validation Provider"
- Section 5.8.2, "Using the Password Validation Provider with the WebLogic Authentication Provider"
- Section 5.8.3, "Using the Password Validation Provider with an LDAP Authentication Provider"
- Section 5.8.4, "Using WLST to Create and Configure the Password Validation Provider"

For information about configuring the Password Validation provider in the WebLogic Server Administration Console, see "Configure the Password Validation provider" in the *Oracle WebLogic Server Administration Console Online Help*.

5.8.1 Password Composition Rules for the Password Validation Provider

By default, the Password Validation provider is configured to require passwords that have a minimum length of eight characters. When used with one of the supported LDAP authentication providers listed in the preceding section, the Password Validation provider also requires that passwords meet the additional criteria listed in Table 5–9.

Table 5–9 Additional Password Composition Rules Required by Password Validation Provider When Used with an LDAP Authentication Provider

LDAP Authentication Provider	Additional Password Composition Requirement
■ Oracle Internet Directory Authentication provider ■ Oracle Virtual Directory Authentication provider	At least one of the characters in the password must be numeric.
■ WebLogic Authentication provider ■ LDAP Authentication provider ■ Active Directory Authentication provider ■ iPlanet Authentication provider ■ Novell Authentication provider ■ Open LDAP Authentication provider	At least one of the characters in the password must be non-alphabetic. For example, a numeric character, an asterisk (*), or an octothorpe (#).

The password composition rules you optionally can configure for the Password Validation provider include the following:

- User name policies — Rules that determine whether the password may consist of or contain the user's name, or the reverse of that name
- Password length policies — Rules for the minimum or maximum number of characters in a password (composition rules may specify both a minimum and maximum length)
- Character policies — Rules regarding the inclusion of the following characters in the password:
  - Numeric characters
  - Lowercase alphabetic characters
  - Uppercase alphabetic characters

- Non-alphanumeric characters

For information about the specific composition rules that may be configured for the Password Validation provider, including the settings for these rules that Oracle recommends for a production environment, see "System Password Validation Provider: Provider Specific" in the *Oracle WebLogic Server Administration Console Online Help*.

Caution: Setting password composition rules is only one component of hardening the WebLogic Server environment against brute-force password attacks. To protect user accounts, you should also configure user lockout. User lockout specifies the number of incorrect passwords that may be entered within a given interval of time before the user is locked out of his or her account. For more information, see Section 14.7, "Protecting User Accounts".

5.8.2 Using the Password Validation Provider with the WebLogic Authentication Provider

By default, the WebLogic Authentication provider requires a minimum password length of 8 characters, of which one is non-alphabetic. However, the minimum password length enforced by this provider can be customized. If the WebLogic Authentication provider and Password Validation provider are both configured in the security realm, and you attempt to create a password that does not meet the minimum length enforced by the WebLogic Authentication provider, an error is generated. For example, the following message is displayed in the Administration Console:

```
Error [Security:090285]password must be at least 8 characters long
Error Errors must be corrected before proceeding.
```

If the WebLogic Authentication provider rejects a password because it does not meet the minimum length requirement, the Password Validation provider is not called. To ensure that the Password Validator is always used in conjunction with the WebLogic Authentication provider, make sure that the minimum password length is the same for both providers.

Using the Administration Console, you can set the minimum password length for WebLogic Authentication provider by completing the following steps:

1. If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit**.
2. In the left pane, select **Security Realms** and click the name of the realm you are configuring (for example, *myrealm*).
3. Select **Providers** > **Authentication** and click **DefaultAuthenticator**.
4. Select **Configuration** > **Provider Specific** and enter the minimum password length in the field labeled Minimum Password Length.
5. Click **Save** to save your changes.
6. To activate these changes, in the Change Center, click **Activate Changes**.

For information about how to set the minimum password length in the Password Validation provider, see Section 5.8.4, "Using WLST to Create and Configure the Password Validation Provider".

5.8.3 Using the Password Validation Provider with an LDAP Authentication Provider

When the Password Validation provider and an LDAP Authentication provider (for example, Oracle Internet Directory Authentication provider) are configured in the security realm, passwords are validated through two separate policy checks: one from Password Validation provider, and the other from the LDAP server, which has its own password policy check. For example, Oracle Internet Directory has its own password validation mechanism, which is controlled by the LDAP server administrator. These two password validation mechanisms are separate, and each has its own set of password composition rules. If the composition rules are inconsistent, failures may occur in the WebLogic Server Administration Console when you try to create or reset a password, even if the rules for the Password Validation provider are enforced. Therefore you should make sure that the password composition rules for the Password Validation provider do not conflict with those for the LDAP server.

5.8.4 Using WLST to Create and Configure the Password Validation Provider

The Password Validation provider can be administered in the security realm via a WLST script that performs operations on the `SystemPasswordValidatorMBean`, described in the *MBean Reference for Oracle WebLogic Server*. You may create and configure the Password Validation provider from a single WLST script, or you may have separate scripts that perform these functions separately. The following topics explain how, providing sample WLST code snippets:

- Section 5.8.4.1, "Creating an Instance of the Password Validation Provider"
- Section 5.8.4.2, "Specifying the Password Composition Rules"

5.8.4.1 Creating an Instance of the Password Validation Provider

The Password Validation provider is created automatically in the security realm when you create a new domain. However, you can use WLST to create one as well, as shown in Example 5–1. This code does the following:

1. Gets the current realm and Password Validation provider.
2. Determines whether an instance of the Password Validator provider (named `SystemPasswordValidator`) has been created:
   - If the provider has been created, the script displays a message confirming its presence.
   - If the provider has not been created, the script creates it in the security realm and displays a message indicating that it has been created.

Example 5–1 Creating the System Password Validator

```
edit()
startEdit()

realm = cmo.getSecurityConfiguration().getDefaultRealm()
pwdvalidator = realm.lookupPasswordValidator('SystemPasswordValidator')

if pwdvalidator:
   print 'Password Validator provider is already created'

else:
# Create SystemPasswordValidator
 syspwdValidator = realm.createPasswordValidator('SystemPasswordValidator',

'com.bea.security.providers.authentication.passwordvalidator.SystemPasswordValidat
```

```
or')
 print "---  Creation of System Password Validator succeeded!"

save()
activate()
```

5.8.4.2 Specifying the Password Composition Rules

Example 5–2 shows an example of WLST code that sets the composition rules for the Password Validation provider. For information about the rule attributes that can be set in this script, see the description of the `SystemPasswordValidatorMBean` in the *MBean Reference for Oracle WebLogic Server*.

Example 5–2 Configuring the Password Composition Rules

```
edit()
startEdit()

# Configure SystemPasswordValidator
try:
  pwdvalidator.setMinPasswordLength(8)
  pwdvalidator.setMaxPasswordLength(12)
  pwdvalidator.setMaxConsecutiveCharacters(3)
  pwdvalidator.setMaxInstancesOfAnyCharacter(4)
  pwdvalidator.setMinAlphabeticCharacters(1)
  pwdvalidator.setMinNumericCharacters(1)
  pwdvalidator.setMinLowercaseCharacters(1)
  pwdvalidator.setMinUppercaseCharacters(1)
  pwdvalidator.setMinNonAlphanumericCharacters(1)
  pwdvalidator.setMinNumericOrSpecialCharacters(1)
  pwdvalidator.setRejectEqualOrContainUsername(true)
  pwdvalidator.setRejectEqualOrContainReverseUsername(true)
  print " --- Configuration of SystemPasswordValidator complete  ---"
except Exception,e:
        print e

save()
activate()
```

5.9 Configuring Identity Assertion Providers

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

WebLogic Server includes the following Identity Assertion providers:

- WebLogic Identity Asserter
- LDAP X.509 Identity Asserter
- Negotiate Identity Asserter
- SAML Identity Asserter (for SAML 1.1)
- SAML 2.0 Identity Asserter

Multiple Identity Assertion providers can be configured in a security realm, but none are required. Identity Assertion providers can support more than one token type, but only one token type per Identity Assertion provider can be active at a given time. In the Active Type field on the Provider Specific configuration page in the Administration Console, define the active token type. The WebLogic Identity Assertion provider supports identity assertion with X.509 certificates and CORBA Common Secure Interoperability version 2 (CSI v2). If you are using CSI v2 identity assertion, define the list of client principals in the Trusted Principals field.

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

With the WebLogic Identity Assertion provider, you can use a user name mapper to map the tokens authenticated by the Identity Assertion provider to a user in the security realm. For more information about configuring a user name mapper, see Section 4.8, "Configuring a WebLogic Credential Mapping Provider".

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

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

For more information see "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*. In addition, see the following sections:

- Section 5.9.1, "How an LDAP X509 Identity Assertion Provider Works"
- Section 5.9.2, "Configuring an LDAP X509 Identity Assertion Provider: Main Steps"
- Section 5.9.3, "Configuring a Negotiate Identity Assertion Provider"
- Section 5.9.4, "Configuring a SAML Identity Assertion Provider for SAML 1.1"
- Section 5.9.5, "Configuring a SAML 2.0 Identity Assertion Provider for SAML 2.0"
- Section 5.9.6, "Ordering of Identity Assertion for Servlets"
- Section 5.9.7, "Configuring Identity Assertion Performance in the Server Cache"
- Section 5.9.8, "Configuring a User Name Mapper"
- Section 5.9.9, "Configuring a Custom User Name Mapper"

5.9.1 How an LDAP X509 Identity Assertion Provider Works

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

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

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

5.9.2 Configuring an LDAP X509 Identity Assertion Provider: Main Steps

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

To use an LDAP X509 Identity Assertion provider:

1. Obtain certificates for users and put them in an LDAP Server. See Chapter 11, "Configuring Identity and Trust".

   A correlation must exist between the Subject DN in the certificate and the location of the object for that user in the LDAP server. The LDAP object for the user must also include configuration information for the certificate and the username that will be used in the Subject.
2. In your security realm, configure an LDAP X509 Identity Assertion provider. See "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*.
3. In the WebLogic Server Administration Console, configure the LDAP X509 Identity Assertion provider to find the LDAP object for the user in the LDAP directory given the certificate's Subject DN.
4. Configure the LDAP X509 Identity Assertion provider to search the LDAP server to locate the LDAP object for the user. This requires the following pieces of data.
   - A base LDAP DN from which to start searching. The Certificate Mapping option for the LDAP X509 Identity Assertion provider tells the identity assertion provider how to construct the base LDAP DN from the certificate's Subject DN. The LDAP object must contain an attribute that holds the certificate.
   - A search filter that only returns LDAP objects that match a defined set of options. The filter narrows the LDAP search. Configure User Filter Search to construct a search filter from the certificate's Subject DN.
   - Where in the LDAP directory to search for the base LDAP DN. The LDAP X509 Identity Assertion provider searches recursively (one level down). This value must match the values in the certificate's Subject DN.
5. Configure the Certificate Attribute attribute of the LDAP X509 Identity Assertion provider to specify how the LDAP object for the user holds the certificate. The LDAP object must contain an attribute that holds the certificate.

6. Configure the User Name Attribute 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.
7. Configure the LDAP server connection for the LDAP X509 Identity Assertion provider. The LDAP server information should be the same as the information defined for the LDAP Authentication provider configured in this security realm.
8. Configure an LDAP Authentication provider for use with the LDAP X509 Identity Assertion provider. The LDAP server information should be the same the information defined for the LDAP X509 Identity Assertion provider configured in Step 7. See Section 5.4, "Configuring LDAP Authentication Providers".

5.9.3 Configuring a Negotiate Identity Assertion Provider

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

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

For information about adding a Negotiate Identity Assertion provider to a security realm, see "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*. For information about using the Negotiate Identity Assertion provider with Microsoft client SSO, see Chapter 6, "Configuring Single Sign-On with Microsoft Clients"

Table 5–10 Negotiate Identity Asserter Attributes

Attribute	Description
Form Based Negotiation Enabled	Indicates whether the Negotiate Identity Assertion provider and servlet filter should negotiate when a Web application is configured for FORM authentication.
Active Types	The token type this Negotiate Identity Assertion provider uses for authentication. Available token types are `Authorization.Negotiate` and `WWW-Authenticate.Negotiate`. Ensure no other identity assertion provider configured in the same security realm has this attribute set to X509.

5.9.4 Configuring a SAML Identity Assertion Provider for SAML 1.1

The SAML Identity Assertion provider acts as a consumer of SAML 1.1 security assertions, allowing WebLogic Server to act as a destination site for using SAML 1.1 for single sign-on. The SAML Identity Assertion provider validates SAML 1.1 assertions by checking the signature and validating the certificate for trust in the certificate registry maintained by the provider. If so, identity is asserted based on the `AuthenticationStatement` contained in the assertion. The SAML Identity Assertion provider can also ensure that the assertion has not been previously used. The SAML Identity Assertion provider must be configured if you want to deploy a SAML Assertion Consumer Service on a server instance.

This release of WebLogic Server includes two SAML Identity Assertion providers for SAML 1.1. SAML Identity Asserter Version 2 provides greatly enhanced configuration options and is recommended for new deployments. SAML Identity Asserter Version 1 has been deprecated in WebLogic Server 9.1. A security realm can have not more than one SAML Identity Assertion provider, and if the security realm has both a SAML Identity Assertion provider and a SAML Credential Mapping provider, both must be of the same version. Do not use a Version 1 SAML provider in the same security realm as a Version 2 SAML provider. For information about configuring the SAML Identity Assertion provider Version 1, see "Configuring a SAML Identity Assertion Provider" in *Securing WebLogic Server* in the WebLogic Server 9.0 documentation.

For information about how to use the SAML Identity Assertion provider in a SAML single sign-on configuration, see Chapter 7, "Configuring Single Sign-On with Web Browsers and HTTP Clients". For general information about SAML support in WebLogic Server, see "Security Assertion Markup Language (SAML)" in *Understanding Security for Oracle WebLogic Server*.

5.9.4.1 Asserting Party Registry

When you configure WebLogic Server to act as a consumer of SAML security assertions, you need to register the parties whose SAML assertions will be accepted. For each SAML Asserting Party, you can specify the SAML profile used, details about the Asserting Party, and the attributes expected in assertions received from the Asserting Party. For information, see:

- Section 7.3.3.3, "Configuring Asserting Parties"
- "Configure a SAML 1.1 Asserting Party" in the *Oracle WebLogic Server Administration Console Online Help*.

5.9.4.2 Certificate Registry

The SAML Identity Assertion provider maintains a registry of trusted certificates. Whenever a certificate is received, it is checked against the certificates in the registry for validity. For each Asserting Party, the following certificates from that partner are contained in this registry:

- The certificate used for validating the signature of assertions received from this Asserting Party.
- The certificate used for verifying signatures on SAML protocol elements from this Asserting Party. This certificate must be set for the Browser/POST profile.
- The TLS/SSL certificate used for verifying trust in the Asserting Party when that partner is retrieving an artifact from the Assertion Retrieval Service (ARS) via an SSL connection.

You can add trusted certificates to the certificate registry through the Administration Console:

1. In the Console, navigate to the **Security Realms** > *RealmName* > **Providers** > **Authentication** page.
2. Click the name of the SAML Identity Assertion provider and open the **Management** > **Certificates** page.

On the **Management** > **Certificates** page, you can add, view, or delete certificates from the registry.

5.9.5 Configuring a SAML 2.0 Identity Assertion Provider for SAML 2.0

The SAML 2.0 Identity Assertion provider acts as a consumer of SAML 2.0 security assertions, allowing WebLogic Server to act as a Service Provider for the following:

- Web single sign-on
- WebLogic Web Services Security: accepting SAML tokens for identity through the use of the appropriate WS-SecurityPolicy assertions

The SAML 2.0 Identity Assertion provider does the following:

- Validates SAML 2.0 assertions by checking the signature and validating the certificate for trust based on data configured for the partner. The SAML 2.0 Identity Assertion provider then extracts the identity information contained in the assertion, and maps it to a local subject in the security realm.
- Optionally, extracts attribute information contained in an assertion that the SAML Authentication provider, if configured in the security realm, can use to determine the local groups in which the mapped subject belongs. (For more information, see Section 5.7, "Configuring the SAML Authentication Provider".)
- Optionally, verifies that an assertion's specified lifespan and re-use settings are properly valid, rejecting the assertion if it is expired or is not available for reuse.

Configuration of the SAML 2.0 Identity Assertion provider is controlled by setting attributes on the `SAML2IdentityAsserterMBean`. You can access the `SAML2IdentityAsserterMBean` using the WebLogic Scripting Tool (WLST), or through the Administration Console by using the **Security Realms** > *RealmName* > **Providers** > **Authentication** page and creating or selecting `SAML2IdentityAsserter`. For details about these attributes, see `SAML2IdentityAsserterMBean` in the *MBean Reference for Oracle WebLogic Server*.

For information about how to use the SAML 2.0 Identity Assertion provider in a SAML single sign-on configuration, see Chapter 7, "Configuring Single Sign-On with Web Browsers and HTTP Clients". For general information about SAML support in WebLogic Server, see "Security Assertion Markup Language (SAML)" in *Understanding Security for Oracle WebLogic Server*. For information about using the SAML 2.0 Identity Assertion provider in Web Service Security, see "Using Security Assertion Markup Language (SAML) Tokens For Identity" in *Securing WebLogic Web Services for Oracle WebLogic Server*.

5.9.5.1 Identity Provider Partners

When you configure WebLogic Server to act as a Service Provider, you create and configure the Identity Provider partners from whom SAML 2.0 assertions are received and validated. Configuring an Identity Provider partner consists of establishing basic information about that partner, such as the following:

- Partner name and general description
- Name mapper class to be used with this partner
- Whether to consume attribute statements included in assertions received from this partner
- Whether the identities contained in assertions received from this partner should be mapped to virtual users
- Certificates used for validating signed assertions received from this partner

The specific information you establish depends upon whether you are configuring the partner for web single sign-on or web services. Configuring a web single sign-on

Identity Provider partner also involves importing that partner's metadata file and establishing additional basic information about that partner, such as the following:

- Redirect URIs, which are URLs that, when invoked by an unauthenticated user, cause the user request to be redirected to that Identity Provider partner for authentication
- Whether SAML artifact requests received from this partner must be signed
- How SAML artifacts should be delivered to this partner

For details about configuring web single sign-on Identity Provider partners, see:

- Section 7.4.4.5, "Create and Configure Web Single Sign-On Identity Provider Partners"
- "Create a SAML 2.0 Web Single Sign-on Identity Provider partner" in the *Oracle WebLogic Server Administration Console Online Help*

Configuring a web service Identity Provider partner does not use a metadata file, but does consist of establishing the following information about that partner:

- Issuer URI, which is a string that uniquely identifies this Identity Provider partner, distinguishing it from other partners in your SAML federation
- Audience URIs, which specify an audience restriction to be included in assertions received from this partner

  In WebLogic Server, the Audience URI attribute is overloaded to also include the partner lookup string, which is required by the web service run time to discover the partner. See Section 5.9.5.1.1, "Partner Lookup Strings Required for Web Service Partners".
- Custom name mapper class that overrides the default name mapper and that is to be used specifically with this partner

For more information about configuring web service Service Provider partners, see "Create a SAML 2.0 Web Service Identity Provider partner" in the *Oracle WebLogic Server Administration Console Online Help*.

5.9.5.1.1 Partner Lookup Strings Required for Web Service Partners

For web service Identity Provider partners, you also configure Audience URIs. In WebLogic Server, the Audience URI attribute is overloaded to perform two distinct functions:

- Specify an audience restriction consisting of a target URL, per the OASIS SAML 2.0 specification.
- Contain a partner lookup string, which is required at run time by WebLogic Server to discover the Identity Provider partner for which a SAML 2.0 assertion needs to be validated.

The partner lookup string specifies an endpoint URL, which is used for partner lookup and can optionally also serve as an Audience URI restriction that must be included in the assertion received from this Identity Provider partner.

> **Note:** You must configure a partner lookup string for an Identity Provider partner so that partner can be discovered at run time by the web service run time.

Lookup String Syntax

The partner lookup string has the following syntax:

`[target:char:]<endpoint-url>`

In this syntax, `target:char:` is a prefix that designates the partner lookup string, where char represents one of three special characters: a hyphen, plus sign, or asterisk (`-`, `+`, or `*`). This prefix determines how partner lookup is performed, as described in Table 5–11.

Note: A WebLogic Server instance that is configured in the role of Service Provider always strips off the transport, host, and port portions of an endpoint URL that is passed in to the SAML 2.0 Identity Assertion provider. Therefore, the endpoint URLs you configure in any lookup string for an Identity Provider partner should contain only the portion of the URL that follows the host and port. For example, `target:*:/myserver/xxx`.

When you configure a Service Provider site, this behavior enables you to configure a single Identity Provider partner that can be used to validate all assertions for the same web service, regardless of the variations in the transport protocol (i.e., HTTP vs. HTTPS), host name, IP address, and port information across all the machines in a domain that host that web service.

Table 5–11 Identity Provider Partner Lookup String Syntax

Lookup String	Description
`target:-:<endpoint-url>`	Specifies that partner lookup is conducted for an exact match of the URL, `<endpoint-url>`. For example, `target:-:/myserver/myservicecontext/my-endpoint` specifies the endpoint that can be matched to this Identity Provider partner, for which an assertion should be validated. This form of partner lookup string excludes the endpoint URL from being added as an Audience URI for this Identity Provider partner.
`target:+:<endpoint-url>`	Specifies that partner lookup is conducted for an exact match of the URL, `<endpoint-url>`. Note: Using the plus sign (+) in the lookup string results in the endpoint URL being added as an Audience URI in the assertion received from this Identity Provider partner. Because this form of lookup string is unlikely to produce a match for an Identity Provider partner, it should be avoided.
`target:*:<endpoint-url>`	Specifies that partner lookup is conducted for an initial-string pattern match of the URL, `<endpoint-url>`. For example, `target:*:/myserver` specifies that any endpoint URL beginning with `/myserver` can be matched to this Identity Provider, such as: `/myserver/contextA/endpointA` and `/myserver/contextB/endpointB`. If more than one Identity Provider partner is discovered that is a match for the initial string, the partner with the longest initial string match is selected. This form of partner lookup string excludes the endpoint URL from being added as an Audience URI for this Identity Provider partner.

Notes: Configuring one or more partner lookup strings for an Identity Provider partner is required in order for that partner to be discovered at run time. If this partner cannot be discovered, no assertions for this partner can be validated.

If you configure an endpoint URL without using the target lookup prefix, it will be handled as a conventional Audience URI that must be contained in assertions received from this Identity Provider partner. (This also enables backwards-compatibility with existing Audience URIs that may be configured for this partner.)

Specifying Default Partners

To support the need for a default Identity Provider partner entry, one or more of the default partner's Audience URI entries may contain a wildcard match that works for all targets. For example, `target:*:/`.

5.9.5.1.2 Management of Partner Certificates The SAML 2.0 Identity Assertion provider manages the trusted certificates for configured partners. Whenever a certificate is received during an exchange of partner messages, the certificate is checked against the certificates maintained for the partner. Partner certificates are used for the following purposes:

- To validate trust when the Service Provider site receives a signed assertion or a signed SAML artifact request.
- To validate trust in an Identity Provider partner that is retrieving a SAML artifact from the Artifact Resolution Service (ARS) via an SSL connection.

The following certificates, which are obtained from each configured Identity Provider partner, are required:

- The certificate used to verify signed SAML documents received from the partner, such as assertions and artifact requests

  The certificate used to verify signed SAML documents in web single sign-on is included in the metadata file received from the Identity Provider partner. When configuring web service Identity Provider partners, you obtain this certificate from your partner and import it into this partner's configuration via the Assertion Signing Certificate tab of the partner management page in the Administration Console.

- The Transport Layer Security (TLS) client certificate that is used to verify the connection made by the partner to the local site's SSL binding for retrieving SAML artifacts (used in web single sign-on only)

  When configuring a web single sign-on Identity Provider partner, you must obtain the TLS client certificate directly from the partner. It is not automatically included in the metadata file. You can import this certificate into the configuration data for this partner via the Transport Layer Client Certificate tab of the partner management page in the Administration Console.

5.9.5.1.3 Java Interface for Configuring Identity Provider Partner Attributes Operations on web service partners are available in the `com.bea.security.saml2.providers.registry.Partner` Java interface.

5.9.6 Ordering of Identity Assertion for Servlets

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

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

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

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

5.9.7 Configuring Identity Assertion Performance in the Server Cache

When you use an Identity Assertion provider, either for an X.509 certificate or some other type of token, subjects are cached within the server. (A subject is a grouping of related information for a single entity (such as a person), including an identity and its security-related configuration options.) Caching subjects within the server greatly enhances performance for servlets and EJB methods with `<run-as>` tags as well as in

other situations where identity assertion is used but not cached in the HTTPSession, for example, in signing and encrypting XML documents).

> **Note:** Caching can violate the desired semantics.

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

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

To improve the performance of identity assertion, specify a higher value for this command-line argument.

> **Note:** As identity assertion performance improves, the Identity Assertion provider is less responsive to changes in the configured Authentication provider. For example, a change in the user's group will not be reflected until the subject is flushed from the cache and recreated. Setting a lower value for the command-line argument makes authentication changes more responsive at a cost for performance.

5.9.8 Configuring a User Name Mapper

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

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

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

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

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

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

For more information, see "Configure a user name mapper" in the *Oracle WebLogic Server Administration Console Online Help*.

5.9.9 Configuring a Custom User Name Mapper

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

For more information, see "Configure a custom user name mapper" in the *Oracle WebLogic Server Administration Console Online Help*.

6

Configuring Single Sign-On with Microsoft Clients

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

This chapter includes the following sections:

6.1 Overview of Single Sign-On with Microsoft Clients

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

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

For more information about Windows and Kerberos, see `http://technet.microsoft.com/en-us/library/bb742431.aspx`.

6.2 System Requirements for SSO with Microsoft Clients

To use SSO with Microsoft clients, you need to meet the requirements described in the following sections:

- Section 6.2.1, "Host Computer Requirements for Supporting SSO with Microsoft Clients"
- Section 6.2.2, "Client Computer Requirements for Supporting Microsoft Clients Using SSO"

6.2.1 Host Computer Requirements for Supporting SSO with Microsoft Clients

The host computer that supports SSO for Microsoft clients must meet the following system requirements:

- A version of Microsoft Windows that is supported by WebLogic Server for SSO with Microsoft clients

  For information about these supported versions, see the Oracle Fusion Middleware Supported System Configurations page on the Oracle Technology Network.

- Fully-configured Active Directory authentication service. Specific Active Directory requirements include:
  - User accounts for mapping Kerberos services
  - Service Principal Names (SPNs) for those accounts
  - Keytab files created and copied to the start-up directory in the WebLogic domain
- WebLogic Server installed and configured properly to authenticate through Kerberos, as described in this chapter

Oracle recommends encrypting the user accounts that are mapped to Kerberos services on the WebLogic Server host. However, the ability to encrypt these accounts imposes additional requirements. The specific requirements depend on the encryption algorithm used, as shown in Table 6–1. For each encryption algorithm listed in Table 6–1, see the Oracle Fusion Middleware Supported System Configurations page on the Oracle Technology Network for information about:

- The corresponding version of Microsoft Windows that is supported as the Active Directory platform.
- The specific versions of the Internet Explorer and Mozilla FireFox client types that are supported.

Table 6–1 Client Type Requirements for Using Encrypted User Accounts

Encryption Algorithm	Supported Client Type
DES	■ Internet Explorer ■ Mozilla FireFox ■ .NET Web service ■ Java SE client
AES-128, AES-256, and RC4	■ Internet Explorer ■ Mozilla FireFox ■ Java SE client[1]

[1] User accounts accessed with a Java SE client can also be encrypted with DES, AES-128, AES-256, and RC4 and defined in Active Directory running on a Microsoft Windows platform supported by WebLogic Server for this purpose.

6.2.2 Client Computer Requirements for Supporting Microsoft Clients Using SSO

The computer hosting a Microsoft client that uses SSO must meet the following requirements:

- An installation of Microsoft Windows
- Include one of the client types listed in the following table, which also includes links to the instructions for configuring those clients:

For the following client type . . .	. . . see the following topic
Internet Explorer[1]	Section 6.6.2, "Configuring an Internet Explorer Browser"
Mozilla FireFox[1]	Section 6.6.3, "Configuring a Mozilla Firefox Browser"
.NET Framework with properly configured web service client	Section 6.6.1, "Configuring a .NET Web Service"
Standalone Java SE client application	Section 6.6.4, "Configuring a Java SE Client Application"

[1] For information about the specific version supported for accessing user accounts that are defined in Active Directory and encrypted with AES-128, AES-256, or RC4, see the Oracle Fusion Middleware Supported System Configurations page on the Oracle Technology Network.

Clients must be logged on to a Microsoft Windows domain and have Kerberos credentials acquired from the Active Directory server in the domain. Local logins are not supported.

Note: For information about the versions of Microsoft Windows that are supported for hosting clients using SSO, and the encryption algorithms with which user accounts accessed by those clients can be defined in Active Directory, see the Oracle Fusion Middleware Supported System Configurations page on the Oracle Technology Network.

6.3 Single Sign-On with Microsoft Clients: Main Steps

Configuring SSO with Microsoft clients requires set-up procedures in the Microsoft Active Directory, the client, and the WebLogic domain. (These procedures are detailed in the sections that follow.)

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

To configure SSO with Microsoft clients:

1. Configure your network domain to use Kerberos. See Section 6.4, "Configuring Your Network Domain to Use Kerberos".
2. Create a Kerberos identification for WebLogic Server.
   a. Create a user account in the Active Directory for the host on which WebLogic Server is running.
   b. Create a service principal name (SPN) for this account.
   c. Create a user mapping and keytab file for this account.

   See Section 6.5, "Creating a Kerberos Identification for WebLogic Server".
3. Choose a Microsoft client (either a Web service or browser) or a Java SE client and configure it to use Windows Integrated Authentication. See Section 6.6, "Configuring Microsoft Clients to Use Windows Integrated Authentication".
4. Set up the WebLogic domain to use Kerberos authentication.
   a. Create a JAAS login file that points to the Active Directory server in the Microsoft domain and the keytab file created in Step 1. See Section 6.7, "Creating a JAAS Login File".
   b. Configure a Negotiate Identity Assertion provider in the WebLogic Server security realm. See Section 5.9.3, "Configuring a Negotiate Identity Assertion Provider".
5. Start WebLogic Server using specific start-up arguments. See Section 6.9, "Using Startup Arguments for Kerberos Authentication with WebLogic Server".

The following sections describe these steps in detail.

6.4 Configuring Your Network Domain to Use Kerberos

A Windows domain controller can serve as the Kerberos Key Distribution Center (KDC) server for Kerberos-based client and host systems. On any domain controller, the Active Directory and the Kerberos services are running automatically.

Java GSS requires a Kerberos configuration file. The default name and location of the Kerberos configuration file depends on the operating system being used. Java GSS uses the following order to search for the default configuration file:

1. The file referenced by the Java property `java.security.krb5.conf`.
2. `${java.home}/lib/security/krb5.conf`.
3. `%windir%\krb5.ini` on Microsoft Windows platforms.
4. `/etc/krb5/krb5.conf` on Solaris platforms.
5. `/etc/krb5.conf` on other UNIX platforms.

To configure Kerberos in your Windows domain controller, you need to configure each machine that will access the KDC to locate the Kerberos realm and available KDC servers. For example:

Example 6–1 Sample krb5.ini File

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

```
[realms]

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

[domain_realm]
.mydom.com = MYDOM.COM

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

6.5 Creating a Kerberos Identification for WebLogic Server

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

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

- `setspn`—Microsoft Windows Resource Kit
- `ktpass`—Microsoft Windows distribution CD in `Program Files\Support Tools`

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

The process for creating a Kerberos identification consists of the following steps:

- Step 1: Create a User Account for the Host Computer
- Step 2: Configure the User Account to Comply with Kerberos
- Step 3: Define a Service Principal Name and Create a Keytab for the Service
- Step 4: Verify Correct Setup
- Step 5: Update Default JDK Security Policy Files

6.5.1 Step 1: Create a User Account for the Host Computer

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

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

Note the password you defined when creating the user account. You will need it for the instructions described in Step 3: Define a Service Principal Name and Create a Keytab for the Service. Do not select the **User must change password at next logon** option or any other password options.

6.5.2 Step 2: Configure the User Account to Comply with Kerberos

Configure the new user account to comply with the Kerberos protocol as follows. The user account's encryption type must be DES, at a minimum, and the account must require Kerberos pre-authentication. Stronger encryption types are supported, including AES-128, AES-256, and RC4.

> **Note:** The use of a particular encryption type has a dependency on the version of the Microsoft Windows platform on which Active Directory runs. For more information, including a list of supported encryption types, see the Oracle Fusion Middleware Supported System Configurations page on the Oracle Technology Network.

1. Right-click the name of the user account in the Users tree in the left pane and select **Properties**.
2. Select the **Account** tab and check the following:
   - If you plan to use DES encryption, check the box **Use DES encryption types for this account**.
   - If you plan to use AES encryption, check the boxes **This account supports Kerberos AES 128** and **This account supports Kerberos AES 256**, and make sure that **Use Kerberos DES Encryption** is unchecked.

   Make sure no other boxes are checked, particularly the box "Do not require Kerberos pre-authentication."
3. Click **OK**.

> **Caution:** Setting the encryption type may corrupt the password. Therefore, reset the user password by right-clicking the name of the user account, selecting **Reset Password**, and re-entering the password created in Step 1: Create a User Account for the Host Computer.

6.5.3 Step 3: Define a Service Principal Name and Create a Keytab for the Service

As mentioned in Section 6.5, "Creating a Kerberos Identification for WebLogic Server", an SPN is a unique name that identifies an instance of a service and is associated with the logon account under which the service instance runs. The SPN is used in the process of mutual authentication between the client and the server hosting a particular service. The client finds a computer account based on the SPN of the service to which it is trying to connect. So, in a specific project, you need to link the service that will be

invoked by your WebLogic clients to the account you just defined for your WebLogic Server. For example, the service invoked by the WebLogic browser clients is `HTTP/myhost.example.com`, which needs to be linked to the `myhost` account.

Windows account names are not multipart as Kerberos principal names. Because of this, it is not possible to directly create an account using the name `HTTP/`*`hostname.dns.com`*. Such a principal instance is created through SPN mappings. In this case, an account is created with a meaningful name *`hostname`*, and an SPN mapping is added for `HTTP/`*`hostname.dns.com`*.

The specific steps for defining an SPN and creating a keytab for the service depend on the underlying platform on which WebLogic Server is running. They are provided in the following sections:

- Section 6.5.3.1, "Defining an SPN and Creating a Keytab on Windows Systems"
- Section 6.5.3.2, "Defining an SPN and Creating a Keytab on UNIX Systems"

6.5.3.1 Defining an SPN and Creating a Keytab on Windows Systems

If WebLogic Server runs on a Windows system, complete the following steps:

1. Use the `setspn` utility to create the SPN for the HTTP service for the WebLogic Server account created in Step 1. For example:

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

2. Identify the SPNs that are associated with your user account by entering the `setspn -L` command. For example:

   ```
   setspn -L myhost
   ```

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

3. Use the `ktab` utility to create a keytab to be exported to the WebLogic Server machine. The command to run the `ktab` utility has the following syntax (note that the Kerberos realm name must be entered in all uppercase):

   ```
   ktab -k keytab-file-name -a account-name@REALM.NAME
   ```

   For example:

   ```
   ktab -k mykeytab -a myhost@MYDOM.COM
   ```

   When prompted for a password, enter the password created in Section 6.5.1, "Step 1: Create a User Account for the Host Computer."

4. Save the keytab file in a secure location and export it to the domain directory of your WebLogic Server instance (for example, to `myhost`).

6.5.3.2 Defining an SPN and Creating a Keytab on UNIX Systems

If WebLogic Server runs on a UNIX system, create a service principal name (SPN) and a keytab file for the HTTP service for the WebLogic Server account by using the `ktpass` command-line tool. This tool enables an administrator to configure a non-Windows Server Kerberos service as a security principal in the Windows Server Active Directory.

The `ktpass` command configures the SPN for the service in Active Directory and generates a Kerberos keytab file containing the shared secret key of the service. The tool allows UNIX-based services that support Kerberos authentication to use the interoperability features provided by the Windows Server Kerberos KDC service.

The `ktpass` command has the following syntax:

```
ktpass -princ HTTP/hostname@REALM-NAME -mapuser account-name -pass password -out
keytab-file-name -crypto algorithm -ptype KRB5_NT_PRINCIPAL
```

In the preceding syntax, *`algorithm`* identifies the encryption algorithm to use. If you are using AES, specify `AES128-SHA1` or `AES256-SHA1`. For example:

```
ktpass -princ HTTP/myhost.example.com@MYDOM.COM -mapuser myhost -pass password
-out mykeytab -crypto AES256-SHA1 -ptype KRB5_NT_PRINCIPAL
```

Note: On UNIX systems, creating an SPN that uses a DES or an AES encryption algorithm is supported as of JDK 1.6.0_24 or later.

To verify that the SPN and the keytab file are set up correctly, you can do the following:

- Use the `setspn -l` command to verify that the SPN is mapped successfully.
- Use the klist command to show `Key type: 17` for AES-128, and `Key type: 18` for AES-256. For example:

  ```
  -klist -e -k keytab-file-name
  ```

- Use the kinit command to verify that the Kerberos setup and keytab are valid.

Note: The `ktpass` command changes the principal name in the Active Directory server from *`account-name`* to `HTTP/`*`account-name`*. Consequently, the keytab file is generated for a principal named `HTTP/`*`account-name`*. However, sometimes the name change does not happen. If not, you should change it manually in the Active Directory server; otherwise the keytab you generate will not work properly.

To modify the user logon name manually:

1. Right-click on the user node, select **Properties**, and click on the **Account** tab.
2. Export the generated keytab file to your UNIX machine hosting the WebLogic Server in the WebLogic domain directory.

6.5.4 Step 4: Verify Correct Setup

You can use the following utilities to verify that your SPN and keytab files are set up correctly.

- Use the `setspn -l` command to verify that the SPN is mapped successfully.
- Use the `klist` command to verify the key type. For example:

  ```
  -klist -e -k keytab-file-name
  ```

  For AES 128, this command displays `Key type: 17`. For AES 256, `Key type: 18` is displayed.

- Use the `kinit` utility to verify that Kerberos is set up properly and that your principal and keytab are valid.

  The `kinit` utility is provided by the JRE and may be used to obtain and cache Kerberos ticket-granting tickets. You can run the `kinit` utility by entering the following command:

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

  The output should appear similar to the following:

  ```
  New ticket is stored in cache file C:\Documents and Settings\Username\krb5cc_
  myhost
  ```

6.5.5 Step 5: Update Default JDK Security Policy Files

AES-256 requires the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files, which are available at the following URL:

`http://www.oracle.com/technetwork/java/javase/downloads/index.html`

If you plan to use AES encryption, complete the following steps:

1. Download JCE Unlimited Strength Jurisdiction Policy Files from the preceding URL.
2. Uncompress and extract the jurisdiction policy files.
3. Complete the installation procedure described in the included `README.txt` file.

6.6 Configuring Microsoft Clients to Use Windows Integrated Authentication

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

- Section 6.6.1, "Configuring a .NET Web Service"
- Section 6.6.2, "Configuring an Internet Explorer Browser"
- Section 6.6.3, "Configuring a Mozilla Firefox Browser"
- Section 6.6.4, "Configuring a Java SE Client Application"

Note: If the SPN for the user account on the WebLogic Server host to which the Kerberos ticket is mapped is configured to use DES or AES-256 encryption (see Section 6.5.2, "Step 2: Configure the User Account to Comply with Kerberos"), the client must be running with a supported JDK. For information, see the Oracle Fusion Middleware Supported System Configurations page on the Oracle Technology Network.

6.6.1 Configuring a .NET Web Service

To configure a .NET Web service to use Windows Integrated Authentication:

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

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

This setting is usually the default.

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

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

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

6.6.2 Configuring an Internet Explorer Browser

To configure an Internet Explorer browser to use Windows Integrated Authentication, complete the procedures described in the following sections:

- Section 6.6.2.1, "Configure Local Intranet Domains"
- Section 6.6.2.2, "Configure Intranet Authentication"
- Section 6.6.2.3, "Verify the Proxy Settings"
- Section 6.6.2.4, "Set Integrated Authentication for Internet Explorer 6.0"

6.6.2.1 Configure Local Intranet Domains

In Internet Explorer:

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

6.6.2.2 Configure Intranet Authentication

In Internet Explorer:

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

6.6.2.3 Verify the Proxy Settings

If you have a proxy server enabled:

1. In Internet Explorer, select **Tools** > **Internet Options**.
2. Select the Connections tab and click **LAN Settings**.
3. Verify that the proxy server address and port number are correct.
4. Click **Advanced**.
5. In the Proxy Settings dialog box, ensure that all desired domain names are entered in the Exceptions field.
6. Click **OK** to close the Proxy Settings dialog box.

6.6.2.4 Set Integrated Authentication for Internet Explorer 6.0

If the version of Internet Explorer you are configuring is 6.0, you also must complete the following steps:

1. In Internet Explorer, select **Tools** > **Internet Options**.
2. Select the Advanced tab.
3. Scroll to the Security section.
4. Verify that the Enable Integrated Windows Integrated Authentication option is checked and click **OK**.

   If this option was not checked, check it, click **OK**, and restart the computer.

6.6.3 Configuring a Mozilla Firefox Browser

To configure a Firefox browser to use Windows Integrated Authentication, complete the following steps:

1. Start Firefox.
2. In the Location Bar, enter `about:config`.
3. Enter the filter string `network.negotiate`.
4. Set the preferences as shown in Table 6–2.

Table 6–2 Preferences Required in Firefox for Windows Integrated Authentication

Preference Name	Status	Type	Value
network.negotiate-auth.allow-proxies	default	boolean	true
network.negotiate-auth.delegation-uris	user set	string	http://,https://
network.negotiate-auth.gsslib	default	string	<blank>[1]
network.negotiate-auth.trusted-uris	user set	string	http://,https://
network.negotiate-auth.using-native-gsslib	default	boolean	true

[1] The value for the `network.negotiate-auth.gsslib` preference should be kept blank.

6.6.4 Configuring a Java SE Client Application

To configure a Java SE client application to use Windows Integrated Authentication, complete the following steps:

1. Ensure that your Java SE client is running with a supported JDK. For information, see the Oracle Fusion Middleware Supported System Configurations page on the Oracle Technology Network.
2. Create a JAAS configuration file that identifies the Kerberos login module, `com.sun.security.auth.module.Krb5LoginModule`. This configuration file defines two login entries:
   - `com.sun.security.jgss.krb5.initiate` — Invoked for the Java client.
   - `com.sun.security.jgss.krb5.accept` — Invoked for the WebLogic Server instance that is represented by a Kerberos identity and that hosts the Web application to which the client wants access.

   For each login entry, options are included that:
   - Require that authentication of the principal must succeed (that is, the user of the client application who is defined in the Microsoft domain).
   - Set `useKeyTab` to `true`, which causes the principal's key to be obtained from the keytab.
   - Identify the name of the keytab.
   - Set `storeKey` to `true`, which causes the principal's key to be stored in the Subject.
   - Optionally, set the `debug` option to `true`.

   The following example shows JAAS configuration file for the Kerberos login module used for the principal `negotiatetester`, who is defined in the Microsoft domain, `SECURITYQA.COM`, in which the Active Directory server runs:

   ```
   com.sun.security.jgss.krb5.initiate {
      com.sun.security.auth.module.Krb5LoginModule
      required principal="negotiatetester@SECURITYQA.COM"
      useKeyTab=true
      keyTab=negotiatetester_keytab storeKey=true debug=true; };

   com.sun.security.jgss.krb5.accept {
      com.sun.security.auth.module.Krb5LoginModule
      required principal="negotiatetester@SECURITYQA.COM"
      useKeyTab=true keyTab=negotiatetester_keytab storeKey=true debug=true; };
   ```

3. In the Java command that starts the client application, pass the following values as arguments:
   - The Microsoft domain in which the Active Directory server runs
   - The host name of the computer running the Kerberos Key Distribution Center (KDC) server
   - The JAAS configuration file that identifies the Kerberos login module
   - The `javax.security.auth.useSubjectCredsOnly=false` property, which specifies that it is permissible to use an authentication mechanism other than Subject credentials
   - The name of the Java SE client class
   - The Web application resource to which access is requested

   For example:

   ```
   java -Djava.security.krb5.realm = SECURITYQA.COM\
   -Djava.security.krb5.kdc = rno05089.example.com\
   ```

```
-Djava.security.auth.login.config = negotiatetester_krb5Login.conf\
-Djavax.security.auth.useSubjectCredsOnly = false\
RunHttpSpnego http://myhost.example.com:7001/AuthenticatedServlet.jsp
```

6.7 Creating a JAAS Login File

If you are running WebLogic Server on either the Windows or UNIX platforms, you need a JAAS login file. The JAAS login file tells the WebLogic Security Framework to use Kerberos authentication and defines the location of the keytab file which contains Kerberos identification information for WebLogic Server. You specify the location of the JAAS login file in the `java.security.auth.login.config` startup argument for WebLogic Server, as described in Section 6.9, "Using Startup Arguments for Kerberos Authentication with WebLogic Server".

Notes: The JAAS Login Entry names are `com.sun.security.jgss.krb5.initiate` and `com.sun.security.jgss.krb5.accept`.

Example 6–2 shows a sample JAAS login file for Kerberos authentication. Significant sections are shown in **bold**.

Example 6–2 Sample JAAS Login File for Kerberos Authentication

```
com.sun.security.jgss.krb5.initiate {

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

com.sun.security.jgss.krb5.accept {

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

};
```

For the principal option, specify the value of the `userPrincipalName` attribute of the account under which the service is running. (Incorrectly specifying the user principal name results in an error such as "Unable to obtain password from user.")

The keytab file specified in the `keytab` option must be accessible by the WebLogic Server process. Ensure that the appropriate permissions are set. If you are unsure of the search path WebLogic Server is using, provide the absolute path to the file. Make sure you enclose the path in double quotes, and replace any backslash (`\`) in the path with a double backslash (`\\`) or a forward slash (`/`).

6.8 Configuring the Identity Assertion Provider

WebLogic Server includes a security provider, the Negotiate Identity Assertion provider, to support single sign-on (SSO) with Microsoft clients. This identity assertion provider decodes Simple and Protected Negotiate (SPNEGO) tokens to obtain Kerberos tokens, validates the Kerberos tokens, and maps Kerberos tokens to WebLogic users. You need to configure a Negotiate Identity Assertion provider in your WebLogic security realm in order to enable SSO with Microsoft clients. See

Section 5.9.3, "Configuring a Negotiate Identity Assertion Provider" in this document, and see also "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*.

6.9 Using Startup Arguments for Kerberos Authentication with WebLogic Server

To use Kerberos authentication with WebLogic Server, use the following arguments in the Java command to start WebLogic Server:

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

In the preceding list of arguments:

- `javax.security.auth.useSubjectCredsOnly` specifies that it is permissible to use an authentication mechanism other than Subject credentials.
- `java.security.auth.login.config` specifies the JAAS login file, `krb5Login.conf`, described in Section 6.7, "Creating a JAAS Login File".
- `java.security.krb5.realm` defines the Microsoft domain in which the Active Directory server runs.
- `java.security.krb5.kdc` defines the host name on which the Active Directory server runs.

Java GSS messages are often very useful during troubleshooting, so you might want to add `-Dsun.security.krb5.debug=true` as part of the initial setup.

6.10 Verifying Configuration of SSO with Microsoft Clients

To verify that SSO with Microsoft clients is configured properly, point a browser (that you have configured as described in Section 6.6.2, "Configuring an Internet Explorer Browser") to the Microsoft Web application or Web service you want to use. If you are logged on to a Windows domain and have Kerberos credentials acquired from the Active Directory server in the domain, you should be able to access the Web application or Web service without providing a username or password.

7

Configuring Single Sign-On with Web Browsers and HTTP Clients

This chapter describes how to set up single sign-on (SSO) with Web browsers or other HTTP clients by using authentication based on the Security Assertion Markup Language (SAML) versions 1.1 and 2.0. SAML enables cross-platform authentication between Web applications or Web services running in a WebLogic domain and Web browsers or other HTTP clients. WebLogic Server supports single sign-on (SSO) based on SAML. When users are authenticated at one site that participates in a single sign-on (SSO) configuration, they are automatically authenticated at other sites in the SSO configuration and do not need to log in separately.

This chapter includes the following sections:

- Configuring Single Sign-On Using SAML White Paper
- SAML for Web Single Sign-On Scenario API Example
- Configuring SAML 1.1 Services
- Configuring SAML 2.0 Services
- Enabling Debugging for SAML 1.1 and 2.0

Notes: Note the following:

- A WebLogic Server instance that is configured for SAML 2.0 SSO cannot sent a request to a server instance configured for SAML 1.1, and vice-versa.
- WebLogic Server does not support encrypted SAML assertions.

For an overview of SAML-based single sign on, see the following topics in *Understanding Security for Oracle WebLogic Server*:

- "Security Assertion Markup Language (SAML)"
- "Web Browsers and HTTP Clients via SAML"
- "Single Sign-On with the WebLogic Security Framework"

7.1 Configuring Single Sign-On Using SAML White Paper

The *Configuring Single Sign-On using SAML in WebLogic Server 9.2* white paper (`http://www.oracle.com/technetwork/articles/idm/sso-with-saml-099684.html`) provides step-by-step instructions for configuring the single sign-on capability between two simple Java EE Web applications running on two different WebLogic

domains. The SAML configuration for single sign-on is performed using the WebLogic Server 9.2 Administration Console with no programming involved. The tutorial also briefly introduces the basic interactions between WebLogic containers, the security providers, and the security framework during the single sign-on process.

Although it is based on a previous version of WebLogic Server, you may find this tutorial to be a useful resource as you develop your own SAML implementation.

7.2 SAML for Web Single Sign-On Scenario API Example

When you install the Server Examples component of WebLogic Server, which is available by performing a custom installation, WebLogic Server installs several API code examples. The Server Examples provide access to code examples and sample applications that offer several approaches to learning about and working with WebLogic Server.

Included among the security API examples is SAML for Web SSO Scenario. This example, which you build, run, and deploy, shows a variety of single sign-on (SSO) configurations for your applications using WebLogic Server and SAML. The following three scenarios are included:

- SAML 2.0 POST binding
- SAML 1.1
- SAML 2.0 Artifact binding with custom attributes

All files needed to build, deploy, and run the example are included, as are the scripts that configure the WebLogic domains that are used. For more information about the examples, including the directories in which they are installed, see "Sample Application and Code Examples" in *Understanding Oracle WebLogic Server*.

7.3 Configuring SAML 1.1 Services

This topic includes the following sections:

- Section 7.3.1, "Enabling Single Sign-on with SAML 1.1: Main Steps"
- Section 7.3.2, "Configuring a SAML 1.1 Source Site for Single Sign-On"
- Section 7.3.3, "Configuring a SAML 1.1 Destination Site for Single Sign-On"
- Section 7.3.4, "Configuring Relying and Asserting Parties with WLST"

In addition to the topics described in these sections, see "Creating Assertions for Non-WebLogic SAML 1.1 Relying Parties" in *Developing Applications with the WebLogic Security Service* for information on how to create a custom SAML name mapper that maps Subjects to specific SAML 1.1 assertion attributes required by a third-party SAML Relying Party.

7.3.1 Enabling Single Sign-on with SAML 1.1: Main Steps

To enable single sign-on with SAML, configure WebLogic Server as either a source site or destination site as described in the sections that follow.

7.3.1.1 Configuring a Source Site: Main Steps

To configure a WebLogic Server instance in the role of a source site, complete the following main steps:

1. Create and configure a SAML Credential Mapping provider V2 in your security realm.
2. Configure the federation services for the server instance in the realm that will serve as a source site.
3. Create and configure the relying parties for which SAML assertions will be produced.
4. If you want to require relying parties to use SSL certificates to connect to the source site, add any such certificates to the SAML credential mapping provider's Certificate Registry.

7.3.1.2 Configuring a Destination Site: Main Steps

To configure a WebLogic Server instance in the role of a destination site, complete the following main steps:

1. Create and configure a SAML Identity Assertion provider V2 in your security realm.
2. Configure the federation services for the server instance realm that will serve as a destination site.
3. Create and configure the asserting parties from which SAML assertions will be consumed.
4. Establish trust by registering the asserting parties' SSL certificates in the certificate registry maintained by the SAML Identity Assertion provider.

7.3.2 Configuring a SAML 1.1 Source Site for Single Sign-On

The following topics explain how to configure a WebLogic Server instance as a SAML 1.1 source site:

- Section 7.3.2.1, "Configure the SAML 1.1 Credential Mapping Provider"
- Section 7.3.2.2, "Configure the Source Site Federation Services"
- Section 7.3.2.3, "Configure Relying Parties"
- Section 7.3.2.4, "Replacing the Default Assertion Store"

7.3.2.1 Configure the SAML 1.1 Credential Mapping Provider

In your security realm, create a SAML Credential Mapping Provider V2 instance. The SAML Credential Mapping provider is not part of the default security realm. See Section 4.10, "Configuring a SAML Credential Mapping Provider for SAML 1.1".

Configure the SAML Credential Mapping provider as a SAML authority, using the Issuer URI, Name Qualifier, and other attributes.

7.3.2.2 Configure the Source Site Federation Services

Configuration of a WebLogic Server instance as a SAML 1.1 source site is controlled by the `FederationServicesMBean`. Access the `FederationServicesMBean` with the WebLogic Scripting Tool or through the Administration Console, on the **Environment** > **Servers** > *ServerName* > **Configuration** > **Federation Services** > SAML 1.1 Source Site page. See "Configure SAML 1.1 source services" in the *Oracle WebLogic Server Administration Console Online Help*.

Configure SAML source site attributes as follows:

- **Enable the SAML Source Site**. Allow the WebLogic server instance to serve as a SAML source site by setting Source Site Enabled to true.
- **Set Source Site URL and Service URIs**. Set the URL for the SAML source site. This is the URL that hosts the Intersite Transfer Service and the Assertion Retrieval Service. The source site URL is encoded as a source ID in hex and Base64. When you configure a SAML Asserting Party for Browser/Artifact profile, you specify the encoded source ID.

  Specify the URIs for the Intersite Transfer Service and (to support Browser/Artifact profile) the Assertion Retrieval Service. (You also specify the Intersite Transfer Service URI when you configure a Relying Party.)

  The default URI `FederationServicesMBean.IntersiteTransferURIs` values are shown in Table 7–1.

Table 7–1 Intersite Transfer URIs

Default URI Values	Description
/samlits_ba/its	BASIC authentication, POST or Artifact profile
/samlits_ba/its/post	BASIC authentication, POST profile
/samlits_ba/its/artifact	BASIC authentication, Artifact profile
/samlits_cc/its	Client cert authentication, POST or Artifact profile
/samlits_cc/its/post	Client cert authentication, POST profile
/samlits_cc/its/artifact	Client cert authentication, Artifact profile

  The Intersite Transfer URI text box allows you to accept the default values as-is, or modify them as you choose. Each URI includes the application context, followed by `/its`, `/its/post`, or `/its/artifact`. The provided application contexts are `/samlits_ba` (BASIC authentication) or `/samlits_cc` (client certificate authentication). You could also specify an application-specific context if needed, for example `/yourapplication/its`, but in most cases the defaults provide the easiest configuration option.

  If you specify these URIs as `/samlits_ba/its`, if a redirect occurs and the user's session on the source site has timed out, a BASIC authentication dialog is presented. If you instead want to use a FORM dialog, the URI should point to a custom Web application that authenticates users and then forwards to the actual ITS URI.
- **Add signing certificate**. The SAML source site requires a trusted certificate with which to sign assertions. Add this certificate to the keystore and enter the credentials (alias and passphrase) to be used to access the certificate. The server's SSL identity key/certificates will be used by default if a signing alias and passphrase are not supplied.
- **Configure SSL for the Assertion Retrieval Service**. You can require all access to the Assertion Retrieval Service to use SSL by setting `FederationServicesMBean.arsRequiresSSL` to `true`. You can require two-way SSL authentication for the Assertion Retrieval Service by setting both `arsRequiresSSL` and `ARSRequiresTwoWaySSL` to `true`.

7.3.2.3 Configure Relying Parties

A SAML Relying Party is an entity that relies on the information in a SAML assertion produced by the SAML source site. You can configure how WebLogic Server produces

SAML assertions separately for each Relying Party or use the defaults established by the Federation Services source site configuration for producing assertion.

You configure a Relying Party in the Administration Console, on the **Security Realms** > *RealmName* > **Providers** > **Credential Mapper** > *SAMLCredentialMapperName* > **Management** > **Relying Parties** page. See "Create a SAML 1.1 Relying Party" and "Configure a SAML 1.1 Relying Party" in the *Oracle WebLogic Server Administration Console Online Help*.

You can also configure a Relying Party with the WebLogic Scripting Tool. See Section 7.3.4, "Configuring Relying and Asserting Parties with WLST".

7.3.2.3.1 Configure Supported Profiles When you configure a SAML Relying Party, you can specify support for Artifact profile or POST profile, for the purposes of SAML SSO. As an alternative configure a Relying Party to support WSS/Holder-of-Key or WSS/Sender-Vouches profiles for Web Services Security purposes. Be sure to configure support for the profiles that the SAML destination sites support.

If you support the POST profile, optionally create a form to use in POST profile assertions for the Relying Party and set the pathname of that form in the POST Form attribute.

7.3.2.3.2 Assertion Consumer Parameters For each SAML Relying Party, you can configure one or more optional query parameters that will be added to the ACS URL when redirecting to the destination site. In the case of POST profile, these parameters will be included as form variables when using the default POST form. If a custom POST form is in use, the parameters will be made available as a Map of names and values, but the form may or may not constructed to include the parameters in the POSTed data.

For WebLogic Server browser SSO configurations that communicate with another WebLogic Server instance, set the ID of the SAML Asserting Party (APID) in the relying party ACS parameters.

This parameter is required with the V2 providers in order for the browser profile configurations to work. That is, the ACS looks for an asserting party ID (APID) as a form parameter of the incoming request, and uses this to look up the configuration before performing any other processing.

The APID parameter also removes the need for you to specify a Target URL parameter for browser SSO. The Target URL is used for Web service configurations.

7.3.2.4 Replacing the Default Assertion Store

WebLogic Server uses a simple assertion store to maintain persistence for produced assertions. You can replace this assertion store with a custom assertion store class that implements `weblogic.security.providers.saml.AssertionStoreV2`. Configure WebLogic Server to use your custom assertion store class, rather than the default class, using the `FederationServicesMBean.AssertionStoreClassName` attribute. You can configure properties to be passed to the `initStore()` method of your custom assertion store class by using the `FederationServicesMBean.AssertionStoreProperties` attribute. Configure these attributes in the Administration Console on the **Environment: Servers** > *ServerName* > **Configuration** > **Federation Services** > **SAML 1.1 Source Site** page.

7.3.3 Configuring a SAML 1.1 Destination Site for Single Sign-On

The following topics describe how to configure WebLogic Server as a SAML destination site:

- Section 7.3.3.1, "Configure SAML Identity Assertion Provider"
- Section 7.3.3.2, "Configure Destination Site Federation Services"
- Section 7.3.3.3, "Configuring Asserting Parties"

7.3.3.1 Configure SAML Identity Assertion Provider

In your security realm, create and configure a SAML Identity Assertion Provider V2 instance. The SAML Identity Assertion provider is not part of the default security realm. See Section 5.9.4, "Configuring a SAML Identity Assertion Provider for SAML 1.1".

7.3.3.2 Configure Destination Site Federation Services

Before you configure WebLogic as a SAML destination site, you must first create a SAML Identity Assertion Provider V2 instance in your security realm. Configuration of a WebLogic Server instance as a SAML destination site is controlled by the `FederationServicesMBean`. You can access the `FederationServicesMBean` using the WebLogic Scripting Tool or through the Administration Console, using the **Environment: Servers** > *ServerName* > **Configuration** > **Federation Services** > **SAML 1.1 Destination Site** page.

Configure the SAML destination site attributes as follows.

7.3.3.2.1 Enable the SAML Destination Site Allow the WebLogic Server instance to serve as a SAML destination site by setting Destination Site Enabled to `true`.

7.3.3.2.2 Set Assertion Consumer URIs Set the URIs for the SAML Assertion Consumer Service. This is the URL that receives assertions from source sites, so that the destination site can use the assertions to authenticate users. The Assertion Consumer URI is also specified in the configuration of a Relying Party.

7.3.3.2.3 Configure SSL for the Assertion Consumer Service You can require all access to the Assertion Consumer Service to use SSL by setting `FederationServicesMBean.acsRequiresSSL` to `true`.

7.3.3.2.4 Add SSL Client Identity Certificate The SSL client identity is used to contact the ARS at the source site for Artifact profile. Add this certificate to the keystore and enter the credentials (alias and passphrase) to be used to access the certificate.

7.3.3.2.5 Configure Single-Use Policy and the Used Assertion Cache or Custom Assertion Cache Optionally, you can require that each POST profile assertion be used no more than once. WebLogic Server maintains a cache of used assertions so that it can support a single-use policy for assertions. You can replace this assertion cache with a custom assertion cache class that implements `weblogic.security.providers.saml.SAMLUsedAssertionCache`. Configure WebLogic Server to use your custom assertion cache class, rather than the default class, using the `FederationServicesMBean.SAMLUsedAssertionCache` attribute. You can configure properties to be passed to the `initCache()` method of your custom assertion cache class using the `FederationServicesMBean.UsedAssertionCacheProperties` attribute. You can configure these attributes in the Administration Console on the **Environment** > **Servers** > *ServerName* > **Configuration** > **Federation Services** > **SAML 1.1 Destination Site** page.

7.3.3.2.6 Configure Recipient Check for POST Profile Optionally, you can require that the recipient of the SAML Response must match the URL in the HTTP Request. Do this by setting the POST Recipient Check Enabled attribute.

7.3.3.3 Configuring Asserting Parties

A SAML Asserting Party is a trusted SAML Authority (an entity that can authoritatively assert security information in the form of SAML Assertions).

Configure an Asserting Party in the Administration Console, using the **Security Realms** > *RealmName* > **Providers** > **Authentication** > **SAMLIdentityAsserterV2** > **Management: Asserting Parties** page. See "Create a SAML 1.1 Asserting Party" and "Configure a SAML 1.1 Asserting Party" in the *Oracle WebLogic Server Administration Console Online Help*.

You can also configure an Asserting Party with the WebLogic Scripting Tool. See Section 7.3.4, "Configuring Relying and Asserting Parties with WLST".

7.3.3.3.1 Configure Supported Profiles When you configure a SAML Asserting Party, you can specify support for Artifact profile or POST profile, for the purposes of SAML SSO. Alternatively, configure an Asserting Party to support WSS/Holder-of-Key or WSS/Sender-Vouches profiles for Web Services Security purposes.

7.3.3.3.2 Configure Source Site ITS Parameters For each SAML Asserting Party, configure zero or more optional query parameters that will be added when redirecting to the source site.

For WebLogic Server browser SSO configurations that communicate with another WebLogic Server instance, you must set the ID of the SAML Relying Party (RPID) in the Asserting Party ITS parameters.

This parameter is required with the V2 providers in order for the browser profile configurations to work. That is, the ITS looks for the RPID as a form parameter of the incoming request, and uses this to look up the configuration before performing any other processing.

The RPID parameter also removes the need for you to specify a Target URL parameter for WebLogic Server-to-WebLogic Server browser SSO configurations only. The Target URL is used for Web service configurations.

7.3.4 Configuring Relying and Asserting Parties with WLST

SAML partners (Relying Parties and Asserting Parties) are maintained in a registry. You can configure SAML partners using the WebLogic Administration Console or using WebLogic Scripting Tool. The following example shows how you might configure two Relying Parties using WLST in online mode.

Note that the example sets the ID of the SAML Asserting Party (APID) in the relying party Assertion Consumer Service parameters. For WebLogic Server browser SSO configurations that communicate with another WebLogic Server instance, you must set the ID of the SAML Asserting Party (APID) in the relying party ACS parameters. (You would also set the ID of the SAML Relying Party (RPID) in the asserting party ITS parameters.)

The `demoidentity` certificate alias referenced in the example comes from the source site's demo SSL identity for the domain.

The APID is required for WebLogic Server-to-WebLogic Server browser SSO configurations only. This parameter is required with the V2 providers in order for the browser profile configurations to work.

Example 7–1 Creating Relying Parties with WLST

```
connect('weblogic','weblogic','t3://localhost:7001')
rlm=cmo.getSecurityConfiguration().getDefaultRealm()
```

```
cm=rlm.lookupCredentialMapper('samlv2cm')

rp=cm.newRelyingParty()
rp.setDescription('test post profile')
rp.setProfile('Browser/POST')
rp.setAssertionConsumerURL('http://domain.example.com:7001/saml_destination/acs')
rp.setAssertionConsumerParams(array(['APID=ap_00001'],String))
rp.setSignedAssertions(true)
rp.setEnabled(true)
cm.addRelyingParty(rp)

rp=cm.newRelyingParty()
rp.setDescription('test artifact profile')
rp.setProfile('Browser/Artifact')
rp.setAssertionConsumerURL('http://domain.example.com:7001/saml_destination/acs')
rp.setAssertionConsumerParams(array(['APID=ap_00002'],String))
rp.setARSUsername('foo')
rp.setARSPassword('password')
rp.setSSLClientCertAlias('demoidentity')
rp.setEnabled(true)
cm.addRelyingParty(rp)

disconnect()
exit()
```

The following example shows how you might edit an existing Asserting Party. The example gets the Asserting Party, using its Asserting Party ID, and sets the Assertion Retrieval URL.

Example 7–2 Editing an Asserting Party with WLST

```
connect('weblogic','weblogic','t3://localhost:7001')
rlm=cmo.getSecurityConfiguration().getDefaultRealm()
ia=rlm.lookupAuthenticationProvider('samlv2ia')
ap=ia.getAssertingParty('ap_00002')
ap.setAssertionRetrievalURL('https://hostname:7002/samlars/ars')
ia.updateAssertingParty(ap)
disconnect()
exit()
```

7.4 Configuring SAML 2.0 Services

This topic includes the following sections:

- Section 7.4.1, "Configuring SAML 2.0 Services: Main Steps"
- Section 7.4.2, "Configuring SAML 2.0 General Services"
- Section 7.4.3, "Configuring an Identity Provider Site for SAML 2.0 Single Sign-On"
- Section 7.4.4, "Configuring a Service Provider Site for SAML 2.0 Single Sign-On"
- Section 7.4.5, "Viewing Partner Site, Certificate, and Service Endpoint Information"
- Section 7.4.6, "Web Application Deployment Considerations for SAML 2.0"

7.4.1 Configuring SAML 2.0 Services: Main Steps

A summary of the main steps you take to configure SAML 2.0 services is as follows:

1. Determine whether you plan to have SAML 2.0 services running in more than one WebLogic Server instance in the domain. If so, do the following:

a. Create a domain in which the RDBMS security store is configured.

   The RDBMS security store is required by the SAML 2.0 security providers in production environments so that the data they manage can be synchronized across all the WebLogic Server instances that share that data.

   Note that Oracle does not recommend upgrading an existing domain in place to use the RDBMS security store. If you want to use the RDBMS security store, you should configure the RDBMS security store at the time of domain creation. If you have an existing domain with which you want to use the RDBMS security store, create the new domain and migrate your existing security realm to it.

   For information, see Chapter 9, "Managing the RDBMS Security Store".

b. Ensure that all SAML 2.0 services are configured identically in each WebLogic Server instance. If you are configuring SAML 2.0 services in a cluster, each Managed Server in that cluster must be configured individually.

c. Note the considerations described in Section 7.4.6, "Web Application Deployment Considerations for SAML 2.0".

2. If you are configuring a SAML 2.0 Identity Provider site:

   a. Create and configure an instance of the SAML 2.0 Credential Mapping provider in the security realm.

   b. Configure the SAML 2.0 general services identically and individually in each WebLogic Server instance in the domain that will run SAML 2.0 services.

   c. Configure the SAML 2.0 Identity Provider services identically and individually in each WebLogic Server instance in the domain that will run SAML 2.0 services.

   d. Publish the metadata file describing your site, and manually distribute it to your Service Provider partners.

   e. Create and configure your Service Provider partners.

3. If you are configuring a SAML 2.0 Service Provider site:

   a. Create and configure an instance of the SAML 2.0 Identity Assertion provider in the security realm.

      If you are allowing virtual users to log in via SAML, you need to create and configure an instance of the SAML Authentication provider. For information, see Section 5.7, "Configuring the SAML Authentication Provider".

   b. Configure the SAML 2.0 general services identically and individually in each WebLogic Server instance in the domain that will run SAML 2.0 services.

   c. Configure the SAML 2.0 Service Provider services identically and individually in each WebLogic Server instance in the domain that will run SAML 2.0 services.

   d. Publish the metadata file describing your site, and manually distribute it to your Identity Provider partners.

   e. Create and configure your Identity Provider partners.

The sections that follow provide details about each set of main steps.

7.4.2 Configuring SAML 2.0 General Services

Regardless of the SAML 2.0 role in which you wish to configure a WebLogic Server instance — that is, as either a Service Provider or Identity Provider — you need to configure the server's general SAML 2.0 services. Configuration of the SAML 2.0 general services for a WebLogic Server instance is controlled by the `SingleSignOnServicesMBean`. You can access the `SingleSignOnServicesMBean` with the WebLogic Scripting Tool or through the Administration Console, on the **Environment** > **Servers** > *ServerName* > **Configuration** > **Federation Services** > **SAML 2.0 General** page.

> **Note:** You cannot configure SAML 2.0 general services in a WebLogic Server instance until you have first configured either the SAML 2.0 Identity Assertion or SAML 2.0 Credential Mapping provider and restarted the server instance.

The following sections describe SAML 2.0 general services:

- Section 7.4.2.1, "About SAML 2.0 General Services"
- Section 7.4.2.2, "Publishing and Distributing the Metadata File"

7.4.2.1 About SAML 2.0 General Services

The general SAML 2.0 services you configure include the following:

- Whether you wish to enable the replicated cache

  Enabling the replicated cache is required if you are configuring SAML 2.0 services on two or more WebLogic Server instances in a domain, such as in a cluster. The replicated cache enables server instances to share and be synchronized with the data that is managed by the SAML 2.0 security providers; that is, either or both the SAML 2.0 Identity Assertion provider and the SAML 2.0 Credential Mapping provider.

  The RDBMS security store is required by the SAML 2.0 security providers in production environments so that the data they manage can be synchronized across all the WebLogic Server instances that share that data. (Use LDAP as the security store with the SAML 2.0 security providers only in development environments.)

  Therefore, prior to configuring SAML 2.0 services, the preferred approach is first to create a domain that is configured to use the RDBMS security store. For more information, see Chapter 9, "Managing the RDBMS Security Store".

- Information about the local site

  The site information you enter is primarily for the benefit of the business partners in the SAML federation with whom you share it. Site information includes details about the local contact person who is your partners' point of contact, your organization name, and your organization's URL.

- Published site URL

  This URL specifies the base URL that is used to construct endpoint URLs for the various SAML 2.0 services. The published site URL should specify the host name and port at which the server is visible externally, which might not be the same at which the server is accessed locally. For example, if SAML 2.0 services are configured in a cluster, the host name and port may correspond to the load balancer or proxy server that distributes client requests to the Managed Servers in that cluster.

The published site URL should be appended with /saml2. For example:

```
https://www.avitek.com:7001/avitek-domain/aviserver/saml2
```

- Entity ID

  The entity ID is a human-readable string that uniquely distinguishes your site from the other partner sites in your federation. When your partners need to generate or consume an assertion, the SAML 2.0 services use the entity ID as part of the process of identifying the partner that corresponds with that assertion.

- Whether recipient check is enabled

  If enabled, the recipient of the authentication request or response must match the URL in the HTTP Request.

- Whether TLS/SSL client authentication is required for invocations on the Artifact Resolution Service. If enabled, SAML artifacts are encrypted when transmitted to partners.

- Transport Layer Security keystore alias and passphrase, the values used for securing outgoing communications with partners.

- Whether Basic authentication client authentication is required when your partners invoke the HTTPS bindings of the local site.

  If you enable this setting, you also specify the client username and password to be used. These credentials are then included in the published metadata file that you share with your federated partners.

- Whether requests for SAML artifacts received from your partners must be signed.

- Configuration settings for the SAML artifact cache.

- Keystore alias and passphrase for the key to be used when signing documents sent to your federated partners, such as authentication requests or responses.

For information about the steps for configuring SAML 2.0 general services, see "Configure SAML 2.0 general services" in the *Oracle WebLogic Server Administration Console Online Help*.

7.4.2.2 Publishing and Distributing the Metadata File

The local site information that is needed by your federated partners — such as the local site contact information, entity ID, published site URL, whether TLS/SSL client authentication is required, and so on — is published to a metadata file by clicking **Publish Meta Data** in the SAML 2.0 General console page.

When you publish the metadata file, you specify an existing directory on the local machine in which the file can be created. The process of distributing the metadata file to your federated partners is a detail that is not implemented by WebLogic Server. However, you may send this file via a number of commonly used mechanisms suitable for securely transferring electronic documents, such as encrypted email or secure FTP.

Keep the following in mind regarding the metadata file:

- Before you publish the metadata file, you should configure the Identity Provider and/or Service Provider services for the SAML 2.0 roles in which the WebLogic Server instances in your domain are enabled to function.

  The configuration data for the SAML 2.0 services your site offers that is needed by your federated partners is included in this metadata file, greatly simplifying the tasks your partners perform to import your signing certificates, identify your site's SAML 2.0 service endpoints, and use the correct binding types for connecting to your site's services, and so on.

- You should have only a single version of the metadata file that you share with your federated partners, even if your site functions in the role of Service Provider with some partners and Identity Provider with others. By having only a single version of the metadata file, you reduce the likelihood that your configuration settings might become incompatible with those of a partner.
- If you change the local site's SAML 2.0 configuration, you should update your metadata file. Because the metadata file is shared with your partners, it will be convenient to minimize the frequency with which you update your SAML 2.0 configuration so that your partners can minimize the need to make concomitant updates to their own partner registries.
- When you receive a metadata file from a federated partner, place it in a location that can be accessed by all the nodes in your domain in which SAML 2.0 services are configured. At the time you create a partner, you bring the contents the partner's metadata file into the partner registry.

Operations on the metadata file are available via the `com.bea.security.saml2.providers.registry.Partner` Java interface.

7.4.3 Configuring an Identity Provider Site for SAML 2.0 Single Sign-On

This section presents the following topics:

- Section 7.4.3.1, "Configure the SAML 2.0 Credential Mapping Provider"
- Section 7.4.3.2, "Configure SAML 2.0 Identity Provider Services"
- Section 7.4.3.3, "Create and Configure Web Single Sign-On Service Provider Partners"

7.4.3.1 Configure the SAML 2.0 Credential Mapping Provider

In your security realm, create a SAML 2.0 Credential Mapping provider instance. The SAML 2.0 Credential Mapping provider is not part of the default security realm. See Section 4.11, "Configuring a SAML 2.0 Credential Mapping Provider for SAML 2.0".

Configure the SAML 2.0 Credential Mapping provider as a SAML authority. Attributes you specify include the following:

- Issuer URI
- Name Qualifier
- Life span attributes for generated SAML 2.0 assertions
- Name mapper class name
- Whether generated assertions should include attribute information, which specify the groups to which the identity contained in the assertion belongs

After you configure the SAML 2.0 Credential Mapping provider, configure SAML 2.0 general services, as described in Section 7.4.2, "Configuring SAML 2.0 General Services".

7.4.3.2 Configure SAML 2.0 Identity Provider Services

Configuration of a WebLogic Server instance as a SAML 2.0 Identity Provider site is controlled by the `SingleSignOnServicesMBean`. You can access the `SingleSignOnServicesMBean` using the WebLogic Scripting Tool (WLST), or through the Administration Console by using the **Environment** > **Servers** > *ServerName* > **Configuration** > **Federation Services** > **SAML 2.0 Identity Provider** page.

The sections that follow summarize the configuration tasks. For more information about performing these tasks, see "Configure SAML 2.0 Identity Provider services" in the *Oracle WebLogic Server Administration Console Online Help*.

7.4.3.2.1 Enable the SAML 2.0 Identity Provider Site From the SAML 2.0 Identity Provider page in the console, allow the WebLogic Server instance to serve as an Identity Provider site by setting the Enabled attribute to `true`.

7.4.3.2.2 Specify a Custom Login Web Application Optionally, you may use a custom login web application to authenticate users into the Identity Provider site. To configure a custom login web application, enable the Login Customized attribute and specify the URL of the web application.

7.4.3.2.3 Enable Binding Types Oracle recommends enabling all the available binding types for the endpoints of the Identity Provider services; namely, POST, Redirect, and Artifact. Optionally you may select a preferred binding type.

7.4.3.2.4 Publish Your Site's Metadata File After you have configured the SAML 2.0 general services and Identity Provider services, publish your site's metadata file and distribute it to your federated partners, as described in Section 7.4.2.2, "Publishing and Distributing the Metadata File".

7.4.3.3 Create and Configure Web Single Sign-On Service Provider Partners

A SAML 2.0 Service Provider partner is an entity that consumes the SAML 2.0 assertions generated by the Identity Provider site. The configuration of Service Provider partners is available from the Administration Console, using the **Security Realms** > *RealmName* > **Providers** > **Credential Mapper** > *SAML2CredentialMapperName* > **Management** page.

The attributes that can be set on this console page can also be accessed programmatically via a set of Java interfaces, which are identified in the sections that follow.

See "Create a SAML 2.0 Web Single Sign-on Service Provider partner" in the *Oracle WebLogic Server Administration Console Online Help* for complete details about the specific steps for configuring a Service Provider partner. For a summary of the site information, signing certificates, and service endpoint information available when you configure a web single sign-on partner, see Section 7.4.5, "Viewing Partner Site, Certificate, and Service Endpoint Information".

7.4.3.3.1 Obtain Your Service Provider Partner's Metadata File Before you configure a Service Provider partner for web single sign-on, you need to obtain the partner's SAML 2.0 metadata file via a trusted and secure mechanism, such as encrypted email or an SSL-enabled FTP site. Your partner's metadata file describes the partner site and binding support, includes the partner's certificates and keys, contains your partner's SAML 2.0 service endpoints, and more. Copy the partner's metadata file into a location that can be accessed by each node in your domain configured for SAML 2.0.

The SAML 2.0 metadata file is described in Section 7.4.2.2, "Publishing and Distributing the Metadata File".

7.4.3.3.2 Create Partner and Enable Interactions To create and enable a Service Provider partner for web single sign-on:

1. From the Management tab of the SAML 2.0 Credential Mapping provider page, specify the partner's name and metadata file.

2. From the General tab of the partner configuration page, enable interactions between the partner and the WebLogic Server instance.

WebLogic Server provides the `com.bea.security.saml2.providers.registry.Partner` Java interface for configuring these attributes.

7.4.3.3.3 Configure How Assertions are Generated Optionally from the General tab of the partner configuration page in the console, you can configure the following attributes of the SAML 2.0 assertions generated specifically for this Service Provider partner:

- The Service Provider Name Mapper Class name

  This is the Java class that overrides the default username mapper class with which the SAML 2.0 Credential Mapping provider is configured in this security realm.

- Time to Live attributes

  The Time to Live attributes specify the interval of time during which the assertions generated for this partner are valid. These attributes prevent expired assertions from being used.

- Whether to generate attribute information that is included in assertions

  If enabled, the SAML 2.0 Credential Mapping provider adds, as attributes in the assertion, the groups to which the corresponding user belongs.

- Whether the assertions sent to this partner must be disposed of immediately after use

- Whether this server's signing certificate is included in assertions generated for this partner

WebLogic Server provides the `com.bea.security.saml2.providers.registry.SPPartner` Java interface for configuring these attributes.

7.4.3.3.4 Configure How Documents Are Signed You can use the General tab of the Service Provider partner configuration page to determine how the following documents exchanged with this partner must be signed:

- Assertions

  Operations on this attribute are available in the `com.bea.security.saml2.providers.registry.SPPartner` interface.

- Authentication requests

  Operations on this attribute are available in the `com.bea.security.saml2.providers.registry.WebSSOSPPartner` interface.

- Artifact requests

  Operations on this attribute are available in the `com.bea.security.saml2.providers.registry.WebSSOPartner` interface.

The attributes for specifying whether this partner accepts only signed assertions, or whether authentication requests must be signed, are read-only: they are derived from the partner's metadata file.

7.4.3.3.5 Configure Artifact Binding and Transport Settings Optionally, you also use the General tab of the Service Provider partner configuration page to configure the following:

- Whether SAML artifacts are delivered to this partner via the HTTP POST binding. If so, you may also specify the URI of a custom web application that generates the HTTP POST form for sending the SAML artifact.
- The URI of a custom web application that generate the HTTP POST form for sending request or response messages via the POST binding.

Operations on these attributes are available via the `com.bea.security.saml2.providers.registry.WebSSOPartner` Java interface.

For added security in the exchange of documents with this partner, you can also specify a client user name and password to be used by the Service Provider partner when connecting to the local site's binding using Basic authentication. This attribute is available via the `com.bea.security.saml2.providers.registry.BindingClientPartner` Java interface.

7.4.4 Configuring a Service Provider Site for SAML 2.0 Single Sign-On

This section presents the following topics:

- Section 7.4.4.1, "Configure the SAML 2.0 Identity Assertion Provider"
- Section 7.4.4.2, "Configure the SAML Authentication Provider"
- Section 7.4.4.3, "Configure SAML 2.0 General Services"
- Section 7.4.4.4, "Configure SAML 2.0 Service Provider Services"
- Section 7.4.4.5, "Create and Configure Web Single Sign-On Identity Provider Partners"

Note: Use of cookie-path.

As described in session-descriptor, the cookie-path element defines the session tracking cookie path. If not set, this element defaults to / (slash), where the browser sends cookies to all URLs served by WebLogic Server.

The WebLogic Server SAML 2.0 Service Providers require that the cookie-path be / (slash). If you set any other value for cookie-path, SSO fails for the SAML 2.0 Service Providers.

7.4.4.1 Configure the SAML 2.0 Identity Assertion Provider

In your security realm, create an instance of the SAML 2.0 Identity Assertion provider. The SAML 2.0 Identity Assertion provider is not part of the default security realm. The attributes you specify for the SAML 2.0 Identity Assertion provider include the following:

- Whether the replicated cache is enabled

  If you are configuring SAML 2.0 Identity Provider services in two or more server instances in the domain, this attribute must be enabled.
- A custom name mapper class that overrides the default SAML 2.0 assertion name mapper class

For more information about this security provider, see Section 5.9.5, "Configuring a SAML 2.0 Identity Assertion Provider for SAML 2.0".

7.4.4.2 Configure the SAML Authentication Provider

If you plan to enable virtual users, or consume attribute statements contained in assertions that you receive from your Identity Provider partners, you need to create and configure an instance of the SAML Authentication provider. For more information, see Section 5.7, "Configuring the SAML Authentication Provider".

7.4.4.3 Configure SAML 2.0 General Services

After configuring the SAML 2.0 Identity Assertion provider, and optionally the SAML Authentication provider, configure the SAML 2.0 general services, as described in Section 7.4.2, "Configuring SAML 2.0 General Services".

7.4.4.4 Configure SAML 2.0 Service Provider Services

Configuration of a WebLogic Server instance as a SAML 2.0 Service Provider site is controlled by the `SingleSignOnServicesMBean`. You can access the `SingleSignOnServicesMBean` using the WebLogic Scripting Tool (WLST), or through the Administration Console using the **Environment** > **Servers** > *ServerName* > **Configuration** > **Federation Services** > **SAML 2.0 Service Provider** page.

You configure the SAML 2.0 Service Provider site attributes as summarized in the sections that follow. For more information about these configuration tasks, see "Configure SAML 2.0 Service Provider services" in the *Oracle WebLogic Server Administration Console Online Help*.

7.4.4.4.1 Enable the SAML 2.0 Service Provider Site From the Federation Services: SAML 2.0 Identity Provider page in the console, allow the WebLogic Server instance to serve as a Service Provider site by setting the Enabled attribute to `true`.

7.4.4.4.2 Specify How Documents Must Be Signed Optionally you may enable the attributes that set the following document signing requirements:

- Whether authentication requests sent to Identity Provider partners are signed
- Whether assertions received from Identity Provider partners are signed

7.4.4.4.3 Specify How Authentication Requests Are Managed Optionally you may enable the following attributes of the authentication request cache:

- Maximum cache size
- Time-out value for authentication requests, which establishes the time interval beyond which stored authentication requests are expired

7.4.4.4.4 Enable Binding Types Oracle recommends enabling all the available binding types for the endpoints of the Service Provider services; namely, POST, and Artifact. Optionally you may specify a preferred binding type.

7.4.4.4.5 Set Default URL Optionally, you may specify the URL to which unsolicited authentication responses are sent if they do not contain an accompanying target URL.

7.4.4.5 Create and Configure Web Single Sign-On Identity Provider Partners

A SAML 2.0 Identity Provider partner is an entity that generates SAML 2.0 assertions consumed by the Service Provider site. The configuration of Identity Provider partners is available from the Administration Console, using the **Security Realms** > *RealmName* > **Providers** > **Authentication** > *SAML2IdentityAsserterName* > **Management** page.

The attributes that can be set on this console page can also be accessed programmatically via a set of Java interfaces, which are identified in the sections that follow.

See "Create a SAML 2.0 Web Single Sign-on Identity Provider partner" in the *Oracle WebLogic Server Administration Console Online Help* for complete details about the specific steps for configuring a Service Provider partner.

For a summary of the site information, signing certificates, and service endpoint information available when you configure a web single sign-on partner, see Section 7.4.5, "Viewing Partner Site, Certificate, and Service Endpoint Information".

The following sections summarize tasks for configuring an Identity Provider partner.

7.4.4.5.1 Obtain Your Identity Provider Partner's Metadata File Before you configure an Identity Provider partner for web single sign-on, you need to obtain the partner's SAML 2.0 metadata file via a trusted and secure mechanism, such as encrypted email or an SSL-enabled FTP site. Your partner's metadata file describes that partner site and binding support, includes the partner's certificates and keys, and so on. Copy the partner's metadata file into a location that can be accessed by each node in your domain configured for SAML 2.0.

The SAML 2.0 metadata file is described in Section 7.4.2.2, "Publishing and Distributing the Metadata File".

7.4.4.5.2 Create Partner and Enable Interactions To create an Identity Provider partner and enable interactions for web single sign-on:

- From the Management tab of the SAML 2.0 Identity Assertion configuration page, specify the partner's name and metadata file.
- From the General tab of the partner configuration page, enable interactions between the partner and the WebLogic Server instance.

WebLogic Server provides the `com.bea.security.saml2.providers.registry.Partner` Java interface for configuring these attributes.

7.4.4.5.3 Configure Authentication Requests and Assertions Optionally, you can configure the following attributes of the authentication requests generated for, and assertions received from, this Identity Provider partner:

- The Identity Provider Name Mapper Class name

  This is the custom Java class that overrides the default username mapper class with which the SAML 2.0 Identity Assertion provider is configured in this security realm. The custom class you specify is used only for identities contained in assertions received from this particular partner.

  Operations on this attribute are available in the `com.bea.security.saml2.providers.registry.IdPPartner` Java interface.

- Whether the identities contained in assertions received from this partner are mapped to virtual users in the security realm

  Note: To use this attribute, you must have a SAML Authentication provider configured in the realm.

Operations on this attribute are available in the `com.bea.security.saml2.providers.registry.IdPPartner` Java interface.

- Whether to consume attribute information contained in assertions received from this partner

  If enabled, the SAML 2.0 Identity Assertion provider extracts attribute information from the assertion, which it uses in conjunction with the SAML Authentication provider (which must be configured in the security realm) to determine the groups in the security realm to which the corresponding user belongs.

  Operations on this attribute are available in the `com.bea.security.saml2.providers.registry.IdPPartner` Java interface.

- Whether authentication requests sent to this Identity Provider partner must be signed. This is a read-only attribute that is derived from the partner's metadata file.

  Operations on this attribute are available in the `com.bea.security.saml2.providers.registry.WebSSOIdPPartner` Java interface.

- Whether SAML artifact requests received from this Identity Provider partner must be signed.

  Operations on this attribute are available in the `com.bea.security.saml2.providers.registry.WebSSOIdPPartner` Java interface.

7.4.4.5.4 Configure Redirect URIs You can configure a set of URIs that, if invoked by an unauthenticated user, cause the user request to be redirected to the Identity Provider partner where the user can be authenticated.

> **Note:** If you configure one or more redirect URIs, remember to set a security policies on them as well; otherwise the web container will not attempt to authenticate the user and, consequently, not redirect the user's request to the Identity Provider partner.

WebLogic Server provides the `com.bea.security.saml2.providers.registry.WebSSOIdPPartner` Java interface for configuring this attribute.

7.4.4.5.5 Configure Binding and Transport Settings Optionally, you also use the General tab of the Service Provider partner configuration page to configure the following:

- Whether SAML artifacts are delivered to this partner via the HTTP POST method. If so, you may also specify the URI of a custom web application that generates the HTTP POST form for sending the SAML artifact.
- The URL of the custom web application that generates the POST form for carrying the SAML response for POST bindings to this Identity Provider partner.
- The URL of the custom web application that generates the POST form for carrying the SAML response for Artifact bindings to this Identity Provider partner.

Operations on these attributes are available via the `com.bea.security.saml2.providers.registry.WebSSOPartner` Java interface.

For added security in the exchange of documents with this partner, you can also specify a client user name and password to be used by this Identity Provider partner when connecting to the local site's binding using Basic authentication. This attribute is available via the

`com.bea.security.saml2.providers.registry.BindingClientPartner` Java interface.

7.4.5 Viewing Partner Site, Certificate, and Service Endpoint Information

When you configure SAML 2.0 partners, the partner configuration pages displayed by the Administration Console include tabs for viewing and configuring the following additional information about the partner:

- The Site tab displays information about the Service Provider partner, which is derived from the partner's metadata file. The data in this tab is read-only.

  WebLogic Server provides the `com.bea.security.saml2.providers.registry.MetadataPartner` Java interface for partner site information.

- The Single Sign-On Signing Certificate tab displays details about the partner's signing certificate, which are also derived from the partner's metadata file. The data in this tab is read-only.

  Operations on these attributes are available from the `com.bea.security.saml2.providers.registry.WebSSOPartner` Java interface.

- The Transport Layer Client Certificate tab displays partner's transport layer client certificate. You can optionally import this certificate by clicking **Import Certificate from File**.

  Operations on this attribute are available from the `com.bea.security.saml2.providers.registry.BindingClientPartner` Java interface.

- When configuring Service Provider partners, the Assertion Consumer Service Endpoints tab is available, which displays the Service Provider partner's ACS endpoints. This data is also available from the `com.bea.security.saml2.providers.registry.WebSSOSPPartner` Java interface.

- When configuring Identity Provider partners, the Single Sign-On Service Endpoints tab is available, which displays the Identity Provider partner's single sign-on service endpoints. This data is also available from the `com.bea.security.saml2.providers.registry.WebSSOIdPPartner` Java interface.

- The Artifact Resolution Service Endpoints tab displays the partner's ARS endpoints. This data is also available from the `com.bea.security.saml2.providers.registry.WebSSOPartner` Java interface.

7.4.6 Web Application Deployment Considerations for SAML 2.0

When deploying web applications for SAML-based SSO in a clustered environment, note the considerations presented in the following sections for preventing SAML-based single sign-on from failing:

- Section 7.4.6.1, "Deployment Descriptor Recommendations"
- Section 7.4.6.2, "Login Application Considerations for Clustered Environments"
- Section 7.4.6.3, "Enabling Force Authentication and Passive Attributes is Invalid"

7.4.6.1 Deployment Descriptor Recommendations

Note the following recommendations regarding the use of the following elements in deployment descriptor files:

- relogin-enabled
- cookie-name

7.4.6.1.1 Use of relogin-enabled with CLIENT-CERT Authentication If a user logs in to a web application and tries to access a resource for which that user is not authorized, an HTTP FORBIDDEN (403) response is generated. This is standard web application behavior. However, for backwards compatibility with earlier releases, WebLogic Server permits web applications to use the `relogin-enabled` element in the `weblogic.xml` deployment descriptor file, so that the response to an access failure results in a request to authenticate. In certain circumstances, it can cause SAML 2.0 based web single sign-on to fail.

Normally, the SAML 2.0 Assertion Consumer Service (ACS) logs the user into the application and redirects the user request to the target web application. However, if that web application is enabled for SAML 2.0 single sign-on, is protected by `CLIENT-CERT` authentication, and has the `relogin-enabled` deployment descriptor element set to true, an infinite loop can occur in which a request to authenticate a user is issued repeatedly. This loop can occur when a user is logged in to the web application and attempts to access a resource for which the user is not permitted: instead of generating a FORBIDDEN message, a new authentication request is generated that triggers another SAML 2.0 based web single sign-on attempt.

To prevent this situation from occurring in a web application that is protected by `CLIENT-CERT` authentication, either remove the `relogin-enabled` deployment descriptor element for the web application, or set the element to `false`. This enables standard web application authentication behavior.

7.4.6.1.2 Use of Non-default Cookie Name When the Assertion Consumer Service logs in the Subject contained in an assertion, an HTTP servlet session is created using the default cookie name `JSESSIONID`. After successfully processing the assertion, the ACS redirects the user's request to the target web application. If the target web application uses a cookie name other than `JSESSIONID`, the Subject's identity is not propagated to the target web application. As a result, the servlet container treats the user as if unauthenticated, and consequently issues an authentication request.

To avoid this situation, do not change the default cookie name when deploying web applications in a domain that are intended to be accessed by SAML 2.0 based single sign-on.

7.4.6.2 Login Application Considerations for Clustered Environments

Note the following two login limitations that are rare in clustered environments, but if they occur, they may prevent a single sign-on session from succeeding.

- When an Identity Provider's single sign-on service receives an authentication request, it redirects that request to the login application to authenticate the user. The login application must execute on the same cluster node as that single sign-on service. If not, the Identity Provider is unable to produce a SAML 2.0 assertion even if the authentication succeeds.

  Under normal circumstances, the login application executes on the same node as the single sign-on service, so likelihood of the authentication request being redirected to a login application executing on a different node in the domain is very small. However, it may happen if an authentication request is redirected by a cluster node different than the one hosting the login application. You can almost always prevent this situation from occurring if you configure the Identity Provider to use the default login URI with Basic authentication.

- When the SAML 2.0 Assertion Consumer Service (ACS) successfully consumes an assertion, it logs in the Subject represented by the assertion. The ACS then redirects the user request to the target application. Normally, the target application executes on the same node as the ACS. However, in rare circumstances, the target application to which is the user request is redirected executes on a cluster node other than the one hosting the ACS on which the login occurred. When this circumstance occurs, the identity represented by the assertion is not propagated to the target application node. The result is either another attempt at the single sign-on process, or denied access.

  Because the target application executes on the same node as the ACS, this situation is expected to occur very rarely.

7.4.6.3 Enabling Force Authentication and Passive Attributes is Invalid

When configuring SAML 2.0 Service Provider services, enabling both the Force Authentication and Passive attributes is an invalid configuration that WebLogic Server is unable to detect. If both these attributes are enabled, and an unauthenticated user attempts to access a resource that is hosted at the Service Provider site, an exception is generated and the single sign-on session fails.

Note that the Force Authentication attribute has no effect because SAML logout is not supported in WebLogic Server. So even if the user is already authenticated at the Identity Provider site and Force Authentication is enabled, the user is not forced to authenticate again at the Identity Provider site.

7.5 Enabling Debugging for SAML 1.1 and 2.0

You can enable debugging for a web application that uses SAML for SSO by setting the desired `ServerDebug` configuration attributes to `true`. WebLogic Server provides a variety of ways to do this, as explained in the following sections:

- Section 7.5.1, "About SAML Debug Scopes and Attributes"
- Section 7.5.2, "Enabling Debugging Using the Command Line"
- Section 7.5.3, "Enabling Debugging Using the WebLogic Server Administration Console"
- Section 7.5.4, "Enabling Debugging Using the WebLogic Scripting Tool"
- Section 7.5.5, "Sending Debug Messages to Standard Out"

7.5.1 About SAML Debug Scopes and Attributes

Table 7–2 and Table 7–3 list and describe the registered debug scopes and attributes provided in WebLogic Server for SAML 1.1 and 2.0.

Table 7–2 SAML 1.1 Debug Scopes and Attributes

Scope	Attribute	Description
`weblogic.security.saml.atn`	`DebugSecuritySAMLAtn`	Prints information about SAML 1.1 authentication provider processing.
`weblogic.security.saml.credmap`	`DebugSecuritySAMLCredMap`	Prints information about SAML 1.1 credential mapping provider processing.

Table 7–2 (Cont.) SAML 1.1 Debug Scopes and Attributes

Scope	Attribute	Description
weblogic.security.saml.lib	DebugSecuritySAMLLib	Prints information about SAML 1.1 library processing.
weblogic.security.saml.service	DebugSecuritySAMLService	Prints information about SAML 1.1 SSO profile services.

Table 7–3 SAML 2.0 Debug Scopes and Attributes

Scope	Attribute	Description
weblogic.security.saml2.atn	DebugSecuritySAML2Atn	Prints information about SAML 2.0 authentication provider processing.
weblogic.security.saml2.credmap	DebugSecuritySAML2CredMap	Prints information about SAML 2.0 credential mapping provider processing.
weblogic.security.saml2.lib	DebugSecuritySAML2Lib	Prints information about SAML 2.0 library processing.
weblogic.security.saml2.service	DebugSecuritySAML2Service	Prints information about SAML 2.0 SSO profile services.

7.5.2 Enabling Debugging Using the Command Line

You can enable debug scopes or attributes by passing them as options in the command that starts WebLogic Server. The command line options you can use for enabling SAML debugging by attribute are listed in Table 7–4.

Table 7–4 Command Line Options for SAML Debugging

SAML Version	Available Command Line Options for Debugging
SAML 1.1	-Dweblogic.debug.DebugSecuritySAMLAtn=true -Dweblogic.debug.DebugSecuritySAMLCredMap=true -Dweblogic.debug.DebugSecuritySAMLLib=true -Dweblogic.debug.DebugSecuritySAMLService=true
SAML 2.0	-Dweblogic.debug.DebugSecuritySAML2Atn=true -Dweblogic.debug.DebugSecuritySAML2CredMap=true -Dweblogic.debug.DebugSecuritySAML2Lib=true -Dweblogic.debug.DebugSecuritySAML2Service=true

This method for enabling SAML debugging is static and can only be used at server startup.

7.5.3 Enabling Debugging Using the WebLogic Server Administration Console

To configure SAML debugging using the WebLogic Server Administration Console, complete the following steps:

1. If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit** (see "Use the Change Center").
2. In the left pane of the console, expand Environment and select Servers.

3. On the Summary of Servers page, click the server on which you want to enable or disable debugging to open the settings page for that server.
4. Click **Debug**.
5. Expand **weblogic**.
6. Expand **security**.
7. Enable SAML debugging as follows:
   - To enable the SAML 1.1 debug scope, which encompasses all the SAML 1.1 attributes, select **saml**, then click **Enable**.
   - To enable one or more individual SAML 1.1 debug attributes, expand **saml**, expand the scope of the desired attribute, select the desired individual SAML 1.1 attribute, then click **Enable**. For example, expand **saml**, expand **atn**, and select the **DebugSecuritySAMLAtn** attribute to debug SAML 1.0 authentication processing.
   - To enable the SAML 2.0 debug scope, which encompasses all the SAML 2.0 attributes, select **saml2**, then click **Enable**.
   - To enable one or more individual SAML 2.0 debug attributes, expand **saml2**, expand the scope of the desired attribute, select the desired individual SAML 2.0 attribute, then click **Enable**. For example, expand **saml2**, expand **credmap**, and select the **DebugSecuritySAML2Credmap** attribute to debug SAML 2.0 credential mapping provider processing.

   For a description of each registered SAML debug attribute, see Section 7.5.1, "About SAML Debug Scopes and Attributes".
8. To activate these changes, in the Change Center of the Administration Console, click **Activate Changes** (see "Use the Change Center").

Changes to SAML debug scopes and attributes take effect immediately — no restart is necessary. Using the Administration Console to enable or disable SAML debugging is dynamic and can be used while the server is running. For more information, see "Define debug settings" in the *Oracle WebLogic Server Administration Console Online Help*.

7.5.4 Enabling Debugging Using the WebLogic Scripting Tool

You can use the WebLogic Scripting Tool (WLST) to configure SAML debugging attributes. For example, the following command runs a program for setting debugging attributes called `debug.py`:

```
java weblogic.WLST debug.py
```

The `debug.py` program contains the following code, which enables debugging for the attribute `DebugSecuritySAMLAtn`.

```
user='user1'
password='password'
url='t3://localhost:7001'
connect(user, password, url)
edit()
cd('Servers/myserver/ServerDebug/myserver')
startEdit()
set('DebugSecuritySAMLAtn','true')
save()
activate()
```

Note that you can also use WLST from Java. The following example shows the source file of a Java program that sets the `DebugSecuritySAMLAtn` debugging attribute:

```
import weblogic.management.scripting.utils.WLSTInterpreter;
import java.io.*;
import weblogic.jndi.Environment;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class test {
        public static void main(String args[]) {
       try {
              WLSTInterpreter interpreter = null;
              String user="user1";
              String pass="pw12ab";
              String url ="t3://localhost:7001";
              Environment env = new Environment();
              env.setProviderUrl(url);
              env.setSecurityPrincipal(user);
              env.setSecurityCredentials(pass);
              Context ctx = env.getInitialContext();

              interpreter = new WLSTInterpreter();
              interpreter.exec
                     ("connect('"+user+"','"+pass+"','"+url+"')");
              interpreter.exec("edit()");
              interpreter.exec("startEdit()");
              interpreter.exec
                     ("cd('Servers/myserver/ServerDebug/myserver')");
              interpreter.exec("set('DebugSecuritySAMLAtn','true')");
              interpreter.exec("save()");
              interpreter.exec("activate()");

       } catch (Exception e) {
       System.out.println("Exception "+e);
       }
       }
}
```

Using the WLST is a dynamic method and can be used to enable debugging while the server is running.

7.5.5 Sending Debug Messages to Standard Out

Messages corresponding to enabled debug attributes are sent to the server log file. Optionally, you can also send debug messages to standard out by passing the `StdoutSeverity=Debug` attribute on the LogMBean in the command to start WebLogic Server. For example, `-Dweblogic.log.StdoutSeverity=Debug`.

For more information, see "Message Output and Logging" in *Command Reference for Oracle WebLogic Server*.

8 Migrating Security Data

This chapter describes how to export security data from one security realm or security provider and import the data into another realm or provider.

This chapter includes the following sections:

8.1 Overview of Security Data Migration

WebLogic security realms persist different kinds of security data — for example, users and groups (for the WebLogic Authentication provider), security policies (for the XACML Authorization provider), security roles (for the XACML Role Mapping provider), and credential maps (for the WebLogic Credential Mapping provider). When you configure a new security realm or a new security provider, you may prefer to use the security data from your existing realm or provider, rather than recreate all the users, groups, policies, roles, and credential maps. Several WebLogic security providers support security data migration. This means you can export security data from one security realm, and import it into a new security realm. You can migrate security data for each security provider individually, or migrate security data for all the WebLogic security providers at once (that is, security data for an entire security realm). Note that you can only migrate security data from one provider to another if the providers use the same data format. See Section 8.3, "Formats and Constraints Supported by WebLogic Security Providers". You migrate security data through the WebLogic Administration Console or by using the WebLogic Scripting Tool (WLST).

Migrating security data may be helpful when you:

- Transition from development to production mode.
- Copy production mode security configurations to security realms in new WebLogic domains.
- Move data from one security realm to a new security realm in the same WebLogic domain, where one or more of the default WebLogic security providers will be replaced with new security providers.

The remainder of this section describes security migration concepts, the formats and constraints supported by the WebLogic security providers, and steps for migrating security data with WLST.

To migrate security data with the WebLogic Administration Console, see the following topics in the *Oracle WebLogic Server Administration Console Online Help*:

- "Export data from security realms"
- "Import data into security realms"
- "Export data from a security provider"
- "Import data into a security provider"

8.2 Migration Concepts

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

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

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

8.3 Formats and Constraints Supported by WebLogic Security Providers

In order for security data to be exported and imported between security providers, both security providers must process the same format. Some data formats used for the WebLogic Server security providers are unpublished; therefore, you cannot currently migrate security data from a WebLogic security provider to a custom security provider, or vice versa, using the unpublished formats.

WebLogic security providers support the import and export formats provided in Table 8–1.

Table 8–1 Import and Export Formats Supported by the WebLogic Security Providers

WebLogic Provider	Supported Format
WebLogic Authentication provider	DefaultAtn—unpublished format
XACML Authorization Provider	XACML—standard XACML 2.0 format DefaultAtz—unpublished format
WebLogic Authorization Provider	DefaultAtz—unpublished format
XACML Role Mapping Provider	XACML—standard XACML 2.0 format DefaultRoles—unpublished format
WebLogic Role Mapping Provider	DefaultRoles—unpublished format
WebLogic Credential Mapping Provider	DefaultCreds—unpublished format

Table 8–1 (Cont.) Import and Export Formats Supported by the WebLogic Security

WebLogic Provider	Supported Format
SAML Identity Asserter V2 SAML Credential Mapping Provider V2	XML Partner Registry—An XML format defined by the SAML partner registry schema JKS Key Store—A key store file format for importing and exporting partner certificates only LDIF Template—LDIF format

WebLogic security providers support the import and export constraints provided in Table 8–2.

Table 8–2 Constraints Supported by the WebLogic Security Providers

WebLogic Security Provider	Supported Constraints	Description
Default Authentication	users groups	Export all users or all groups
■ XACML Authorization ■ WebLogic Authorization ■ XACML Role Mapping ■ WebLogic Role Mapping	none	N/A
WebLogic Credential Mapping	passwords	With the constraint passwords=cleartext, passwords will be exported in clear text. Otherwise, they will be exported in encrypted form.
■ SAML Identity Asserter V2 ■ SAML Credential Mapping V2	partners	Which partners to import or export. The constraint value can be one of: ■ all—all partners ■ none—no partners ■ list—only listed partners ■ enabled—only enabled partners ■ disabled—only disabled partners
■ SAML Identity Asserter V2 ■ SAML Credential Mapping V2	certificates	Which certificates to import or export. The constraint value can be one of the following: ■ all—all certificates ■ none—no certificates ■ list—only listed certificates ■ referenced—only certificates referenced by a partner
■ SAML Identity Asserter V2 ■ SAML Credential Mapping V2	passwords	With the constraint passwords=cleartext, passwords will be exported in clear text. Otherwise, they will be exported in encrypted form.

Table 8–2 (Cont.) Constraints Supported by the WebLogic Security Providers

WebLogic Security Provider	Supported Constraints	Description
■ SAML Identity Asserter V2 ■ SAML Credential Mapping V2	importMode	Specifies how to resolve name conflicts between the imported data and existing data in the SAML registry. The constraint value can be one of the following: ■ fail—the import operation will fail if conflicts are detected (default) ■ rename—rename the imported entry that conflicts ■ replace—replace the existing entry with the conflicting imported entry

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

Note: By default, the WebLogic Authentication provider stores passwords using a one-way hash. Passwords that have been encrypted by this provider cannot be unencrypted when you export data even if you use the `passwords=cleartext` constraint. If you want to be able to export passwords in clear text from this provider, you must set the Enable Password Digests attribute to `true` prior to creating or updating those passwords. For more information, see "Default Authentication Provider: Provider Specific" in *Oracle WebLogic Server Administration Console Online Help*.

8.4 Migrating Data with WLST

You can use the WebLogic Scripting Tool (WLST) to export and import data from a security provider. Access the Runtime MBean for the security provider and use its `importData` or `exportData` operation. For example, you might use WLST to import data using commands like these:

```
domainRuntime()
cd('DomainServices/DomainRuntimeService/DomainConfiguration/mydomain
/SecurityConfiguration/mydomain/DefaultRealm/myrealm/path-to-MBean/mbeanname')
cmo.importData(format,filename,constraints)
```

where:

- *mbeanname*—Name of the security provider MBean.
- *format*—A format that is valid for the particular security provider. See Table 8–1.

- *filename*—The directory location and filename in which to export or import the security data. Remember that, regardless of whether you are using a UNIX or Windows operating system, you need to use a forward slash, not a back slash, as a path separator for pathname arguments in WLST commands.
- *constraints*—The constraints that limit the data to be exported or imported

For more information, see *Understanding the WebLogic Scripting Tool*.

9

Managing the RDBMS Security Store

This chapter describes the WebLogic Server option of using an external RDBMS as a datastore for the authorization, role mapping, credential mapping, and certificate registry providers. This datastore, called the RDBMS security store, is strongly recommended for using SAML 2.0 services in two or more WebLogic Server instances in that domain, such as in a cluster.

This datastore, called the RDBMS security store, is required by the SAML 2.0 security providers in production environments so that the data they manage can be synchronized across all the WebLogic Server instances that share that data. (Use LDAP as the security store with the SAML 2.0 security providers only in development environments.)

> **Note:** In order to use the RDBMS security store, the preferred approach is first to create a domain in which the external RDBMS server is configured. Prior to booting the domain, you create the tables in the datastore that are required by the RDBMS security store. The WebLogic Server installation directory contains a set of SQL scripts that create these tables for each supported database.

This chapter includes the following sections:

For the most up-to-date details about the specific database systems that are supported for use as the RDBMS security store for WebLogic Server, see the Oracle Fusion Middleware Supported System Configurations page on Oracle Technology Network.

9.1 Security Providers that Use the RDBMS Security Store

The following security providers use the RDBMS security store if that store is configured in a domain:

- XACML Authorization provider
- XACML Role Mapping provider
- The following providers for SAML 1.1:
  - SAML Identity Assertion provider V2
  - SAML Credential Mapping provider V2

- The following providers for SAML 2.0:
  - SAML 2.0 Identity Assertion provider
  - SAML 2.0 Credential Mapping provider
- WebLogic Credential Mapping provider
- PKI Credential Mapping provider
- Certificate Registry

When the RDBMS security store is configured in a domain, an instance of any of the preceding security providers that has been created in the security realm automatically uses only the RDBMS security store as a datastore, and not the embedded LDAP server. WebLogic security providers configured in the domain that are not among those in the preceding list continue to use their respective default stores; for example, the Default Authentication provider continues to use the embedded LDAP server.

Oracle recommends that you configure the RDBMS security store at the time of domain creation. WebLogic Server includes the `RDBMSSecurityStoreMBean`, which is the interface for configuring the RDBMS security store via the WebLogic Scripting Tool (WLST). (The Configuration Wizard does not provide the ability to configure the RDBMS security store.)

9.2 Configuring the RDBMS Security Store

To create and configuring the RDBMS security store in a domain, complete the tasks described in the following sections:

- Section 9.2.1, "Create a Domain with the RDBMS Security Store"
- Section 9.2.2, "Create RDBMS Tables in the Security Datastore"
- Section 9.2.3, "Configure a JMS Topic for the RDBMS Security Store"

9.2.1 Create a Domain with the RDBMS Security Store

To use the RDBMS security store in a domain, Oracle recommends that you configure the RDBMS security store at the time you create that domain. Modifying an existing domain in place to use the RDBMS security store is possible; however, it is not recommended because if the database connection is not configured correctly, the policies necessary for granting access to the domain could become unavailable, resulting in a domain that cannot be used.

You configure the RDBMS security store by using the WebLogic Scripting Tool (WLST) Offline. (The Configuration Wizard does not provide the ability to configure the RDBMS security store.) Operations for creating and configuring the RDBMS security store are available via the `RDBMSSecurityStoreMBean`. You also need to configure the connection properties for the database that serves as the RDBMS security store as explained in the following sections.

9.2.1.1 Specifying Database Connection Properties

When configuring the RDBMS security store, you need to specify or configure the following:

- RDBMS type

  For information about the databases that are supported for containing the RDBMS security store, see the Oracle Fusion Middleware Supported System Configurations page on Oracle Technology Network.

- JDBC driver and class name for connecting to the RDBMS
- RDBMS name, host, port, and URL
- Username and password of the domain user who can access the RDBMS system

> **Caution:** For clarity, the WLST examples provided in this section show passing username and password credentials of the RDBMS system user in clear text. However, you should avoid entering clear-text passwords in WLST commands in general, and you should especially avoid saving on disk WLST scripts that include clear-text passwords. In these instances you should use a mechanism for passing encrypted passwords instead.

- Optionally, any properties that need to be passed to the RDBMS system

The parameters that you specify in the JDBC driver connection properties attribute must be a comma-separated list. The following examples show the use of WLST to configure the database connection properties for Oracle, MS-SQL, and DB2.

9.2.1.1.1 Oracle Example Example 9–1 shows an example of configuring Oracle for the RDBMS security store.

Example 9–1 Configuring Oracle for the RDBMS Security Store

```
create('base_domain','SecurityConfiguration')
cd('/SecurityConfiguration/base_domain')
a=get('DefaultRealm')
cd('Realm/myrealm')
rdbms = create("myRDBMSSecurityStore", "RDBMSSecurityStore")
rdbms.setUsername('ortiz')
rdbms.setPassword('password')
rdbms.setConnectionURL('jdbc:bea:oracle://avitek21:1521')
rdbms.setDriverName('weblogic.jdbc.oracle.OracleDriver')
rdbms.setConnectionProperties('user=ortiz,portNumber=1521,SID=pint101a,serverName=
avitek21')
```

9.2.1.1.2 MS-SQL Example Example 9–2 shows an example of configuring MS-SQL for the RDBMS security store.

Example 9–2 Configuring MS-SQL for the RDBMS Security Store

```
create('base_domain','SecurityConfiguration')
cd('/SecurityConfiguration/base_domain')
a=get('DefaultRealm')
cd('Realm/myrealm')
rdbms = create("myRDBMSSecurityStore", "RDBMSSecurityStore")
rdbms.setUsername('garnett')
rdbms.setPassword('password')
rdbms.setConnectionURL('jdbc:bea:sqlserver://avitek6:1433')
rdbms.setDriverName('weblogic.jdbc.sqlserver.SQLServerDriver')
rdbms.setConnectionProperties('user=garnett,portNumber=1433,databaseName=wls3,serv
erName=avitek6')
```

9.2.1.1.3 DB2 Example Example 9–3 shows an example of configuring DB2 for the RDBMS security store.

Note: If you choose DB2, you have the option of selecting the WebLogic Type 4 JDBC driver for DB2 that is provided in WebLogic Server. However, if you use this JDBC driver, you must also specify the additional property `BatchPerformanceWorkaround` and set it to `true`. If you do not set the `BatchPerformanceWorkaround` to `true` in this configuration, WebLogic Server may fail to boot, generating a SecurityServiceException message.

Example 9–3 Configuring DB2 for the RDBMS Security Store

```
create('base_domain','SecurityConfiguration')
cd('/SecurityConfiguration/base_domain')
a=get('DefaultRealm')
cd('Realm/myrealm')
rdbms = create("myRDBMSSecurityStore", "RDBMSSecurityStore")
rdbms.setUsername('brady')
rdbms.setPassword('password')
rdbms.setConnectionURL('jdbc:bea:db2://avitek3:50000')
rdbms.setDriverName('weblogic.jdbc.db2.DB2Driver')
rdbms.setConnectionProperties('user=brady,portNumber=50000,databaseName=wls,server
Name=avitek3,batchPerformanceWorkaround=true')
```

For more information about specifying connection properties for the WebLogic Type 4 JDBC driver for DB2, see "Using DataDirect Documentation" in *Developing JDBC Applications for Oracle WebLogic Server*.

9.2.1.1.4 For More Information About Default Connection Properties Internally, the RDBMS security store uses Oracle Kodo to connect to and interoperate with the database using the WebLogic Type 4 JDBC driver for DB2. The attributes set on the `RDBMSSecurityStoreMBean` are converted into attributes set on the properties of Kodo's `javax.sql.DataSource` implementation.

For more information about these attributes, see the following topics:

- For more information about the attributes you can set on the `RDBMSSecurityStoreMBean`, see "RDBMSSecurityStoreMBean" in the *MBean Reference for Oracle WebLogic Server*.
- For information about the default database connection properties in the Kodo DataDource, see "Using the Kodo DataSource" in the JDBC chapter of the *Kodo JPA/JDO Reference Guide*.

9.2.1.2 Testing the Database Connection

When you configure the RDBMS security, Oracle strongly recommends testing the database connection to verify that the connection is set up properly. If there were a problem with the database connection, you might not be able subsequently to boot the domain if the security providers that control access to that domain are unable to obtain the necessary security policies.

9.2.2 Create RDBMS Tables in the Security Datastore

Prior to booting the domain, the database administrator needs to run the SQL script that creates the RDBMS tables in the datastore used by the RDBMS security store. A set of SQL scripts for creating these tables for, and also removing them from, each supported RDBMS system is available in the following WebLogic Server installation directory:

WL_HOME/server/lib

When running the appropriate SQL script for the database serving as the RDBMS security store, be sure to specify the same connection properties, including the credentials of the user who has access, the database URL, etc., as specified for that RDBMS during domain creation.

Table 9–1 identifies the name of each of these SQL scripts.

Table 9–1 SQL Scripts for Creating and Removing RDBMS Datastore Tables

RDBMS	Script for Creating Datastore Tables	Script for Removing Datastore Tables
Oracle 9i, 10g, 11g	rdbms_security_store_oracle.sql	rdbms_security_store_oracle_remove.sql
MS-SQL 2000, 2005	rdbms_security_store_sqlserver.sql	rdbms_security_store_sqlserver_remove.sql
DB2 9.2, 9.5	rdbms_security_store_db2.sql	rdbms_security_store_db2_remove.sql
Derby	rdbms_security_store_derby.sql	rdbms_security_store_derby_remove.sql

9.2.3 Configure a JMS Topic for the RDBMS Security Store

If the RDBMS security store is configured in a domain that includes two or more WebLogic Server instances, or a cluster, Oracle strongly recommends that you also perform the following tasks:

1. Enable JMS notifications for that domain.
2. Configure a JMS topic that can be used by the RDBMS security store.

JMS notifications enable the security data that is contained in the RDBMS security store, and that is managed by security providers in the realm, to be synchronized among all server instances in the domain.

Caution: If you do not configure a JMS topic that can be used by the RDBMS security store when configured in a multi-server or clustered domain, care should be taken when making security policy or security configuration updates. If no JMS topic is configured, it may be necessary to reboot the domain to ensure that all server instances function consistently with regards to those security updates.

You can enable JMS notifications by booting the domain in which the RDBMS security store has been configured, and configuring attributes on the `RDBMSSecurityStoreMBean` via either of the following mechanisms:

- WebLogic Scripting Tool
- The **Security Realms** > *RealmName* > **RDBMS Security Store** page in the Administration Console

The attributes of the `RDBMSSecurityStoreMBean` that must be set to enable JMS notifications are listed and described in Table 9–2.

Table 9–2 RDBMSSecurityStoreMBean Attributes for Configuring a JMS Topic

Attribute Name	Description
JMSTopic	The JMS topic to which the Kodo remote commit provider should publish notifications and subscribe for notifications sent from other JVMs. The target JMS topic needs to be pre-deployed.
JMSTopicConnectionFactory	The JNDI name of a javax.jms.TopicConnectionFactory instance to use for finding JMS topics. The topic "Connection Factory Configuration" in *Administering JMS Resources for Oracle WebLogic Server* describes the WebLogic JMS connection factory, weblogic.jms.ConnectionFactory, which is a javax.jms.TopicConnectionFactory instance. Refer to this topic for information about configuring a connection factory.
NotificationProperties	A comma-delimited list of key-value properties to pass to the JNDI InitialContext on construction, in the form of *xx*Key=*xx*Value, *xx*Key=*xx*Value. The following properties must be specified: ■ java.naming.provider.url — Property for specifying configuration information for the service provider to use. The value of the property should contain a URL string. For example: iiops://*localhost*:7002 ■ java.naming.factory.initial — Property for specifying the initial context factory to use. The value of the property should be the fully-qualified class name of the factory class that will create an initial context. For example: weblogic.jndi.WLInitialContextFactory
JNDIUserName	The identity of any valid user in the security realm who has access to JNDI.
JNDIPassword	The password of the user specified in the JNDIUserName attribute.
JMSExceptionReconnectAttempts	The number of reconnect attempts to be made if the JMS system notifies Kodo of a serious connection error. The default is 0, which causes an error to be logged, but does not result in a reconnect attempt.

For more information, see the following topics:

- "Configure topics" in the *Oracle WebLogic Server Administration Console Online Help*
- "Configuring Basic JMS System Resources" in *Administering JMS Resources for Oracle WebLogic Server*
- "Configure the RDBMS security store" in the *Oracle WebLogic Server Administration Console Online Help*
- "RDBMSSecurityStoreMBean" in the *MBean Reference for Oracle WebLogic Server*

9.2.3.1 Configuring JMS Connection Recovery in the Event of Failure

Normally, the WebLogic Security Service contained in each WebLogic Server instance in a multi-node domain connects at startup to the JMS server. If a security provider that uses the RDBMS security store makes a change to its security data, all WebLogic Server instances are notified via JMS, and the local caches used by the WebLogic Security Service in each server instance are synchronized to that change.

If the JMS connection fails in a WebLogic Server instance that has been successfully started, the WebLogic Security Service associated with that server instance starts the JMS connection recovery process. The recovery process sleeps one second between reconnect attempts. The recovery process is stopped if the JMS connection failure persists after the number of reconnect attempts with which the

`JMSExceptionReconnectAttempts` property has been configured is reached. No further reconnect attempts are made: If a change is made to the security data in one WebLogic Server instance, the local caches managed by the WebLogic Security Service in other WebLogic Server instances are not synchronized to that change. However, if the JMS connection is successfully recovered by other means (such as a server reboot), those caches become synchronized.

If the JMS connection is not successfully started at the time a WebLogic Server instance is booted, a timer task that makes reconnect attempts is automatically started. The timer task is cancelled once the connection is successfully made. Two system properties may be configured for this timer task:

- `com.bea.common.security.jms.initialConnectionRecoverInterval`

  Specifies the delay, in milliseconds, before the connection recovery task is executed. The default value is 1000, which causes the connection recovery process to be executed after a delay of one second.

- `com.bea.common.security.jms.initialConnectionRecoverAttempts`

  Specifies the maximum number of reconnect attempts that can be made prior to cancelling the timer task. The default value is 3600, which causes the timer task to be cancelled once 3600 reconnect attempts have been made. No further reconnect attempts are made.

You can calculate the maximum connection polling duration by multiplying the values specified by each of the preceding system properties. For example, multiplying the default values of these two properties yields a maximum polling duration of one hour (1000 millisecond delay multiplied by 3600 reconnect attempts).

9.3 Upgrading a Domain to Use the RDBMS Security Store

To upgrade a domain to use the RDBMS security store, Oracle recommends creating a new domain in which the RDBMS security store is configured. After you create the new domain, you should export the security data from the security realm of the old domain, and import it into a security realm of the new domain. When you import security data into a security realm in a domain that uses the RDBMS security store, the data for the security providers that use the RDBMS security store is automatically loaded into that datastore. Data for security providers that do not use the RDBMS security store is automatically imported into the stores that those providers normally use by default.

It is possible to selectively migrate security providers individually from one security realm to another. However, when migrating security data to a domain that uses the RDBMS security store, Oracle recommends migrating the security realm's data in a single operation.

For information about migrating security realms, see the following topics:

- Section 8, "Migrating Security Data"
- "Export data from security realms" and "Import data into security realms" in the *Oracle WebLogic Server Administration Console Online Help*

10 Managing the Embedded LDAP Server

This chapter describes how to configure and manage the embedded LDAP server, which is included in WebLogic Server and that acts as the default security provider data store for the Default Authentication, Authorization, Credential Mapping, and Role Mapping providers.

This chapter includes the following sections:

10.1 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 domain has an embedded LDAP server configured with the default values set for each type of information. The Default Authentication, Authorization, Credential Mapping, and Role Mapping providers use the embedded LDAP server as their data store. If you use any of these providers in a new security realm, you may want to change the default values for the embedded LDAP server to optimize its use in your environment.

> **Note:** The performance of the embedded LDAP server is best with fewer than 10,000 users. If you have more users, consider using a different LDAP server and Authentication provider.

See "Configure the embedded LDAP server" in the *Oracle WebLogic Server Administration Console Online Help*.

The data file and change log file used by the embedded LDAP server can potentially grow quite large. You can configure maximum sizes for these files with the following `weblogic.Server` command line arguments:

- `-Dweblogic.security.ldap.maxSize=<max bytes>`, which limits the size of the data file used by the embedded LDAP server. When the data file exceeds the specified size, WebLogic Server eliminates from the data file space occupied by deleted entries.

- `-Dweblogic.security.ldap.changeLogThreshold=<number of entries>`, which limits the size of the change log file used by the embedded LDAP server. When the change log file exceeds the specified number of entries, WebLogic Server truncates the change log by removing all entries that have been sent to all Managed Servers.

10.2 Embedded LDAP Server Replication

The WebLogic Server embedded LDAP server for a domain consists of a master LDAP server, maintained in the domain's Administration Server, and a replicated LDAP server maintained in each Managed Server in the domain. When changes are made using a Managed Server, updates are sent to the embedded LDAP server on the Administration Server. The embedded LDAP server on the Administration Server maintains a log of all changes. The embedded LDAP server on the Administration Server also maintains a list of Managed Servers and the current change status for each one. The embedded LDAP server on the Administration Server sends appropriate changes to each Managed Server and updates the change status for each server. This process occurs when an update is made to the embedded LDAP server on the Administration 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.

You can configure the behavior of the embedded LDAP server on the Administration Server and the Managed Servers in a domain using the Administration Console. By selecting the **Domain** > **Security** > **Embedded LDAP** page in the Administration Console, you can set these attributes:

- **Refresh Replica At Startup** — Specifies whether the embedded LDAP server in a Managed Server should refresh all replicated data at boot time. This setting is useful if you have made many changes when 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.
- **Master First** — Specifies whether a Managed Server should always connect to the embedded LDAP server on the Administration Server, instead of connecting to the local replicated LDAP server.

See "Configure the embedded LDAP server" in the *Oracle WebLogic Server Administration Console Online Help*.

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

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

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

1. Download and install an external LDAP browser. You can find one LDAP browser at the following location:

   `http://www.openldap.org/`

   In this procedure it is assumed that you are using this LDAP browser; other LDAP browsers may differ in detail.

2. In the WebLogic Server Administration Console, change the credential for the embedded LDAP server:
   a. Expand **Domain** > **Security** > **Embedded LDAP**.
   b. In the **Credential** field, enter the new credential.
   c. In the **Confirm Credential** field, enter the new credential again.
   d. Click **Save**.
   e. Reboot WebLogic Server.

   Caution: Changing the credential can affect the operation of the domain. Do not perform this step on a production server.

3. Start the LDAP browser. To start the LDAP Browser/Editor mentioned in step 1, use the following command:

   ```
   lbe.sh
   ```

4. In the LDAP browser, configure a new connection in the LDAP browser:
   a. Select the QuickConnect tab.
   b. Set the host field to *localhost*.
   c. Set the port field to 7001 (7002 if SSL is being used).
   d. Set the Base DN field to dc=*mydomain* where *mydomain* is the name of the WebLogic domain you are using.
   e. Uncheck the Anonymous Bind option.
   f. Set the User DN field to cn=Admin.
   g. Set the Password field to the credential you specified in Step 2.
5. Click the new connection.

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

   Note: You can also view the contents of the embedded LDAP server by exporting its data and reviewing the exported file. See Section 10.4, "Exporting and Importing Information in the Embedded LDAP Server".

10.4 Exporting and Importing Information in the Embedded LDAP Server

You can export and import data from the embedded LDAP server using either the WebLogic Server Administration Console or an LDAP browser. To export and import data with the Console, use the Migration page of each security provider. See "Export data from a security provider" and "Import data into a security provider" in the *Oracle WebLogic Server Administration Console Online Help*.

Caution: When you use the Administration Console Migration tab to export security data, the export process deletes any existing files in the target directory with the .dat extension. Always export security data to an empty directory.

This section describes how to use an LDAP browser to export and import data stored in the embedded LDAP server. Table 10–1 summarizes where data is stored in the hierarchy of the embedded LDAP server.

Table 10–1 Location of Security Data in the Embedded LDAP Server

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

To export security data from the embedded LDAP server using the LDAP Browser/Editor:

1. Enter the following command at a command prompt to start the LDAP Browser/Editor:

   ```
   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 into the embedded LDAP server using the LDAP Browser/Editor:

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. In the LDAP Browser/Editor, select the LDIF > Import option.
4. Select Update/Add.
5. Specify the name of the file from which the data will be imported.

10.5 LDAP Access Control Syntax

The embedded LDAP server supports the IETF LDAP Access Control Model for LDAPv3. This section describes how that access control is implemented within the embedded LDAP server. You can apply these rules directly to entries within the directory as intended by the standard or you can configure and maintain them by editing the access control file (`acls.prop`).

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

10.5.1 The Access Control File

The access control file (`acls.prop`) maintained by the embedded LDAP server contains the complete list of access control lists (ACLs) for an entire LDAP directory. Each line in the access control file contains a single access control rule. An access control rule is made up of the following components:

- Location in the LDAP directory where the rule applies. See Section 10.5.2, "Access Control Location".
- Scope within that location to which the rule applies. See Section 10.5.3, "Access Control Scope".
- Access rights (either grant or deny). See Section 10.5.4, "Access Rights".
- Permissions (either grant or deny). See Section 10.5.4.1, "Attribute Permissions" and Section 10.5.4.2, "Entry Permissions".
- Attributes to which the rule applies. See Section 10.5.5, "Attributes Types".
- Subject being granted or denied access. See Section 10.5.6, "Subject Types".

Example 10–1 shows a sample access control file.

Example 10–1 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:
```

10.5.2 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.

10.5.3 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.

10.5.4 Access Rights

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

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

10.5.4.1 Attribute Permissions

The permissions shown in Table 10–2 apply to actions involving attributes.

Table 10–2 Attribute Permissions

Permission	Description
`r` Read	Read attributes. If granted, permits attributes and values to be returned in a Read or Search operation.
`w` Write	Modify or add attributes. If granted, permits attributes and values to be added in a Modify operation.
`o` Obliterate	Modify and delete attributes. If granted, permits attributes and values to be deleted in a Modify operation.
`s` Search	Search entries with specified attributes. If granted, permits attributes and values to be included in a Search operation.
`c` Compare	Compare attribute values. If granted, permits attributes and values to be included in a Compare operation.
`m` Make	Make attributes on a new LDAP entry below this entry.

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

10.5.4.2 Entry Permissions

The permissions shown in Table 10–3 apply to entire LDAP entries.

Table 10–3 Entry Permissions

Permission	Description
a Add	Add an entry below this LDAP entry. If granted, permits creation of an entry in the DIT subject to control on all attributes and values placed on the new entry at the time of creation. In order to add an entry, permission must also be granted to add at least the mandatory attributes.
d Delete	Delete this entry. If granted, permits the entry to be removed from the DIT regardless of controls on attributes within the entry.
e Export	Export entry and all subentries to new location. If granted, permits an entry and its subentries (if any) to be exported; that is, removed from the current location and placed in a new location subject to the granting of suitable permission at the destination. If the last RDN is changed, Rename permission is also required at the current location. In order to export an entry or its subentries, there are no prerequisite permissions to the contained attributes, including the RDN attribute. This is true even when the operation causes new attribute values to be added or removed as the result of the changes to the RDN.
i Import	Import entry and subentries from specified location. If granted, permits an entry and its subentries (if any) to be imported; that is, removed from one location and placed at the specified location (if suitable permissions for the new location are granted). When you import an entry or its subentries, the contained attributes, including the RDN attributes, have no prerequisite permissions. This is true even when the operation causes new attribute values to be added or removed as the result of the changes to RDN.
n RenameDN	Change the DN of an LDAP entry. Granting the Rename permission is necessary for an entry to be renamed with a new RDN, taking into account consequential changes to the DN of subentries. If the name of the superior entry is unchanged, the grant is sufficient. When you rename an entry, 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.
b BrowseDN	Browse the DN of an entry. If granted, this permission permits entries to be accessed using directory operations that do not explicitly provide the name of the entry.
t ReturnDN	Allows DN of entry to be disclosed in an operation result. If granted, this permission allows the distinguished name of the entry to be disclosed in the operation result.

10.5.5 Attributes Types

The attribute types to which an access control rule applies should be listed in the ACL 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.

10.5.6 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, and 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.

10.5.7 Grant/Deny Evaluation Rules

The decision whether to grant or deny a client access to the 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, these guiding principles apply:

- More specific rules override less specific ones (for example, individual user entries in an ACL take precedence over a group entry).
- If a conflict still exists in spite of the specificity of the rule, the subject of the rule determines which rule will be applied. Rules based on an `IP Address` subject are given the highest precedence, followed by rules that are applied to a specific `AuthzID` or `This` subject. Next in priority are rules that apply to `Group` subjects. Last priority is given to rules that apply to `Subtree` and `Public` subjects.
- When there are conflicting ACL values, Deny takes precedence over Grant.
- Deny is the default when there is no access control information. Additionally, an entry scope takes precedence over a subtree scope.

10.6 Backup and Recovery

If any of your security realms use the Default Authentication, Authorization, Credential Mapping, or Role Mapping providers, you should maintain an up-to-date backup of the following directory tree:

domain_name/servers/*adminServer*/data/ldap

In the preceding directory, *domain_name* is the domain root directory and *adminServer* is the directory in which the Administration Server stores run-time and security data. For more information backing up the embedded LDAP server data, see the following topics:

- "Back Up LDAP Repository" in *Administering Server Startup and Shutdown for Oracle WebLogic Server*
- "Configure backups for embedded LDAP servers" in *Oracle WebLogic Server Administration Console Online Help*

If the embedded LDAP server file becomes corrupt or unusable, the Administration Server will generate a `NumberFormatException` and fail to start. This situation is rare but can occur if the disk becomes full and causes the embedded LDAP file to enter into an invalid state.

To recover from an unusable embedded LDAP server file, complete the following steps:

1. Change to the following directory:

   domain_name`/servers/`*adminServer*`/data`

2. Rename the embedded LDAP server file, as in the following example:

   ```
   mv ldap ldap.old
   ```

   By renaming the file, and not deleting it completely, it remains available to you for analysis and potential data recovery.

3. Start the Administration Server.

   When the Administration Server starts, a new embedded LDAP server file is created.

4. Restore any data to the new embedded LDAP server that was added since the time the WebLogic domain was created.

   If you have configured a backup of the embedded LDAP server, you can restore the backed up data by importing it. For information, see Section 10.4, "Exporting and Importing Information in the Embedded LDAP Server".

11 Configuring Identity and Trust

This chapter describes how to configure identity and trust for WebLogic Server.

This chapter includes the following sections:

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

11.1 Private Keys, Digital Certificates, and Trusted Certificate Authorities

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

SSL uses public key encryption technology for authentication. With public key encryption, a public key and a *private key* are generated for a server. Data encrypted with the public key can only be decrypted using the corresponding private key and data encrypted with the private key can only be decrypted using the corresponding public key. 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 in 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 (CA) and digitally signed with the CA's digital certificate. Well-known certificate authorities include Entrust and Symantec Corporation. The trusted CA certificate establishes *trust* for a certificate.

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

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

- The demonstration digital certificates, private keys, and trusted CA certificates in the *DOMAIN_HOME*\security, *WL_HOME*\server\lib, and *JAVA_HOME*\jre\lib\security directories.

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

- The private key and self-signed digital certificate for WebLogic Server that are created by the keytool utility. Note the following:
  - After you create the private key and self-signed certificate, use keytool to generate a Certificate Signing Request (CSR). Submit the CSR to a CA to obtain a digital certificate for WebLogic Server.
  - Use the keytool utility to update the self-signed digital certificate with a new digital certificate.
  - You can also use the keytool utility to obtain trust and identity when using WebLogic Server in a production environment.

  For more information, see Section 11.4.1.1, "Using the Keytool Utility".
- The digital certificates and private keys generated by the CertGen utility. However, these should be used only for demonstration or testing purposes in a development environment, not in a production environment. Use the CertGen utility if you want to set an expiration date in the digital certificate or specify a correct host name in the digital certificate so that you can use host name verification. (The demonstration digital certificate provided by WebLogic Server uses the machine's default host name as the host name.) For more information about using the CertGen utility to obtain private keys and digital certificates, see Section 11.4.1.2, "Using the CertGen Utility".

  Note: The Certificate Request Generator servlet is deprecated. Use the keytool utility instead, described in Section 11.4.1.1, "Using the Keytool Utility".

11.1.1 Supported Formats for Identity and Trust

The PEM (Privacy Enhanced Mail) format is the preferred format for private keys, digital certificates, and trusted certificate authority (CA) certificates. The preferred keystore format is JKS (Java KeyStore).

A .pem format file begins with this line:

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

and ends with this line:

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

A .pem format file supports multiple digital certificates (for example, a certificate chain can be included). The order of certificates within the file is important. The server's

digital certificate should be the first digital certificate in the file, followed by the issuer certificate, and so on. Each certificate in the chain is followed by its issuer certificate. If the last certificate in the chain is the self-signed (self-issued) root certificate of the chain, the chain is considered complete. Note that the chain does not have to be complete.

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

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

Microsoft is often used as a CA. Microsoft issues trusted CA certificates in p7b format, which must be converted to PEM before they can be used with WebLogic Server. For more information, see Section 11.4.1.4, "Converting a Microsoft p7b Format to PEM Format".

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

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

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

Note: OpenSSL can add a header to the PEM certificate it generates. In order to use such certificates with WebLogic Server, everything in front of "`-----BEGIN CERTIFICATE-----`" should be removed from the certificate, which you can do with a text editor.

11.2 Identity and Trust Keystores

When you configure SSL, you must decide how identity and trust will be stored. Although one keystore can be used for both identity and trust, Oracle recommends using separate keystores for both identity and trust because the identity keystore (private key/digital certificate pairs) and the trust keystore (trusted CA certificates) may have different security requirements. For example:

- For trust, you only need the certificates (non-sensitive data) in the keystore. However, for identity, you add the certificate and the private key (sensitive data) in the keystore.
- The identity keystore may be prohibited by company policy from ever being put on the corporate 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 identity keystore password is generally known to fewer people than the password for the trust keystore.

In general, systems within a domain have the same trust rules — they use the same set of trusted CAs — but they tend to have per-server identity. Identity requires a private key, and private keys should not be copied from one system to another. Therefore, you should maintain separate identity keystores for each system, each keystore containing only the server identity needed for that system. However, trust keystores can be copied from system to system, thus making it easier to standardize trust conventions.

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

11.3 How WebLogic Server Locates Trust

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

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

11.4 Configuring Identity and Trust: Main Steps

To create identity and trust for a server:

1. Obtain digital certificates, private keys, and trusted CA certificates. See Section 11.4.1, "Obtaining Private Keys, Digital Certificates, and Trusted Certificate Authority Certificates".
2. Create the identity and trust keystores. See Section 11.4.2.3, "Tools and Utilities for Creating Keystores and Loading Private Keys and Certificates".
3. Store the private keys, digital certificates, and trusted CA certificates in the keystores. See Section 11.4.2, "Storing Private Keys, Digital Certificates, and Trusted Certificate Authority Certificates".

   Note: The preferred keystore format is JKS (Java KeyStore). WebLogic Server supports private keys and trusted CA certificates stored in files or in the WebLogic Keystore provider for the purpose of backward compatibility only.

4. Configure the keystores for WebLogic Server. See Section 11.4.3, "Configuring Identity and Trust Keystores for WebLogic Server".

11.4.1 Obtaining Private Keys, Digital Certificates, and Trusted Certificate Authority Certificates

You have multiple choices for obtaining private keys, digital certificates, and trusted CA certificates for your WebLogic Server environment. When choosing, note the following considerations:

- You can obtain keys, certificates, and trusted CA artifacts from the CertGen utility, the keytool utility, or a reputable vendor such as Entrust or Symantec Corporation. However, for a production environment, you should obtain private keys and digital certificates only from a reputable certificate authority such as Entrust or Symantec Corporation.
- You can also use the digital certificates, private keys, and trusted CA certificates provided by WebLogic Server. However, these should be used in a development environment only.

This section describes the tools and methods for obtaining keys, certificates, and trusted CA certificates. The following topics are included:

- Section 11.4.1.1, "Using the Keytool Utility"
- Section 11.4.1.2, "Using the CertGen Utility"
- Section 11.4.1.3, "Using Your Own Certificate Authority"
- Section 11.4.1.4, "Converting a Microsoft p7b Format to PEM Format"
- Section 11.4.1.5, "Obtaining a Digital Certificate for a Web Browser"
- Section 11.4.1.6, "Using Certificate Chains (Deprecated)"

11.4.1.1 Using the Keytool Utility

Keytool is a key and certificate management utility. It allows users to administer their own public/private key pairs and associated certificates for use in self-authentication (where the user authenticates himself/herself to other users/services) or data integrity and authentication services, using digital signatures. It also allows users to cache the public keys (in the form of certificates) of their communicating peers.

For an example of using keytool to obtain key pairs, a self-signed server certificate, and a trusted CA certificate, see Section 11.5, "Creating a Keystore: An Example". For complete details about keytool, see "keytool — Key and Certificate Management Tool" at
http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html

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

Table 11–1 lists keytool commands commonly used for creating and using JKS keystores with WebLogic Server. In this table, brackets surrounding an option signify that the user is prompted for the value(s) if the option is not specified on the command line.

Caution: Although the `keytool` command includes options for specifying passwords, which are described in Table 11–1, you should never include unencrypted passwords in command lines. Instead, you should allow `keytool` to prompt you for any required passwords after you enter the command, as in the following example. User input is shown in **bold**.

```
C:\DOMAIN_NAME>keytool -genkeypair -keystore MyKeyStore
Enter keystore password:
Re-enter new password:
```

Unlike passwords that are specified in command-line options, passwords entered in response to prompts are not echoed in the command window, and are not captured in logs, which allows for secure password input.

Table 11–1 Commonly Used keytool Commands

Command	Description
`keytool -genkeypair -keystore` *keystorename* `-storepass` *keystorepassword*	Generates a key pair (a public key and associated private key) and self-signed digital certificate in a keystore. If the keystore does not exist, it is created.
`keytool -importcert -alias` *aliasforprivatekey* `-file` *privatekeyfilename*`.pem -keyfilepass` *privatekeypassword* `-keystore` *keystorename* `-storepass` *keystorepassword*	Updates the self-signed digital certificate with one signed by a trusted CA.
`keytool -importcert -alias` *rootCA* `-trustcacerts -file` *RootCA*`.pem -keystore trust.jks -storepass` *keystorepassword* `keytool -importcert -alias` *intermediate* `-trustcacerts -file` *Intermediate*`.pem -keystore` *keystorename* `-storepass` *keystorepassword*	Creates a custom keystore to be used for holding an intermediate CA certificate. ■ The first keytool command creates the keystore, `trust.jks`, which holds the root CA certificate. ■ The second keytool command imports the intermediate CA certificate into `trust.jks`. This enables WebLogic Server's SSL implementation to transmit the intermediate certificate with the server's public certificate to the client during the SSL handshake.
`keytool -importcert -alias` *aliasfortrustedca* `-trustcacerts -file` *trustedcafilename*`.pem -keystore` *keystorename* `-storepass` *keystorepassword*	Loads a trusted CA certificate into a keystore. If the keystore does not exist, it is created.
`keytool -certreq -alias` *alias* `-sigalg` *sigalg* `-file` *certreq_file* `-keyfilepass` *privatekeypassword* `-storetype` *keystoretype* `-keystore` *keystorename* `-storepass` *keystorepassword*	Generates a Certificate Signing Request (CSR), using the PKCS#10 format, and a self-signed certificate with a private key. Stores the CSR in the specified *certreq_file*, and the certificate/private key pair as a key entry in the specified keystore under the specified alias.
`keytool -list -keystore` *keystorename*	Displays the contents of the keystore.
`keytool -delete -keystore` *keystorename* `-storepass` *keystorepassword* `-alias` *privatekeyalias*	Deletes the entry identified by the specified alias from the keystore.
`keytool -help`	Provides online help for keytool.

11.4.1.2 Using the CertGen Utility

> **Note:** The CertGen utility generates digital certificates and private keys that should only be used for demonstration or testing purposes, not in a production environment. For important information about limitations on its use, see Section 11.4.1.2.2, "Limitation on CertGen Usage".

The CertGen utility provides command line options to specify a CA certificate and key to be used for issuing generated certificates. The digital certificates generated by the CertGen utility by default have only the host name of the machine on which they were generated, and not the fully-qualified DNS name, as the value for its common name field (cn). Command line options let you specify values for the cn and other Subject domain name (DN) fields, such as `orgunit`, `organization`, `locality`, `state`, and `countrycode`.

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

By default, the CertGen utility uses the following demonstration digital certificate and private-key files: `CertGenCA.der` and `CertGenCAKey.der`. CertGen looks for these files in the current directory, or in the *`WL_HOME`*`/server/lib directory`, as specified in the `weblogic.home` system property or the `CLASSPATH`. If you want to use these files, you need not specify CA files on the command line. Alternatively, you can specify CA files on the command line.

11.4.1.2.1 Command Syntax and Examples For information about the CertGen utility's syntax and arguments, see "CertGen" in the *Command Reference for Oracle WebLogic Server*.

For an example that generates a certificate and private key using the CertGen utility, and then creates a keystore and stores a private key using the ImportPrivateKey utility, see "ImportPrivateKey" in the *Command Reference for Oracle WebLogic Server*.

> **Note:** If you do not explicitly specify a hostname with the -cn option, CertGen uses the JDK `InetAddress.getHostname()` method to get the hostname that 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). On Solaris, the result of `InetAddress.getHostname()` depends on how the hosts entry is configured in the `/etc/nsswitch.conf` file.
>
> If WebLogic Server is acting as a client (and by default host name verification is enabled), you need to ensure that the host name specified in the URL matches the Subject common name in the server certificate. Otherwise, connections will fail because the host names do not match.

11.4.1.2.2 Limitation on CertGen Usage By default, a WebLogic Server domain is configured with the `DemoIdentity.jks` keystore, which contains a demonstration public certificate and private key for WebLogic Server. This certificate and key are

created by CertGen with the default options of containing only the host name in the common name field (cn), and not the fully-qualified DNS name. As a result, attempts to establish SSL connections may fail in some situations due to a host name verification exception. This section describes this limitation and provides some workarounds.

If you are using the demo certificates in a multi-server domain, Managed Server instances will fail to boot if they cannot establish an SSL connection with the Administration Server. An error message similar to the following may be generated:

```
BAD_CERTIFICATE alert was
received from node-name.avitek.com - xxx.yy.zzz.yyy. Check the peer to
determine why it rejected the certificate chain (trusted CA configuration,
hostname verification). SSL debug tracing may be required to determine the
exact reason the certificate was rejected.
```

This error occurs because the host name verifier, which is enabled by default in all WebLogic domains and which is used during the SSL handshake, compares the value of the cn field in the certificate with the fully-qualified DNS name of the SSL server that accepts the SSL connection. If these names do not match, the SSL connection is dropped.

If you use the demo identity certificates in a WebLogic domain, you can use the following workarounds:

- Specify the SSL listen address of each WebLogic Server instance in a domain as the host name that appears in the certificate's cn field. Avoid using the fully-qualified DNS name or IP address. This workaround consists of two steps:

  1. When using the Configuration Wizard to create the WebLogic domain, specify the listen address of each WebLogic Server instance as a simple host name as it appears in the certificate's cn field, not as a fully-qualified DNS name or IP address. For example, if the host name in the certificate is `avitek01`, the listen address for the server instance should be specified simply as `avitek01`.
  2. At run time, when specifying the SSL listen address of a server instance, make sure the URL also matches the host name for that server as specified as the certificate's cn field. For example:

     ```
     https://avitek01:7002
     ```

- To start a Managed Server instance, pass the URL of the Administration Server's SSL listening address as a parameter to the `startManagedWebLogic` script. The URL should be specified in a form that excludes the domain suffix. For example:

  ```
  C:\mydomain\bin> startManagedWebLogic.cmd https://admin01:7002
  ```

- Disable host name verification. This causes WebLogic Server to skip the verification check of ensuring that the host name in the URL to which a connection is made matches the host name in the digital certificate that the server sends back as part of the SSL connection.

  You can disable host name verification by including a command similar to the following in the setDomainEnv script:

  ```
  set JAVA_OPTIONS=%JAVA_OPTIONS%
  -Dweblogic.security.SSL.ignoreHostnameVerification=true
  ```

  For information about configuring host name verification, see Section 12.5, "Using Host Name Verification".

Note: Oracle does not recommend using the demo certificates or turning off host name verification in a production environment.

11.4.1.3 Using Your Own Certificate Authority

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

1. Ensure the trusted CA certificates are in PEM format.
   - If the trusted CA certificate is in DER format, use the der2pem utility to convert them.
   - If the trusted CA certificate was issued by Microsoft, see Section 11.4.1.4, "Converting a Microsoft p7b Format to PEM Format".
   - If the trusted CA certificate has a custom file type, use the steps in Section 11.4.1.4, "Converting a Microsoft p7b Format to PEM Format" to convert the trusted CA certificate to PEM format.
2. Create a trust keystore and store the trusted CA certificate in it. For more information, see Section 11.4.2, "Storing Private Keys, Digital Certificates, and Trusted Certificate Authority Certificates".
3. Configure WebLogic Server to use the trust keystore. For more information, see Section 11.4.3.1, "Configuring Keystores for Production".

11.4.1.4 Converting a Microsoft p7b Format to PEM Format

Digital certificates issued by Microsoft are in a format (p7b) that cannot be used by WebLogic Server. The following example converts a digital certificate in p7b (PKCS#7) format to PEM format on Windows XP:

1. In Windows Explorer, select the file (*filename*.p7b) you want to convert. Double-click on the file to display a Certificates window.
2. In the left pane of the Certificates window, expand the file.
3. Expand the Certificates folder to display a list of certificates.
4. Select a certificate to convert to PEM format. Right-click on the certificate, then choose **All Tasks** > **Export** to display the Certificate Export Wizard.
5. In the wizard, click **Next**.
6. Select the `Base-64 encoded X.509 (.CER)` option. Then click **Next**. (`Base-64 encoded` is the PEM format.)
7. In the **File name** field, enter a name for the converted digital certificate; then click **Next**.

   Note: The wizard appends a `.cer` extension to the output file. The `.cer` extension is a generic extension which is appended to both base-64 encoded certificates and DER certificates. You can change the extension to `.pem` after you exit the wizard.

8. Verify that the settings are correct. If the settings are correct, click Finish; if they are not correct, click Back and make any necessary modifications.

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.

11.4.1.5 Obtaining a Digital Certificate for a Web Browser

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

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

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

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

11.4.1.6 Using Certificate Chains (Deprecated)

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

To use certificate chains with WebLogic Server:

1. Ensure that all the digital certificates are in PEM format. If they are in DER format, you can convert them using the "der2pem" utility. If you are using a digital certificate issued by Microsoft, see Section 11.4.1.4, "Converting a Microsoft p7b Format to PEM Format". You can use the steps in the section to convert other types of digital certificates. Save the digital certificate in Base 64 format.
2. Open a text editor and include all the digital certificate files into a single file. The order is important. The server digital certificate should be the first digital certificate in the file. The issuer of that digital certificate should be the next in the file and so on until you get to the self-signed root certificate authority (CA) certificate. This digital certificate should be the last certificate in the file.

   You cannot have blank lines between digital certificates.
3. Specify the file in the **Server Certificate File Name** field on the **Configuration** > **SSL** page in the WebLogic Server Administration Console.

Example 11–1 shows a sample certificate chain.

Example 11–1 Sample File with Certificate Chain

```
-----BEGIN CERTIFICATE-----
MIICyzCCAjSgAwIBAgIBLDANBgkqhkiG9w0BAQQFADCBtjELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExFjAUB
gNVBAcTDVNhbiBGcmFuY2lzY28xFTATBgNVBAoTDEJFQSBXZWJMb2dpYzERMA8GA1UECxMIU2VjdXJpdHkxLzAtBgNVBAMTJk
RlbW8gQ2VydGlmaWNhdGUgQXV0aG9yaXR5IENvbnN0cmFpbnRzMR8wHQYJKoZIhvcNAQkBFhBzZWN1cml0eUBiZWEuY29tMB4
XDTAyMTEwMTIwMDIxMloXDTA2MTAxNTIwMDIxMlowgZ8xCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRYwFAYD
VQQHEw1TYW4gRnJhbmNpc2NvMRUwEwYDVQQKEwxCRUEgV2ViTG9naWMxETAPBgNVBAsTCFNlY3VyaXR5MRkwFwYDVQQDExB3Z
WJsb2dpYy5iZWEuY29tMR4wHAYJKoZIhvcNAQkBFg9zdXBwb3J0QGJlYS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAo
GBAMJX8nKUgsFej8pEu/1IVcHUkwY0c2JbBzOryu3sce4QjX+rGxiCjoPm2MY=yts2BvonuJ6CztdZf8B/LBEWCz+qRrtdFn9
mKSZWGvrAkmMPz2RhXEOThpoRo5kZz2FQ9XF/PxIJXTYCM7yooRBwXoKYjquRwiZNtUiU9kYi6Z3prAgMBAAEwDQYJKoZIhvc
NAQEEBQADgYEAh2eqQGxEMUnNTwEUD

0tBq+7YuAkjecEocGXvi2G4YSoWVLgnVzJoJuds3c35KE6sxBe1luJQuQkE9SzALG/6lDIJ5ctPsHFmZzZxY7scLl6hWj5ON8
oN2YTh5Jo/ryqjvnZvqiNIWe/gqr2GLIkajC0mz4un1LiYORPig3fBMH0=

-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----

MIIC+jCCAmOgAwIBAgIBADANBgkqhkiG9w0BAQQFADCBtjELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExFjAUB
gNVBAcTDVNhbiBGcmFuY2lzY28xFTATBgNVBAoTDEJFQSBXZWJMb2dpYzERMA8GA1UECxMIU2VjdXJpdHkxLzAtBgNVBAMTJk
RlbW8gQ2VydGlmaWNhdGUgQXV0aG9yaXR5IENvbnN0cmFpbnRzMR8wHQYJKoZIhvcNAQkBFhBzZWN1cml0eUBiZWEuY29tMB4
XDTAyMTEwMTIwMDIxMVoXDTA2MTAxNjIwMDIxMVowgbYxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRYwFAYD
VQQHEw1TYW4gRnJhbmNpc2NvMRUwEwYDVQQKEwxCRUEgV2ViTG9naWMxETAPBgNVBAsTCFNlY3VyaXR5MS8wLQYDVQQDEyZEZ
W1vIENlcnRpZmljYXRlIEF1dGhvcml0eSBDb25zdHJhaW50czEfMB0GCSqGSIb3DQEJARYQc2VjdXJpdHlAYmVhLmNvbTCBnz
ANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA3ynD8l5JfLob4g6d94dNtI0Eep6QNl9bblmswnrjIYz1BVjjRjNVal9fRs+8jvm
85kIWlerKzIMJgiNsj50WlXzNX6orszggSsW15pqV0aYE9Re9K

CNNnORlsLjmRhuVxg9rJFEtjHMjrSYr2IDFhcdwPgIt0meWEVnKNObSFYcCAwEAAaMWMBQwEgYDVR0TAQH/BAgwBgEB/wIBAT
ANBgkqhkiG9w0BAQQFAAOBgQBS+0oqWxGyqbZO028zf9tQT2RKojfuwywrDoGW96Un5IqpFnBHIu5atliJo3OUpiH18KkwLN8
DVP/3t3K3O3kXdIuLbqAL0i5xyBlAhr7gE5eVhIyeMg7ETBPLyGO2BF13Y24LlsO+MX9jW7fxMraPN608QeJXkZw0E0cGwrw2AQ
==

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

11.4.2 Storing Private Keys, Digital Certificates, and Trusted Certificate Authority Certificates

Once you have obtained private keys, digital certificates, and trusted CA certificates, you need to store them so that WebLogic Server can use them to find and verify identity. Private keys, their associated digital certificates, and trusted CA certificates are stored in keystores. Then you need to configure those keystores with WebLogic Server.

This section contains the following topics:

- Section 11.4.2.1, "Using the Demonstration Keystores"
- Section 11.4.2.3, "Tools and Utilities for Creating Keystores and Loading Private Keys and Certificates"
- Section 11.4.2.4, "Configuring Demo Certificates for Clients"

For a step-by-step example of using the keytool utility to create a keystore and store keys and certificates in it, see Section 11.5, "Creating a Keystore: An Example".

11.4.2.1 Using the Demonstration Keystores

By default, WebLogic Server is configured with two keystores, which are located in the *DOMAIN_HOME*\security and *WL_HOME*\server\lib directories, respectively:

- `DemoIdentity.jks`—Contains a demonstration private key for WebLogic Server. This keystore contains the identity for WebLogic Server.

  Note: As of version 12.1.2 of WebLogic Server, the `DemoIdentity.jks` keystore is generated at domain creation and is located in the *`DOMAIN_HOME`*`\security` directory. The demo CA certificate has a 2048-bit key size, uses the SHA256 message digest algorithm, and has a Key Identifier extension.

- `DemoTrust.jks`—Contains the trusted certificate authorities from the *`WL_HOME`*`\server\lib\DemoTrust.jks` and the JDK `cacerts` keystores. This keystore establishes trust for WebLogic Server.

For testing and development purposes, the keystore configuration is complete. However, do not use the demonstration keystores in a production environment. Because the digital certificates and trusted CA certificates in the demonstration keystores are signed by a WebLogic Server demonstration certificate authority, a WebLogic Server installation using the demonstration keystores will trust *any* WebLogic Server installation that also uses the demonstration keystores. You want to create a secure environment where only your installations trust each other.

11.4.2.2 Interoperating With Keystores From Prior Versions

If you are using WebLogic Server version 12.1.2 together with a previous version of WebLogic Server, be aware that the demo trust keystore of the previous versions does not contain the demo CA certificate used by version 12.1.2. Therefore, if a 12.1.2 instance of WebLogic Server sends its public certificate to an instance of WebLogic Server running a prior version, that public certificate will not automatically be trusted.

For interoperability with prior releases, you can use either of the following methods:

- Use the system property `-Dsecurity.use.interopCA=true` to generate interoperable demo certificates signed by the previous demo CA certificate.
- On the 12.1.2 instance of WebLogic Server, use the CertGen utility with the -cacert -cakey arguments to generate demo certificates signed by the previous demo CA certificate. Then, use ImportPrivateKey to import them into `DemoIdentity.jks`, as shown in the following example:

```
java utils.CertGen
    -certfile <cert_file>
    -keyfile <private_key_file>
    -keyfilepass DemoIdentityPassPhrase
    -cacert $WL_HOME/server/lib/CertGenInteropCA.der
    -cakey $WL_HOME/server/lib/CertGenInteropCAKey.der
    -cakeypass password


java utils.ImportPrivateKey
    -certfile <cert_file>
    -keyfile <private_key_file>
    -keyfilepass DemoIdentityPassPhrase
    -keystore DemoIdentity.jks
    -storepass DemoIdentityKeyStorePassPhrase
    -alias DemoIdentity
    -keypass DemoIdentityPassPhrase
```

11.4.2.3 Tools and Utilities for Creating Keystores and Loading Private Keys and Certificates

A keystore is for the secure storage and management of private keys/digital certificate pairs and trusted CA certificates. Use the following mechanisms to create a keystore and load private keys and trusted CA certificates into the keystore:

- The WebLogic ImportPrivateKey utility. The ImportPrivateKey utility allows you to take private key and digital certificate files and load them into a keystore. For more information, see "ImportPrivateKey" in the *Command Reference for Oracle WebLogic Server*.
- The keytool utility. Use the keytool utility to generate a key pair (a public key and associated private key) and a self-signed digital certificate and store them in the keystore. While you can use the keytool utility to generate new key pairs and digital certificates and add them to a keystore, the utility does not allow you to take an existing private key from a file and import it into the keystore. Instead, use the WebLogic ImportPrivateKey utility.

  Note: The keytool utility does allow you to import trusted CA certificates from a file into a keystore.

  For step-by-step instructions explaining how to use keytool to create a keystore, see Section 11.5, "Creating a Keystore: An Example".
- Custom utilities. WebLogic Server can use keystores created with custom tools or utilities. How to create and use these utilities is outside the scope of this document.

All private key entries in a keystore are accessed by WebLogic Server via unique aliases. You specify the alias when loading the private key into the keystore. Aliases are case-insensitive: the aliases Hugo and hugo would refer to the same keystore entry. Aliases for private keys are specified in the Private Key Alias field on the **Configuration** > **SSL** page in the WebLogic Server Administration Console. 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.

All certificate authorities in a keystore identified as trusted by WebLogic Server are trusted.

11.4.2.4 Configuring Demo Certificates for Clients

To use SSL in development mode between a client such as Eclipse and WebLogic Server, configure the demo certificates in the JVM for both the client and the server:

1. Copy *ORACLE_HOME*`/wlserver/server/lib/cacerts` to the `jre/lib/security` directory of the client's JVM. For example, if you are using Eclipse with its default JDK, copy `cacerts` to *ORACLE_HOME*`/jdk/jre/lib/security`.
2. Copy *ORACLE_HOME*`/wlserver/server/lib/cacerts` to the `jre/lib/security` directory of the WebLogic Server's JVM.
3. Restart both WebLogic Server and the client.

As an alternative, you can import the certificates, rather than copying the `cacerts` files.

11.4.3 Configuring Identity and Trust Keystores for WebLogic Server

This section includes the following topics:

- Section 11.4.3.1, "Configuring Keystores for Production"
- Section 11.4.3.2, "Configuring Keys and Certificates Stored in a File or JKS Keystore Accessed by the WebLogic Keystore Provider"

11.4.3.1 Configuring Keystores for Production

After you have created the identity and trust keystores that you intend to use in a production environment, you must configure them with WebLogic Server. You can use any of the following methods to do this:

- The **Configuration** > **Keystores** page of the WebLogic Server Administration Console. For information, see "Configure keystores" in the *Oracle WebLogic Server Administration Console Online Help*.
- A WLST script. See *Understanding the WebLogic Scripting Tool*.
- The Java Management Extensions (JMX) APIs for creating a new security configuration. See *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*.

11.4.3.2 Configuring Keys and Certificates Stored in a File or JKS Keystore Accessed by the WebLogic Keystore Provider

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

11.5 Creating a Keystore: An Example

This section shows an example of using the keytool utility for creating a keystore and storing keys and certificates in it. Note that this section shows only how to create one keystore. In a production environment, Oracle recommends that you have two keystores: one for trust, and another for identity, as explained in Section 11.2, "Identity and Trust Keystores". For complete details about each of the keytool command options shown in this section, see "keytool — Key and Certificate Management Tool" at `http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html`.

To create a keystore and populate it with private keys and certificates, complete the following steps:

1. Create a directory to hold the keystore; for example: *`ORACLE_HOME`*`/keystores`.
2. Run the following script, which sets the domain-wide environment for starting and running WebLogic Server instances:

   `DOMAIN_HOME``/bin/setDomainEnv`

   In the preceding path, *`DOMAIN_HOME`* represents the WebLogic domain root directory.
3. Change to the directory to hold the keystore, which you created in Step 1.
4. Create the keystore using the following `keytool` command syntax. This command also creates a key pair (a public key and associated private key) and an alias for the private key.

   ```
   keytool -genkeypair -alias alias -keyalg RSA -keysize 1024 -dname dn -keystore
   ```

keystore

In the preceding command syntax:

- *alias* represents the private key alias.
- *dn* represents the X.500 Distinguished Name associated with the private key alias.
- *keystore* represents the name of the keystore being created.

For example:

```
prompt> keytool -genkeypair -alias server_cert -keyalg RSA -keysize 1024
-dname "CN=server.avitek.com,OU=Support,O=Avitek,L=Reading,ST=Berkshire,C=GB"
-keystore keystore.jks
```

Note the following in the preceding example:

- `server.avitek.com` represents the WebLogic Server host and DNS domain name.
- Although the `keytool` command includes the `-storepass` and `-keypass` options for specifying the keystore and private key passwords, respectively, Oracle recommends that you avoid using these command-line options. When you enter a `keytool` command that requires one or more passwords, but you omit the command-line options for passing them, you are subsequently prompted to enter them. However, unlike passwords passed in command-line options, passwords entered in response to a prompt are not displayed in the command window and are not captured in any log.
- Make note of the private key alias and passwords you specify, and be sure to record passwords only in a safe location.

5. Make a backup copy of the keystore created in Step 4.
6. To view the contents of a keystore, use the following `keytool` command syntax, where *keystore* represents the name of the keystore you created:

```
keytool -list -v -keystore keystore
```

When you enter the preceding command, you are prompted for the keystore password. For example, the following command lists the contents of `keystore.jks`:

```
prompt> keytool -list -v -keystore keystore.jks
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: server_cert
Creation date: Sep 13, 2010
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=server.avitek.com, OU=Support, O=Avitek, L=Reading, ST=Berkshire,
C=GB
Issuer: CN=server.avitek.com, OU=Support, O=Avitek, L=Reading, ST=Berkshire,
C=GB
Serial number: 4c8e1ad5
```

```
Valid from: Mon Sep 13 13:36:37 BST 2010 until: Sun Dec 12 12:36:37 GMT 2010
Certificate fingerprints:
MD5: 1A:4A:3B:42:7E:BD:94:65:67:0E:9B:02:28:90:D6:A8
SHA1: C1:53:48:50:EB:F1:FD:A0:DC:28:9F:EF:3B:C8:FB:22:82:9F:8E:EE
Signature algorithm name: SHA1with RSA
Version: 3


*******************************************
*******************************************
```

7. Create a Certificate Signing Request (CSR) using the following `keytool` command syntax:

   ```
   keytool -certreq -v -alias alias -file certreq_file -keystore keystore
   ```

   In the preceding command syntax:

   - *`alias`* represents the private key alias specified in Step 4.
   - *`certreq_file`* represents the name of the file that contains the CSR.
   - *`keystore`* represents the keystore created in Step 4.

   Note that when you create a CSR using the preceding command, you are prompted to enter the passwords for the keystore and the private key.

   For example, the following command creates a CSR in the file `server.csr`:

   ```
   prompt> keytool -certreq -v -alias server_cert -file server.csr -keystore
   keystore.jks
   ```

   The CSR file is encoded in PKCS#10 format and may look similar to the following:

   ```
   -----BEGIN NEW CERTIFICATE REQUEST-----
   MIIBtzCCASACAQAwdzELMAkGA1UEBhMCR0IxEjAQBgNVBAgTCUJlcmtzaGlyZTEQMA
   4GA1UEBxMHUmVhZGluZzEPMA0GA1UEChMGT3JhY2xlMRAwDgYDVQQLEwdTdXBwb3J0
   MR8wHQYDVQQDExZtYXJzaGFsbC51ay5vcmFjbGUuY29tMIGfMA0GCSqGSIb3DQEBA
   QUAA4GNADCBiQKBgQCEopgMZp1lI6jWXxb1rM1kWIc1l8bhiV/0UTcsdKzeaSHxbO
   SLO3Ed9kxNWAZgXaR9f5FBlwkaRJ+IR163e64v3SplHenxHfVRaHYWPZx4KlJz/6p
   Yd1fAlF0PdQm1DNoFtKmCHVk/cRuvGRpsp38l7K2mYlyQ+GxH38llS7g3owIDAQAB
   oAAwDQYJKoZIhvcNAQEFBQADgYEAD/sG1+rSI76OjihHg3WezT+VIbSRJxyly9nbx
   4uwXbDHh8DGgQLAXV51C9ioaMrm+dM0eygVDDMESXFxvJiYipS/pphgYt1xDBgnEH
   GcNiX3BnTaLNtzYlc5eAMsmbDlpk/qOxvQiH3bKN+UKYQlBXJZWPL6FusXu2LMTrk
   zsY=
   -----END NEW CERTIFICATE REQUEST-----
   ```

8. Submit the CSR file to a certificate authority (CA) of your choice. The CA returns a digital certificate for WebLogic Server. This certificate is signed by the CA and is often referred to simply as the server certificate.

9. In the directory you created for your keystore, save the server certificate, and also the trusted root CA certificate, in individual files. For example, the server certificate can be saved as `server.pem`, and the root CA certificate as `rootCA.pem`.

   If the CA has multiple root certificates, save them also in your keystore directory using names such as `rootCA2.pem`, `rootCA3.pem`, and so on.

10. Import each root CA certificate into your keystore using the following `keytool` command syntax:

    ```
    keytool -importcert -v -noprompt -trustcacerts -alias alias -file rootca_file
    -keystore keystore
    ```

    In the preceding syntax:

- *alias* represents the alias of the root CA certificate.
- *rootca_file* represents the name of the file that contains the root CA certificate.
- *keystore* represents the name of your keystore.

For example, the following command imports the root CA certificate in file `rootCA.pem` into the keystore, assigning it the alias `rootcacert`:

```
prompt> keytool -importcert -v -noprompt -trustcacerts -alias rootcacert -file
rootCA.pem -keystore keystore.jks
Enter keystore password:
Certificate was added to keystore
Storing keystore.jks
```

11. Import the server certificate into your keystore using the following `keytool` command syntax:

```
keytool -importcert -v -alias alias -file servercert_file -keystore keystore
```

In the preceding syntax:

- *alias* represents the alias of the server certificate, which must be the same as the private key alias assigned in Step 4.)
- *servercert_file* represents the name of the file that contains the server certificate.
- *keystore* represents the name of your keystore.

For example, the following command imports the server certificate `server.pem` into the keystore, using the alias (`server_cert`) assigned in Step 4:

```
prompt> keytool -importcert -v -alias server_cert -file server.pem -keystore
keystore.jks
Enter keystore password:
Certificate reply was installed in keystore
[Storing keystore.jks
]
```

12. To view the contents of the keystore, use the following `keytool` command syntax, where *keystore* represents the name of your keystore:

```
keytool -list -v -keystore keystore
```

For example:

```
prompt> keytool -list -v -keystore keystore.jks
Enter keystore password:

Alias name: rootcacert
Creation date: Sep 13, 2010
Entry type: trustedCertEntry

Owner: CN=SSL Training CA, OU=Support, O=Avitek, L=Reading, ST=Berkshire, C=GB
Issuer: CN=SSL Training CA, OU=Support, O=Avitek, L=Reading, ST=Berkshire, C=GB
Serial number: c47f4774c2ef014c
Valid from: Fri Jan 09 10:27:18 GMT 2009 until: Mon May 26 11:27:18 BST 2036
Certificate fingerprints:
MD5: E9:24:39:56:DE:34:44:DB:46:93:45:93:8E:82:66:AC
SHA1: 17:39:92:C0:43:9B:28:F3:C2:54:55:9B:5E:97:CA:EE:71:5D:9C:26
Signature algorithm name: SHA1withRSA
Version: 3
```

```
Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 67 57 BA 54 BB 9B C0 38 9A 71 AA 28 82 23 4B 08 gW.T...8.q.(.#K.
0010: 72 B9 FC C1 r...
]
]

#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
CA:true
PathLen:2147483647
]

#3: ObjectId: 2.5.29.35 Criticality=false

[CN=SSL Training CA, OU=Support, O=Avitek, L=Reading, ST=Berkshire, C=GB]
SerialNumber: [ c47f4774 c2ef014c]
]

*******************************************
*******************************************

Alias name: server_cert
Creation date: Sep 13, 2010
Entry type: PrivateKeyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=server.avitek.com, OU=Support, O=Avitek, L=Reading, ST=Berkshire,
C=GB
Issuer: CN=SSL Training CA, OU=Support, O=Avitek, L=Reading, ST=Berkshire, C=GB
Serial number: e
Valid from: Mon Sep 13 14:02:00 BST 2010 until: Sat Sep 22 14:02:00 BST 2012
Certificate fingerprints:
MD5: CB:B8:07:32:22:B5:76:78:44:BB:94:D2:CE:EF:A3:CA
SHA1: 1E:3E:C6:BC:17:EB:43:50:19:01:0B:11:50:D8:23:60:21:B2:57:3E
Signature algorithm name: MD5withRSA
Version: 1
Certificate[2]:
Owner: CN=SSL Training CA, OU=Support, O=Avitek, L=Reading, ST=Berkshire, C=GB
Issuer: CN=SSL Training CA, OU=Support, O=Avitek, L=Readin g, ST=Berkshire,
C=GB
Serial number: c47f4774c2ef014c
Valid from: Fri Jan 09 10:27:18 GMT 2009 until: Mon May 26 11:27:18 BST 2036
Certificate fingerprints:
MD5: E9:24:39:56:DE:34:44:DB:46:93:45:93:8E:82:66:AC
SHA1: 17:39:92:C0:43:9B:28:F3:C2:54:55:9B:5E:97:CA:EE:71:5D:9C:26
Signature algorithm name: SHA1withRSA
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 67 57 BA 54 BB 9B C0 38 9A 71 AA 28 82 23 4B 08 gW.T...8.q.(.#K.
0010: 72 B9 FC C1 r...
]
```

```
]

#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
CA:true
PathLen:2147483647
]

#3: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 67 57 BA 54 BB 9B C0 38 9A 71 AA 28 82 23 4B 08 gW.T...8.q.(.#K.
0010: 72 B9 FC C1 r...
]

[CN=SSL Training CA, OU=Support, O=Avitek, L=Reading, ST=Berkshire, C=GB]
SerialNumber: [ c47f4774 c2ef014c]
]

*******************************************
*******************************************
```

11.6 Using a Certificate Callback Handler to Validate End User Certificates

WebLogic Server provides a means to examine details about information passed by an end user issuing a request to determine whether authentication should succeed or fail. The details may include the end user's certificate, Subject, and IP address. This capability is provided by the `weblogic.security.SSL.CertificateCallback` interface, which you can implement to create a certificate callback handler. When configured with WebLogic Server, this callback handler is invoked automatically whenever a client request is received over a secure RMI connection; for example, one that uses the T3s or IIOPS protocols.

To configure a certificate callback handler so that it is in effect for all secure inbound RMI connections, you define it as a WebLogic Server system property that is passed in the server startup command.

The following topics describe how a certificate callback handler works and how to implement and configure one:

- Section 11.6.1, "How End User Certificate Callback Handlers Work"
- Section 11.6.2, "Creating a Certificate Callback Implementation"
- Section 11.6.3, "Configuring the Certificate Callback with WebLogic Server"

11.6.1 How End User Certificate Callback Handlers Work

When a client makes a secure RMI connection to a WebLogic Server instance that is configured with a certificate callback handler, WebLogic Server invokes the callback handler. The callback evaluates details about the end user that are contained in the connection request, then returns a boolean value indicating whether authentication is successful.

The CertificateCallback interface calls the `validate` method on an `CertificateCallbackInfo` instance, which contains methods to obtain the following information from the end user that is contained in the RMI connection request:

- Client host name, IP address, and port

- Client domain name
- Destination host name, IP address, and port
- Authenticated Subject
- Client certificate

The callback implementation includes the logic that evaluates the client data that is obtained and returns `true` or `false` as follows:

- If the callback returns `true`, authentication succeeds and the client connection to WebLogic Server is made.
- If the callback returns `false`, a RemoteException is thrown containing the "Authentication denied" message.

Note: If you use a certificate callback implementation in WebLogic Server, a callback is generated whenever a request is received over a secure port. As a result, using certificate callbacks may impose a performance overhead that should be taken into consideration.

11.6.2 Creating a Certificate Callback Implementation

The `weblogic.security.SSL.CertificateCallback` interface contains a single invocation on the `validate` method on a `weblogic.security.SSL.CertificateCallbackInfo` instance. The CertificateCallbackInfo instance contains methods to obtain details about the end user that are passed over the secure RMI connection.

You implement logic that evaluates the data that is returned and returns a `true` or `false`. The logic does not need to evaluate all data that is returned. Typically, only the certificate is evaluated; for example, obtaining the common name (cn) or distinguished name (dn).

For more information, see the following Javadoc in the *Java API Reference for Oracle WebLogic Server*:

- `weblogic.security.SSL.CertificateCallback` interface
- `weblogic.security.SSL.CertificateCallbackInfo` class

11.6.3 Configuring the Certificate Callback with WebLogic Server

To configure the callback with WebLogic Server, specify the callback implementation as a system property in the WebLogic Server start command. The property should point to the callback implementation class that is on the server's classpath. For example, if the callback implementation class is `MyCertificateCallback.java` in the package `com.mycompany.security`, and `MyCertificateCallback.class` is in the server's classpath, the following command sets the callback implementation property in WebLogic Server:

```
java weblogic.Server
-Dweblogic.security.SSL.CertificateCallback=com.mycompany.security.MyCertificateCallback
```

Note that if WebLogic Server is configured for one-way SSL, a client certificate is never sent to the server. Oracle recommends using certificate callbacks handlers only when WebLogic Server is configured for two-way SSL. For more information, see Chapter 12, "Configuring SSL".

12 Configuring SSL

This chapter describes how to configure SSL. Configuring SSL is an optional step; however, Oracle recommends SSL for production environments.

> **Note:** The following sections apply to WebLogic Server deployments that use the security features in this release of WebLogic Server as well as deployments that use Compatibility Security.
>
> All machines must be kept up to date with the current set of recommended patches from the operating system vendors.

This chapter includes the following sections:

12.1 SSL: An Introduction

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

network connection. Encryption makes data transmitted over the network intelligible only to the intended recipient.

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

> **Note:** WebLogic Server does not support SSL 2.0.

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

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

12.2 One-Way and Two-Way SSL

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

- With one-way SSL, the server must present a certificate to the client, but the client is not required to present a certificate to the server. The client must authenticate the server, but the server accepts a connection from any client. One-way SSL is common on the Internet where customers want to create secure connections before they share personal data. Often, clients will also use SSL to log on in order that the server can authenticate them.
- With two-way SSL (SSL with client authentication), the server presents a certificate to the client and the client presents a certificate to the server. WebLogic Server can be configured to require clients to submit valid and trusted certificates before completing the SSL connection.

12.3 Java Secure Socket Extension (JSSE) SSL Implementation Supported

This release of WebLogic Server uses an SSL implementation based on Java Secure Socket Extension (JSSE). JSSE is the Java standard framework for SSL and TLS and includes both blocking-IO and non-blocking-IO APIs, and a reference implementation including several commonly-trusted CAs.

The JSSE-based SSL implementation interoperates over SSL with instances of Weblogic Server version 8.1 and later that use the Certicom SSL implementation. That is, when WebLogic Server with JSSE SSL is used as either an SSL client or as the SSL server, it can communicate via SSL with instances of WebLogic Server (version 8.1 and later) that use the Certicom SSL implementation.

See Section 12.13, "Using the JSSE-Based SSL Implementation" for information about using JSSE.

See the *Java Secure Socket Extension (JSSE) Reference Guide* (`http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html`) for complete information on JSSE.

Note: As of WebLogic Server version 12.1.1, JSSE is the only SSL implementation that is supported. The Certicom-based SSL implementation is removed and is no longer supported in WebLogic Server.

12.4 Setting Up SSL: Main Steps

To set up SSL:

1. Obtain an identity (private key and digital certificates) and trust (certificates of trusted certificate authorities) for WebLogic Server. Use the digital certificates, private keys, and trusted CA certificates provided by WebLogic Server, the CertGen utility, the keytool utility, or a reputable vendor such as Entrust or Verisign to perform this step.

   Note: If you use the CertGen utility to generate certificates, see Section 11.4.1.2.2, "Limitation on CertGen Usage" for information about limitations on its use. Certificates generated by CertGen are for demo purposes only and should not be used in a production environment.

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

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

3. Configure the identity and trust keystores for WebLogic Server in the WebLogic Server Administration Console. See "Configure keystores" in the *Oracle WebLogic Server Administration Console Online Help*.

4. Set SSL configuration options for the private key alias and password in the WebLogic Server Administration Console. Optionally, set configuration options that require the presentation of client certificates (for two-way SSL). See "Servers: Configuration: SSL" and "Configure two-way SSL" in the *Oracle WebLogic Server Administration Console Online Help*.

   Note: FIPS mode is supported for JSSE via the RSA JSSE provider.

   To enable FIPS in JSSE, follow these steps:

   1. Put the `cryptojFIPS.jar` jar (*`WL_HOME`*`/server/lib/cryptojFIPS.jar`) in the classpath, ahead of `cryptoj.jar`. For example, you could add `cryptojFIPS.jar` to the `PRE_CLASSPATH` variable in the server start script, typically `startWebLogic.cmd/sh`).
   2. Enable the RSA JSSE provider, as described in Section 12.14, "Using the RSA JSSE Provider in WebLogic Server".

   FIPS 140-2 is a standard that describes U.S. Federal government requirements for sensitive, but unclassified use.

For information about configuring identity and trust for WebLogic Server, see the following sections:

- Section 11.4.1, "Obtaining Private Keys, Digital Certificates, and Trusted Certificate Authority Certificates"
- Section 11.4.2, "Storing Private Keys, Digital Certificates, and Trusted Certificate Authority Certificates"

12.5 Using Host Name Verification

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

WebLogic Server includes two host name verifiers, described in the following sections:

- Section 12.5.1, "Using the Default WebLogic Server Host Name Verifier"
- Section 12.5.2, "Using the Wildcarded Host Name Verifier"

As an alternative to the host name verifiers available from WebLogic Server, you can use a custom host name verifier. For information, see Section 12.5.3, "Using a Custom Host Name Verifier".

The default WebLogic Server host name verifier is enabled by default.

12.5.1 Using the Default WebLogic Server Host Name Verifier

As a function of the SSL handshake, WebLogic Server compares the common name in the SubjectDN in the SSL server's digital certificate with the host name of the SSL server used to accept the SSL connection. If these names do not match exactly, the SSL connection is dropped. The SSL client is the actual party that drops the SSL connection if the names do not match.

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

If you are using the default WebLogic Server host name verifier, host name verification passes if both of the following conditions exist:

- The host name in the certificate matches the local machine's host name.
- The URL specifies `localhost`, `127.0.01`, or the default IP address of the local machine.

> **Note:** If you are using the demo identity certificates in a multi-server domain, Managed Server instances will fail to boot if they are started using the fully-qualified DNS name of the Administration Server. For information about this limitation and suggested workarounds, see Section 11.4.1.2.2, "Limitation on CertGen Usage".

The default host name verifier is configured by default. No action is needed to use it.

For more information, see the following topics in *Oracle WebLogic Server Administration Console Online Help*:

- "Verify host name verification is enabled"
- "Disable host name verification"
- "Servers: Configuration: SSL"

12.5.1.1 Using the Default Host Name Verifier on Mac OS X Platforms

If WebLogic Server is installed on a Mac OS X platform that is running in a network in which the DHCP server assigns host names, by default Mac OS X dynamically overrides the host name set on your machine, using the one assigned by DHCP. Consequently, if you have generated demo identity certificates, host name verification may fail if the host name in your certificate does not match the one that has been dynamically reassigned to your machine. This host name reassignment can occur frequently, such as whenever the network is restarted.

To use demo identity certificates with WebLogic Server on Mac OS X platforms, do one of the following:

- Disable host name verification (not recommended if operating in a production environment).
- Prior to installing WebLogic Server, set a fixed host name on your machine. Depending on your environment, you may be able to do this by changing the value of the `HOSTNAME` property in `/etc/hostconfig` from `-AUTOMATIC-` to the name you wish to assign. For example:

  ```
  HOSTNAME=mymachine.mydomain.com
  ```

  In addition, you may also verify that your desired host name is set in the file `/Library/Preferences/SystemConfiguration/preferences.plist`. For more information, consult the Mac OS X documentation for your platform.

12.5.2 Using the Wildcarded Host Name Verifier

In addition to the default WebLogic Server host name verifier, WebLogic Server includes an alternative host name verifier called the **wildcarded host name verifier**. The wildcarded host name verifier works the same as the default WebLogic Server host name verifier; however, the wildcarded host name verifier also accepts the following additional SSL session certificates:

- Certificates that contain the asterisk wildcard character (`*`) in the host name that is obtained from the certificate's Subject CommonName attribute (that is, the CN domain)
- SubjectAlternativeName dnsName (SAN) certificates

12.5.2.1 How the Wildcarded Host Name Verifier Works

If the host name in the SSL session certificate contains a wildcard character that meets the following criteria, the certificate is accepted by the wildcarded host name verifier:

- The host name contains at least two dot (`.`) characters.
- The host name begins with an asterisk (`*`) and does not contain any additional asterisks.
- When the asterisk (`*`) is stripped from the CN string, the remaining string must:
  - Represent the domain.
  - Include a leading dot (`.`) character .

- Be identical to the ending string of the incoming request domain.
- Not include an additional dot (.) character. (This prevents the wildcard from representing subdomains.

If the host name in the SSL session certificate does not exactly match the expected server name attribute, and the host name also cannot successfully be validated in accordance with the wildcard acceptance criteria, the wildcarded host name verifier attempts to validate the SAN extensions.

The SAN extensions are obtained from the SSL session certificate. The SAN extension values are iterated using a case-insensitive match. For any iterated value, if the dnsName attribute in the certificate matches the request URL, host name verification succeeds.

12.5.2.2 Configuring the Wildcarded Host Name Verifier

The wildcarded host name verifier class name is `weblogic.security.utils.SSLWLSWildcardHostnameVerifier`. To configure the wildcarded host name verifier, specify this class as a custom host name verifier in the Servers: Configuration: SSL page of the Administration Console. The wildcarded host name verifier has no parameters with which it must be configured. For more information, see "Configure a custom host name verifier" in the *Oracle WebLogic Server Administration Console Online Help*.

12.5.3 Using a Custom Host Name Verifier

The class that implements the custom host name verifier must be specified in the CLASSPATH of WebLogic Server (when acting as an SSL client) or a standalone SSL client.

For more information, see "Configure a custom host name verifier" in *Oracle WebLogic Server Administration Console Online Help*.

12.6 Specifying a Client Certificate for an Outbound Two-Way SSL Connection

When making an outbound two-way SSL connection, WebLogic Server by default uses its server certificate to establish its identity as a client. However, you can alternatively specify a separate client certificate to establish identity instead. This capability is particularly useful when WebLogic Server is acting as a client making two-way SSL connection.

> **Note:** Switching WebLogic Server's identity to a client certificate is supported only when making an outbound two-way SSL connection. For inbound SSL connections, where Weblogic Server is acting as an SSL server, the server certificate is always used for identity.

To use this capability, do the following:

1. Add a client certificate to WebLogic Server's identity keystore and note the name of the alias under which the private key and public certificate are stored. This needs to be done only once. After completing the following steps, the ability to use a client identity for making an outbound two-way SSL connection is always available for the current WebLogic Server instance.

   To add a client certificate to the identity keystore, complete the following steps:

a. Create a client key pair (a public key and associated private key) and an alias for the private key and store it the WebLogic Server identity keystore. You can do this using the keytool utility.

b. Generate a Certificate Signing Request (CSR) and submit it to a certificate authority (CA), who returns the CA-signed client certificate. Oracle recommends using the same CA as for the server certificate so that both certificates have the same trusted root CA.

c. Store the CA-signed client certificate in the identity keystore. (If the client certificate is signed by the same CA as the server certificate, you can skip the step of storing the root CA certificate in the trust keystore because it is already there.

2. To initiate an outbound two-way SSL connection using the client certificate, create a WLST script that does the following:

   a. Connects to the WebLogic Server instance.

   b. Sets the `SSLMBean.UseServerCerts` attribute to `true`, which establishes the server identity for the outbound connection.

   c. Switches to the identity of the client certificate by setting the `SSLMBean.UseClientCertForOutbound` attribute to `true`.

   d. Specifies the client certificate private key passphrase, using the `SSLMBean.ClientCertPrivateKeyPassPhrase` attribute, and the client certificate keystore alias, using the `SSLMBean.ClientCertAlias` attribute.

   See Example 12–1 for an example of this WLST script.

Example 12–1 Sample WLST Script that Initiates an Outbound Two-Way SSL Connection Using a Client Identity

```
url="t3://localhost:7001"
adminUsername="weblogic"
adminPassword="password"
connect(adminUsername, adminPassword, url)
edit()
server=cmo.lookupServer('myserver')
cd('Servers')
cd('myserver')
startEdit()
cd('SSL')
cd('myserver')
ssl = server.getSSL()
ssl.setUseServerCerts(true)
ssl.setUseClientCertForOutbound(true)
ssl.setClientCertAlias("myClientCert")
ssl.setClientCertPrivateKeyPassPhrase("myClientCertPrivateKeyPassPhrase")
save()
activate()
disconnect()
exit()
```

To restore use of the server identity certificate for outbound SSL connections, enter a WLST command that sets the `SSLMBean.UseClientCertForOutbound` attribute to `false`.

Note the following:

- Note that the values of the `SSLMBean.ClientCertPrivateKeyPassPhrase` and `SSLMBean.ClientCertAlias` attributes are persisted and are used the next time an outbound two-way SSL connection using a client identity is made (that is, the next time the `SSLMBean.UseClientCertForOutbound` attribute is set to `true`).
- The SSLMBean attributes used for specifying a client certificate for outbound SSL connections are not available from the WebLogic Server Administration Console.

12.7 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 encryption strength that is allowed
- 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

SSL debugging dumps a stack trace whenever an ALERT is created in the SSL process. The types and severity of the ALERTS are defined by the Transport Layer Security (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, confirm whether there is a trace message after the ALERT. An ALERT received after the trace message indicates the failure occurred on the peer. To determine the problem, you need to enable SSL debugging on the peer in the SSL connection.

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

- The correct `config.xml` file was loaded
- The setting for 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.

12.7.1 Command-Line Properties for Enabling SSL Debugging

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

```
-Djavax.net.debug=all
```

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

Note the following:

- The `-Djavax.net.debug=all` property enables debug logging within the JSSE-based SSL implementation.
- The `-Dssl.debug=true` and `-Dweblogic.StdoutDebugEnabled=true` command-line properties enable debug logging of the SSL calling code within Weblogic Server.

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

For information about using WebLogic logging properties with the JSSE SSL logging system, see Section 12.13.4, "Using Debugging with JSSE SSL".

For information about debugging utilities available for JSSE, see "Debugging Utilities" in the *Java™ Secure Socket Extension (JSSE) Reference Guide*, available at the following URL:

http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html#Debug

12.8 SSL Session Behavior

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

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

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

Clients that are not configured to use SSL sessions must call `setEnableSessionCreation(false)` on the SSL socket to ensure that no SSL sessions are cached. This setting only controls whether an SSL session is added to the cache; it does not stop an SSL socket from finding an SSL session that was already cached. For example, SSL socket 1 caches the session, SSL socket 2 sets `setEnableSessionCreation` to false but it can still reuse the SSL session from SSL socket 1 because 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 used by a previous session can resume a previous session as long as you create the SSL socket using an SSL socket factory from the SSL context that has the SSL session in its cache.

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

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.

You can use the `weblogic.security.SSL.sessionCache.ttl` command-line argument to modify the default server-session time-to-live for SSL session caching. For information, see "SSL" in *Command Reference for Oracle WebLogic Server*. Note that the `weblogic.security.SSL.sessionCache.size` command-line argument is ignored.

12.9 Configuring RMI over IIOP with SSL

Use SSL to protect Internet Interop-Orb-Protocol (IIOP) connections to Remote Method Invocation (RMI) remote objects. SSL secures connections through authentication and encrypts the data exchanged between objects.

To use SSL to protect RMI over IIOP connections:

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 *Developing RMI Applications for Oracle WebLogic Server*.

12.10 SSL Certificate Validation

WebLogic Server ensures that each certificate in a certificate chain was issued by a certificate authority. All X509 V3 CA certificates used with WebLogic Server must have the Basic Constraint extension defined as CA, thus ensuring that 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.

Notes: Note the following:

- Weblogic Server uses RSA Cert-J 3.1 for certain certificate processing.
- If WebLogic Server is booted with a certificate chain that will not pass the certificate validation, an information message is logged noting that clients could reject it.

12.10.1 Controlling the Level of Certificate Validation

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

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

Table 12–1 describes the options for the command-line argument.

Table 12–1 Options for -Dweblogic.security.SSL.enforceConstraints

Option	Description

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

`strong_nov1cas`	Functions the same as the `strong` option, described in the preceding row, with the additional constraint that X509 version 1 CA certificates are rejected. For example: `-Dweblogic.security.SSL.enforceConstraints=strong_nov1cas`
`strict`	Use this option to ensure the Basic Constraints extension on the CA certificate is defined as CA and set to critical. This option enforces the IETF RFC 2459 standard. For example: `-Dweblogic.security.SSL.enforceConstraints=strict` This option is not the default because a number of commercially available CA certificates do not conform to the IETF RFC 2459 standard.
`strict_nov1cas`	Functions the same as the `strict` option, described in the preceding row, with the additional constraint that X509 version 1 CA certificates are rejected. For example: `-Dweblogic.security.SSL.enforceConstraints=strict_nov1cas`
`off`	Use this option to turn off checking for the Basic Constraints extension. The rest of the certificate is still validated. For example: `-Dweblogic.security.SSL.enforceConstraints=off` Oracle does not recommend using this option in a production environment. Instead, purchase new CA certificates that comply with the IETF RFC 2459 standard. CA certificates from most commercial certificate authorities should work with the default strong option.

12.10.2 Accepting Certificate Policies in Certificates

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

Note: The `weblogic.security.SSL.allowedcertificatepolicyids` argument is currently not supported in WebLogic Server when the JSSE-based SSL implementation is enabled.

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

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

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

12.10.3 Checking Certificate Chains

Use the WebLogic Server ValidateCertChain command-line utility to confirm whether an existing certificate chain will be rejected by WebLogic Server. The utility validates certificate chains from PEM files, PKCS-12 files, PKCS-12 keystores, and JKS keystores. A complete certificate chain must be used with the utility. The following is the syntax for the ValidateCertChain command-line utility:

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

Example of valid certificate chain:

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

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

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

Certificate chain appears valid
```

Example of invalid certificate chain:

```
java utils.ValidateCertChain -jks mykey mykeystore

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

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

Certificate chain is invalid
```

12.10.4 Using Certificate Lookup and Validation Providers

WebLogic Server SSL has built-in certificate validation. Given a set of trusted CAs, this validation:

- Verifies that the last certificate in the chain is either a trusted CA or is issued by a trusted CA.
- Completes the certificate chain with trusted CAs.
- Verifies the signatures in the chain.
- Ensures that the chain has not expired.

You can use certificate lookup and validation (CLV) providers to perform additional validation on the certificate chain. WebLogic Server includes two CLV providers:

- WebLogic CertPath Provider—Completes certificate paths and validates certificates using the trusted CA configured for a particular server instance, providing the same functionality as the built-in SSL certificate validation. This is configured by default.
- Certificate Registry—The system administrator makes a list of trusted CA certificates that are allowed access to the server; a certificate is valid if the end certificate is in the registry. The administrator revokes a certificate by removing it from the certificate registry, which is an inexpensive mechanism for performing revocation checking. This is not configured by default.

Alternatively, you can write a custom CertPathValidator to provide additional validation on the certificate chain. See "CertPath Providers" in *Developing Security Providers for Oracle WebLogic Server*.

12.10.5 How SSL Certificate Validation Works in WebLogic Server

Outbound SSL and two-way inbound SSL in a WebLogic Server instance receive certificate chains during the SSL handshake that must be validated. An example of two-way inbound SSL is a browser connecting to a Web application over HTTPS where the browser sends the client's certificate chain to the Web application. The inbound certificate validation setting is used for all two-way client certificate validation in the server.

Examples of WebLogic Server using outbound SSL (that is, acting as an SSL client) include:

- Connecting to the Node Manager
- Connecting to another WebLogic Server instance over the Administration port
- Connecting to an external LDAP server, such as the LDAPAuthenticator

Using the Administration Console or WLST, you can independently configure inbound and outbound SSL certificate validation using these `SSLMBean` attributes: `InboundCertificateValidation` and `OutboundCertificateValidation`.

Legal values for both attributes are:

- `BUILTIN_SSL_VALIDATION`: Use the built-in SSL certificate validation code to complete and validate the certificate chain. That is, configure SSL to work as it has in previous releases. This is the default behavior.
- `BUILTIN_SSL_VALIDATION_AND_CERT_PATH_VALIDATORS`: Use the built-in trusted CA-based validation and the configured CertPathValidator providers to perform additional validation. That is, configure SSL to work as it has in previous releases and to do extra validation.

See:

- "SSLMBean" in the *MBean Reference for Oracle WebLogic Server*

- "Set Up SSL" in the *Oracle WebLogic Server Administration Console Online Help*

12.10.6 Troubleshooting Problems with Certificate Validation

If SSL communications that worked properly in a previous release of WebLogic Server start failing unexpectedly, the likely problem is that the certificate chain 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
  ```

  Note: Additional detailed debug logging may be enabled using the following command-line property:

  ```
  -Djavax.net.debug=all
  ```

  For more information, see Section 12.7.1, "Command-Line Properties for Enabling SSL Debugging".

  The following message indicates the SSL failure results from 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 you use one-way SSL, look for this error in the client log. With two-way SSL, look for this error in the client and server logs.

12.11 Using JCE Providers with WebLogic Server

Note: Java Cryptography Extension (JCE) providers are written using the application programming interfaces (APIs) in the JCE. This type of provider is different from the providers written using the WebLogic Security Service Provider Interfaces (SSPIs).

WebLogic Server supports the use of the following JCE providers:

- The RSA JCE Provider. As of WebLogic Server version 12.1.2, the RSA JCE provider is included with WebLogic Server. The RSA JCE provider (Crypto-J 5.0) is located in `cryptoj.jar`, which is in the WebLogic Server classpath by default.

  The RSA CryptoJ documentation describes three ways to use the RSA's JCE Provider:

  – Static registration (for example, by editing `java.security`).

```
security.provider.1=com.rsa.jsafe.provider.JsafeJCE
```

– Dynamic registration at runtime.

```
// Create a Provider object
Provider jceProvider = new com.rsa.jsafe.provider.JsafeJCE();
// Add the JCE Provider class to the current list of providers available on
the system.
Security.insertProviderAt (jceProvider, 1);
```

– Use the RSA's JCE provider directly without registration.

```
// Create a Provider object.
Provider jceProvider = new com.rsa.jsafe.provider.JsafeJCE();
// Pass the provider into the getInstance call.
Cipher cipher = Cipher.getInstance("AES", jceProvider);
```

- The JDK JCE provider (`SunJCE`). For more information about the features in `SunJCE`, see the *Java™ Cryptography Architecture (JCA) Reference Guide* at `http://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html`.

  The JCA framework includes an ability to enforce restrictions regarding the cryptographic algorithms and maximum cryptographic strengths available to applets/applications in different jurisdiction contexts (locations). Any such restrictions are specified in "jurisdiction policy files". For more information, see the *Java™ Cryptography Architecture (JCA) Reference Guide*.

  WebLogic Server will continue to control the strength of the cryptography used by the WebLogic Server Application Programming Interfaces (APIs). Client code without the appropriate domestic strength cryptography setting will only be able to use the Java SE export strength default cryptography. On the server, WebLogic Server will enable either export or domestic strength cryptography.

- The nCipher JCE provider. See `http://www.ncipher.com`.

  SSL is a key component in the protection of resources available in Web servers. However, heavy SSL traffic can cause bottlenecks that affect the performance of Web servers. JCE providers like nCipher that use a hardware card for encryption offload SSL processing from Web servers, which frees the servers to process more transactions. They also provide strong encryption and cryptographic processes to preserve the integrity and secrecy of keys.

12.11.1 Installing the nCipher JCE Provider

To install the nCipher JCE provider:

1. Install and configure the hardware for the nCipher JCE provider according to the product's documentation.
2. Install the files for the nCipher JCE provider. The following files are required:
   - Jurisdiction policy files—The JDK installs these files by default but they are of limited export strength.
   - 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 JCE provider JAR files

Choose an installation method for the files:

- Install files as an extension. Copy the files to one of the following locations:

  JAVA_HOME/jre/lib/ext

  For example:

  ORACLE_HOME/jdk7_10/jre/lib/ext

- Install files in the CLASSPATH of the server.

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

   JAVA_HOME/jre/lib/security/java.security

   Specify the nCipher JCE provider as:

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

   where *n* specifies the preference order that 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=sun.security.rsa.SunRsaSign
   security.provider.3=com.ncipher.provider.km.mCipherKM
   ```

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

12.12 Specifying the Version of the SSL Protocol

At the start of the SSL handshake, the SSL peers determine the highest protocol version both peers support. However, you can configure Weblogic Server to limit the lowest supported versions of SSL and TLS that are enabled for SSL connections.

To specify the SSL and TLS versions enabled for the SSL handshake, you can set either of the following system properties in the command-line argument that starts WebLogic Server:

- `weblogic.security.SSL.protocolVersion`
- `weblogic.security.SSL.minimumProtocolVersion`

Note the following regarding SSL protocol support in WebLogic Server:

- When the JSSE-based SSL implementation is enabled (see Section 12.13, "Using the JSSE-Based SSL Implementation"), SSL protocol support is dependent on the JSSE provider that is installed.
- When WebLogic Server is acting as an SSL server, the protocol that the client specifies as preferred in its client hello message is used, if supported.
- WebLogic Server does not support SSL V2.0.

The following sections explain how to use these two system properties:

- Section 12.12.1, "Using the weblogic.security.SSL.protocolVersion System Property"
- Section 12.12.2, "Using the weblogic.security.SSL.minimumProtocolVersion System Property"

12.12.1 Using the weblogic.security.SSL.protocolVersion System Property

While in most cases the most recent version of the SSL or TLS protocol is desirable, peers may not support it. You may want to specify the enabled SSL or TLS protocol based on circumstances (compatibility, SSL performance, and environments with maximum security requirements) that make the TLS V1 protocol more desirable for enabling acceptable SSL and TLS protocols. Specifying the `weblogic.security.SSL.protocolVersion` system property in a command-line argument that starts WebLogic Server lets you specify the protocol that is used for SSL connections.

The following command-line arguments can be specified so that WebLogic Server supports only SSL V3.0 or TLS connection.

- `-Dweblogic.security.SSL.protocolVersion=SSL3`—Only SSL V3.0 messages are sent and accepted. Attempts by clients to establish connections with a prior SSL version will be denied by WebLogic Server, with a denial message returned to the client.
- `-Dweblogic.security.SSL.protocolVersion=TLS1`— This property value enables any protocol starting with "TLS" for messages that are sent and accepted; for example, TLS V1.0, TLS V1.1, and TLS V1.2.
- `-Dweblogic.security.SSL.protocolVersion=ALL`—This is the default behavior. If ALL is selected, the default depends on the JSSE provider and JDK version. For the supported protocol version table for Sun JSSE, see `http://docs.oracle.com/javase/7/docs/technotes/guides/security/SunProviders.html#SunJSSEProvider`.

Note the following:

- The SSL V3.0 and TLS V1 protocols can not be interchanged. Use only the TLS V1 protocol if you are certain all desired SSL clients are capable of using the protocol.
- The SSL version SSLv2Hello is one of the protocols enabled by default and is enabled only under the following conditions:
  - You do not explicitly set any values for the `weblogic.security.SSL.protocolVersion` or `weblogic.security.SSL.minimumProtocolVersion` system properties.
  - You set the `weblogic.security.SSL.protocolVersion=ALL` system property.
- Not setting the `weblogic.security.SSL.protocolVersion` system property enables the SSLv2Hello, SSLv3, and TLSv1 protocols. In addition, for JSSE, all versions starting with "TLS" are also enabled.
- If you set valid, supported protocols for the `weblogic.security.SSL.minimumProtocolVersion` system property, the protocol value you set for `weblogic.security.SSL.protocolVersion` is ignored.

Caution: If you specify the `TLS1` or `ALL` value in this system property, all versions of TLS V1 supported by the SSL provider are enabled for use in SSL connections. The JSSE-based implementation supports TLS V1.0, TLS V1.1, and TLS V1.2. Oracle recommends the use of TLS V1.1 or later in a production environment, which is available by using the `weblogic.security.SSL.minimumProtocolVersion` system property. For more information, see Section 12.12.2, "Using the weblogic.security.SSL.minimumProtocolVersion System Property".

12.12.2 Using the weblogic.security.SSL.minimumProtocolVersion System Property

In a production environment, Oracle recommends TLS V1.1, or later, for sending and receiving messages in an SSL connection:

To control the minimum versions of SSL V3.0 and TLS V1 that are enabled for SSL connections, set the `weblogic.security.SSL.minimumProtocolVersion=`*`protocol`* system property as an option in the command line that starts WebLogic Server. This system property accepts one of the following values for *`protocol`*:

Value	Description
`SSLv3`	Specifies SSL V3.0 as the minimum protocol version enabled in SSL connections.
`TLSv1`	Specifies TLS V1.0 as the minimum protocol version enabled in SSL connections.
`TLSv`*`x.y`*	Specifies TLS V*x.y* as the minimum protocol version enabled in SSL connections, where: ■ *x* is an integer between 1 and 9, inclusive ■ *y* is an integer between 0 and 9, inclusive For example, `TLSv1.2`.

The specific protocols that are enabled by each of the values you can specify for the `weblogic.security.SSL.minimumProtocolVersion` system property depend upon the SSL implementation with which WebLogic Server is configured. The next section identifies these protocols for the JSSE-based SSL implementation available in WebLogic Server

12.12.2.1 Protocols Enabled with the JSSE-Based SSL Implementation

When WebLogic Server is configured to use the JSSE-based SSL implementation and you specify a minimum protocol version using the `weblogic.security.SSL.minimumProtocolVersion` system property, the specific SSL and TLS protocols that are enabled depend on the protocols that are supported in the SSL implementation, as follows:

- If the particular minimum protocol version you specify is supported, WebLogic Server enables that protocol version *and* all later protocol versions that are supported.

  For example:

If you specify . . .	. . . and the JSSE-based SSL implementation supports . . .	. . . the following protocols are enabled
TLSv1	SSLv2Hello[1] SSLv3 TLSv1 TLSv1.1 TLSv1.2	TLSv1 TLSv1.1 TLSv1.2

[1] The SSLv2Hello protocol is never enabled in the JSSE-based SSL implementation when using the `weblogic.security.SSL.minimumProtocolVersion` system property, even though it may be supported.

- If the particular minimum protocol version you specify is *not* supported, Weblogic Server enables the next lower protocol and all later protocols that are supported. Note that the lowest protocol will be limited to SSLv3. That is, SSLv2Hello will not be enabled, even though it may be the next lowest protocol.

  For example:

If you specify . . .	. . . and the JSSE-based SSL implementation supports . . .	. . . the following protocols are enabled
TLSv1	SSLv2Hello[1] SSLv3 TLSv1.1 TLSv1.2	SSLv3 TLSv1.1 TLSv1.2

- If the exact minimum protocol you specify is *not* supported, and no older (lower) protocol is supported that is SSLv3 or higher, WebLogic Server enables all newer (higher) supported versions. This case usually applies when SSLv3 is set as the minimum; SSLv2Hello is not valid as the next lowest protocol, so only the available later protocols are enabled.

  For example:

If you specify . . .	. . . and the JSSE-based SSL implementation supports . . .	. . . the following protocols are enabled
SSLv3	SSLv2Hello[1] TLSv1 TLSv1.1 TLSv1.2	TLSv1 TLSv1.1 TLSv1.2

- If the particular minimum protocol you specify is invalid, WebLogic Server enables SSLv3 *and* all later protocol versions that are supported.

  For example:

If you specify . . .	. . . and the JSSE-based SSL implementation supports . . .	. . . the following protocols are enabled
TSLv0	SSLv2Hello[1] SSLv3 TLSv1 TLSv1.1 TLSv1.2	SSLv3 TLSv1 TLSv1.1 TLSv1.2

12.13 Using the JSSE-Based SSL Implementation

The JSSE-based SSL implementation is enabled by default in Weblogic Server.

Notes: Note the following:

- As of WebLogic Server version 12.1.1, JSSE is the only SSL implementation that is supported. The Certicom-based SSL implementation is removed and is no longer supported in WebLogic Server.
- SHA-2 signed certificates are supported in the JSSE SSL implementation provided in WebLogic Server.
- Although JSSE supports Server Name Indication (SNI) in its SSL implementation, WebLogic Server does not support SNI.

The following topics explain how to use the JSSE-based SSL implementation and describe key differences with the Certicom-based implementation:

- Section 12.13.1, "System Property Differences Between the JSSE-Based and Certicom SSL Implementations"
- Section 12.13.2, "SSL Performance Considerations"
- Section 12.13.3, "Cipher Suites"
- Section 12.13.4, "Using Debugging with JSSE SSL"

12.13.1 System Property Differences Between the JSSE-Based and Certicom SSL Implementations

Table 12–2 shows the differences in how the JSSE-based SSL implementation handles the WebLogic system properties.

Table 12–2 System Properties Differences

System Property	JSSE Applicability	Description
`weblogic.security.SSL.ignoreHostnameVerification`	This property continues to work and is not affected by the JSSE integration.	Does not verify the hostname in the URL to the hostname in the certificate.
`weblogic.ReverseDNSAllowed`	This property continues to work and is not affected by the JSSE integration.	If set to true then use reverse DNS lookup to figure out if `urlhostname` is a loopback address ("localhost" or "127.0.0.1", or the IPV6 equivalent.
`weblogic.security.SSL.trustedCAKeyStore`	This property continues to work and is not affected by the JSSE integration.	Loads the trusted CA certificates from that keystore.
`weblogic.security.SSL.verbose`	Use this property in combination with `javax.net.debug=all` to get verbose debug output from the SSL calling code and the JSSE-based implementation.[1]	For additional SSL debugging when `-Dssl.debug=true` is used.

Table 12–2 (Cont.) System Properties Differences

System Property	JSSE Applicability	Description
`ssl.debug=true`	Use this property in combination with `javax.net.debug=ssl` to get debug output from the SSL calling code and the JSSE-based implementation.[1]	Displays SSL debug information to the console or logs. This property is for the calling WebLogic code. The JSSE-based SSL implementation has its own logging system, which is activated by the `javax.net.debug` property. **Note:** You can set JSSE logging (`javax.net.debug`) independently of WebLogic SSL logging (`ssl.debug`).
`weblogic.security.SSL.disableJsseCipherSuiteAliases=true|false`	The default is `false`.	Disables the conversion of Certicom cipher suite names to `SunJSSE` cipher suite names, where applicable. By default, Certicom cipher suite names are converted to JSSE cipher suite names when JSSE is used for SSL. For a list of Certicom cipher suite names and their `SunJSSE` equivalents, see Table 12–3.
`weblogic.security.SSL.ignoreHostnameVerify`	This property continues to work and is not affected by the JSSE integration.	See `weblogic.security.SSL.ignoreHostnameVerification`
`weblogic.security.SSL.HostnameVerifier=`*`classname`*	This property continues to work and is not affected by the JSSE integration.	Specifies the class name of a custom hostname verification class.
`weblogic.security.SSL.protocolVersion=`*`protocol`*	This property continues to work and is not affected by the JSSE integration. The supported protocol values are mapped to the corresponding protocols supported by JSSE.	See Section 12.12, "Specifying the Version of the SSL Protocol".
One of the following: ■ `weblogic.security.SSL.allowUnencryptedNullCipher` ■ `SSLMBean.SetAllowUnencryptedNullCipher(boolean)` ■ `weblogic.security.disableNullCipher`	`SunJSSE` supports the following two null ciphers, but they are not enabled by default: ■ `SSL_RSA_WITH_NULL_MD5` ■ `SSL_RSA_WITH_NULL_SHA` If this setting is enabled, these two null ciphers are added to the cipher list.	By default, this control is not set and the use of a null cipher is not allowed on the server. In such a configuration, if the SSL clients want to use the null cipher suite (by indicating `SSL_RSA_WITH_NULL_MD5` as the only supported cipher suite), the SSL handshake will fail. If you set this control, the null cipher suite (for example, `SSL_RSA_WITH_NULL_MD5`) is added to the list of supported cipher suites by the server. The SSL connection has a chance to use the null cipher suite if the client wants to do so. If the null cipher suite is used, the message will be unencrypted. **Caution**: Do not set this control in a production environment unless you are aware of the implications and consequences of doing so.

Table 12–2 (Cont.) System Properties Differences

System Property	JSSE Applicability	Description
`weblogic.security.SSL.enforceConstraints=option`	`Off` is not supported, but other options are supported.	Ensures that the Basic Constraints extension on the CA certificate is defined as CA. See Section 12.10.1, "Controlling the Level of Certificate Validation".
`weblogic.security.SSL.allowedcerticatepolicyids`	Not supported.	WebLogic Server offers limited support for Certificate Policy Extensions in X.509 certificates. See Section 12.10.2, "Accepting Certificate Policies in Certificates".
`weblogic.security.SSL.nojce`	Not supported.	See Section 12.4, "Setting Up SSL: Main Steps".

[1] This WebLogic system property is applicable to both the Certicom and JSSE-based SSL implementations. However, for JSSE, this property affects only the SSL calling code, not the JSSE-based implementation. For more information about the `javax.net.debug` system property and debugging the JSSE-based SSL implementation, see "Debugging Utilities" in the Java Secure Socket Extension (JSSE) Reference Guide at http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html#Debug.

12.13.2 SSL Performance Considerations

When WebLogic Server is configured with JDK 7, you may find that the out-of-the-box SSL performance slower than in previous WebLogic Server releases. This performance change is due to the stronger cipher and MAC algorithm used by default when JDK 7 is used with the JSSE-based SSL provider in WebLogic Server. Specifically, AES is used for encryption, and SHA1 is used for hashes. (This cipher combination is typically designated as AES + SHA1 here; that is, cipher + MAC algorithm.)

Previous versions of WebLogic Server used the RC4 and MD5 cipher combination (RC4 + MD5) for SSL connections. In terms of performance, AES + SHA1 is slower than RC4 + MD5. Although AES + SHA1 is recommended, you can configure WebLogic Server to restrict the stronger ciphers and cause RC4 + MD5 to be used instead for SSL. Although RC4 + MD5 is less secure than AES + SHA1, it may be acceptable depending on the security requirements of a particular WebLogic Server environment.

> **Note:** Oracle strongly recommends the stronger security provided by AES + SHA1 for SSL connections.

To configure WebLogic Server to use RC4 + MD5, add the following property to the file *JAVA_HOME*`/jre/lib/security/java.security`:

```
jdk.tls.disabledAlgorithms=AES, DESede, DES, SHA1, SHA
```

The preceding property disables the stronger ciphers that are used by default for SSL connections and allows RC4 + MD5 to be used instead. For more information about cipher combination priorities in JDK 7, see http://bugs.sun.com/view_bug.do?bug_id=6996365.

12.13.3 Cipher Suites

This topic includes the following sections:

- Section 12.13.3.1, "List of Supported Cipher Suites"
- Section 12.13.3.2, "Backward Compatibility of Supported Cipher Suites"

12.13.3.1 List of Supported Cipher Suites

The set of cipher suites supported by the JDK default JSSE provider, `SunJSSE`, is available at the following URL:

`http://docs.oracle.com/javase/7/docs/technotes/guides/security/SunProviders.html#SunJSSEProvider`

12.13.3.2 Backward Compatibility of Supported Cipher Suites

For backward compatibility, the JSSE-based SSL implementation accepts Certicom cipher suite names for cipher suites that are compatible with `SunJSSE`. (See "Cipher Suites" in *Understanding Security for Oracle WebLogic Server* for a list of Certicom cipher suites.) The Certicom cipher suite names are converted for you to `SunJSSE` equivalents, usually replacing the "TLS_" prefix with "SSL_", as shown in Table 12–3.

Please keep the following in mind as you consider backward compatibility with Certicom cipher suites:

- With JSSE, the cipher suites selected by default are stronger as compared to Certicom SSL and have slower performance. The security policies in your environment typically set the requirements for the cipher suites that must be used. However, for highly secure environments, using the strongest available cipher that provides acceptable performance is recommended.
- For operations where enabled or supported cipher suites are returned, both the Certicom and `SunJSSE` names of the cipher suites are returned. (Note that the `weblogic.security.SSL.disableJsseCipherSuiteAliases=true` property, described in Table 12–2, disables this behavior.)
- For operations where you specify enabled cipher suites, you can use either the equivalent Certicom cipher suite names, or the `SunJSSE` names. The Certicom cipher suites, and their `SunJSSE` equivalents, are listed in Table 12–3. (Oracle does not encourage future use of Certicom cipher suite names.)
- The `_DSS_` cipher suites requires certificates signed with DSS, the Digital Signature Standard defined by NIST FIPS Pub 186. DSA is the key generation scheme as described in FIPS 186.
- The `_anon_` cipher suites are disabled by default, and cannot be managed from the WebLogic Server Administration Console. You must use WLST instead, as described in Setting Cipher Suites Using WLST: An Example.
- To use the Kerberos cipher suites `TLS_KRB5_***`, you must have KDC accounts set up. See the *Java Secure Socket Extension (JSSE) Reference Guide* (`http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html#KRB`) for more details on the Kerberos requirements.

12.13.3.3 Using Anonymous Ciphers

The following anonymous ciphers are not supported out-of-the-box in the JSSE-based WebLogic SSL implementation in WebLogic Server:

- `TLS_DH_anon_WITH_3DES_EDE_CBC_SHA`
- `TLS_DH_anon_WITH_RC4_128_MD5`

- TLS_DH_anon_WITH_DES_CBC_SHA
- TLS_DH_anon_EXPORT_WITH_RC4_40_MD5
- TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA

However, if you want to enable any of the preceding anonymous ciphers, include the following argument in the Java command that starts WebLogic Server:

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

In most cases, enabling anonymous ciphers is required when WebLogic Server, or its deployed application, acts as a SSL client that is making an outbound connection to an SSL server (for example, an LDAP server or RDBMS system) that is configured to use anonymous ciphers only. A typical use case is connecting to an Oracle Internet Directory instance that is configured in no-auth mode.

Note: Oracle does not recommend the use of anonymous ciphers in production environments.

12.13.3.4 Cipher Suite Name Equivalents

By default, Certicom cipher suite names are converted to SunJSSE cipher suite names when WebLogic Server is configured to use the JSSE-based SSL implementation. Table 12–3 lists each cipher suite supported in the (removed) WebLogic Server Certicom SSL implementation and its SunJSSE equivalent. The TLS_ name is the Certicom cipher suite name; the SSL_ name is the equivalent SunJSSE provider cipher suite name.

Table 12–3 Cipher Suite Name Equivalence

Certicom Cipher Suite	SunJSSE Equivalent Cipher Suite
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA	SSL_DHE_DSS_WITH_DES_CBC_SHA
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DHE_RSA_WITH_DES_CBC_SHA	SSL_DHE_RSA_WITH_DES_CBC_SHA
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA	SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
TLS_DH_anon_EXPORT_WITH_RC4_40_MD5	SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
TLS_DH_anon_WITH_DES_CBC_SHA	SSL_DH_anon_WITH_DES_CBC_SHA
TLS_DH_anon_WITH_RC4_128_MD5	SSL_DH_anon_WITH_RC4_128_MD5
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5	SSL_RSA_EXPORT_WITH_RC4_40_MD5
TLS_RSA_WITH_3DES_EDE_CBC_SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA
TLS_RSA_WITH_RC4_128_MD5	SSL_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA	SSL_RSA_WITH_RC4_128_SHA

12.13.3.5 Setting Cipher Suites Using WLST: An Example

The following example shows using a WLST script that sets the cipher suites `SSL_RSA_WITH_RC4_128_MD5`, `SSL_RSA_WITH_RC4_128_SHA`, and `SSL_RSA_WITH_3DES_EDE_CBC_SHA`. After this script is run, the cipher suites are set in the domain configuration (that is, the `config.xml` file) and the SSL listeners are restarted with the new cipher suite settings.

```
url="t3://localhost:7001"
adminUsername="weblogic"
adminPassword="password"
connect(adminUsername, adminPassword, url)
edit()
server=cmo.lookupServer('myserver')
cd('Servers')
cd('myserver')
startEdit()
cd('SSL')
cd('myserver')
ssl = server.getSSL()
ciphers = ['SSL_RSA_WITH_RC4_128_MD5', 'SSL_RSA_WITH_RC4_128_SHA', 'SSL_RSA_WITH_
3DES_EDE_CBC_SHA']
ssl.setCiphersuites(ciphers)
save()
activate()
disconnect()
exit()
```

12.13.4 Using Debugging with JSSE SSL

As described in Section 12.7, "SSL Debugging", SSL debugging provides more detailed information about the SSL events that occurred during an SSL handshake and other operations.

If you debug SSL when the JSSE-based SSL implementation is enabled, you can use the logging properties listed and described in Table 12–2. However, some properties affect only the SSL calling code and not the JSSE implementation. The JSSE-based SSL implementation has its own logging system, which is activated by the `javax.net.debug` property. The `javax.net.debug` property provides multiple levels of control over the amount of output and can be used independently of WebLogic SSL logging (`ssl.debug`).

See the "Debugging Utilities" section of the *Java Secure Socket Extension (JSSE) Reference Guide*, available at the following URL, for more details about the `javax.net.debug` property:

http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html#Debug

12.14 Using the RSA JSSE Provider in WebLogic Server

RSA JSSE is a third-party JSSE provider that can be statically registered in the JVM if you wish to use it. To install and configure the RSA JSSE provider, complete the following steps:

1. Using the following URL, download and install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files that correspond to the version of your JDK.

   http://www.oracle.com/technetwork/java/javase/downloads/index.html

2. Using a text editor, modify the file *JAVA_HOME*/jre/lib/security/java.security by making the RSA JSSE provider, com.rsa.jsse.JsseProvider, as the first provider in the list.

   For example, before making this update, the list of providers might appear as follows:

```
#
# List of providers and their preference orders (see above):
#
security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
security.provider.4=com.sun.crypto.provider.SunJCE
security.provider.5=sun.security.jgss.SunProvider
security.provider.6=com.sun.security.sasl.Provider
security.provider.7=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.8=sun.security.smartcardio.SunPCSC
security.provider.9=sun.security.mscapi.SunMSCAPI
```

   After you add the RSA JSSE provider, the list might appear as follows:

```
#
# List of providers and their preference orders (see above):
#
security.provider.1=com.rsa.jsse.JsseProvider
security.provider.2=sun.security.provider.Sun
security.provider.3=sun.security.rsa.SunRsaSign
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
security.provider.8=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.9=sun.security.smartcardio.SunPCSC
security.provider.10=sun.security.mscapi.SunMSCAPI
```

   That is, you need to update the sequence number of each subsequent provider. For example, security.provider.**1**=sun.security.provider.Sun is changed to security.provider.**2**=sun.security.provider.Sun (change shown **in bold**).

3. Add the SSL-J jar *WL_HOME*/server/lib/sslj.jar to the Classpath.
4. Restart WebLogic Server for the change to the RSA JSEE provider to take effect.

12.15 X.509 Certificate Revocation Checking

WebLogic Server's JSSE implementation supports X.509 certificate revocation (CR) checking, which checks a certificate's revocation status as part of the SSL certificate path validation process. CR checking improves the security of certificate usage by ensuring that received certificates have not been revoked by the issuing certificate authority.

The following sections describe WebLogic Server support for X.509 certificate revocation checking:

- Section 12.15.1, "Certificate Revocation Checking Overview"
- Section 12.15.2, "Enabling the Default CR Checking Configuration"
- Section 12.15.3, "Choosing the CR Checking Methods to Be Used by WebLogic Server"

12.15.1 Certificate Revocation Checking Overview

In WebLogic Server, CR checking can be used for several purposes, including the following:

- Inbound SSL — validating client certificates
- Outbound SSL — validating server certificates

WebLogic Server's CR checking mechanism includes the following features:

- Support for the following certificate revocation methods:
  - Online Certificate Status Protocol (OCSP)
  - Certificate revocation lists (CRLs)
- You can configure CR checking on a domain-wide basis for all certificate authorities (CAs). And optionally, you can also configure certificate authority overrides for specific CAs.

  A certificate authority override contains changes to the domain-wide CR checking configuration that you want to have in effect for certificates that have been issued by a specific CA. For example, you can configure a particular OCSP responder URL to be used, or require SSL certificate path validation to fail if certificate revocation status cannot be determined. Each certificate authority override you create applies to only one specific CA.

CR checking is disabled by default in WebLogic Server. But using either the WebLogic Server Administration Console or WLST, you can enable CR checking and configure the properties described in the sections that follow.

Note: CR checking is available for a WebLogic Server instance only when JSSE is enabled.

12.15.2 Enabling the Default CR Checking Configuration

In WebLogic Server, CR checking is disabled by default. However, when you enable CR checking, WebLogic Server provides, on a domain-wide basis, a comprehensive set of mechanisms to obtain current revocation status of each certificates it validates. This topic describes the default behavior WebLogic Server provides when you enable CR checking. The subsequent sections explain customizations you can make that can be applied domain-wide or, selectively, to specific certificate authorities.

When the default CR checking configuration is enabled, WebLogic Server automatically does the following when performing SSL certificate path validation:

1. Checks the **OCSP response local cache** to obtain certificate revocation status. The OCSP response local cache is an in-memory cache that holds the latest certificate status that is provided by **OCSP responders**.

   Certificate status in OCSP has a specific validity period. If the certificate status has expired, WebLogic Server does the following:

a. Obtains the **OCSP responder URI** from the certificate. This URI is included in the Authority Information Access (AIA) value in the certificate, which indicates how to access information and services from the issuer of the certificate.

b. Submits an **OCSP request** to the OCSP responder.

   The OCSP responder returns an **OCSP response**, which includes a certificate status of `good`, `revoked`, or `unknown`.

c. Updates the OCSP response local cache with the OCSP response.

For certificates that have a valid, non-expired entry in the OCSP response local cache, WebLogic Server can obtain its revocation status from the cache instead of requesting a fresh OCSP response. This provides improved performance and reduced use of network bandwidth.

Notes: Note the following:

- Cached entries expire based on the OCSP validity period, but the cache behavior can be customized.
- The local OCSP response cache is never used when OCSP nonce is enabled. This ensures the freshest response.

---

2. If the certificate has an OCSP status of `unknown`, WebLogic Server checks the **CRL local cache** for valid CRLs to determine whether the certificate has been revoked. (If either a `revoked` or `not revoked` status is determined by OCSP, CRL is not used for the certificate.)

   By default, the CRL local cache is a file-based store that is maintained on each server instance in a WebLogic domain and that is updated on demand from **CRL distribution points**. A CRL distribution point is a network-accessible server that provides CRLs for download.

   If no valid CRLs are available in the CRL local cache, WebLogic Server does the following:

   a. Obtains the **CRL distribution point URL**, which is included in the CRLDistributionPoints extension in the certificate.

   b. Using the CRL distribution point URL, downloads a fresh CRL and adds it to the cache.

   c. Searches the CRL for an entry that corresponds to the certificate.

   If the certificate serial number is not found in the CRL from the issuer, the certificate status is set to `not revoked`.

Note the following:

- If the certificate has an OCSP status of `revoked`, or is included in a valid CRL, WebLogic Server automatically fails SSL certificate path validation.
- If the revocation status is unknown or cannot be determined after using OCSP and checking the available CRLs, certificate path validation by default is *not* failed.

12.15.2.1 Configuring Default CR Checking

Enabling the default CR checking capability in a WebLogic domain is available through the following MBean attribute:

MBean Attribute	Description	Default Value
`CertRevocMBean.CheckingEnabled`	Specifies whether CR checking is enabled domain-wide.	`False`

For information about how to use the WebLogic Server Administration Console to enable CR checking in a WebLogic domain, see "Enable certificate revocation checking in a domain" in the *Oracle WebLogic Server Administration Console Online Help*.

You can configure a CA override for this MBean attribute. For information, see Section 12.15.7, "Configuring Certificate Authority Overrides".

12.15.2.2 Customizing the CR Checking Configuration

The default CR checking behavior in WebLogic Server is appropriate for deployment environments in which CR checking is desired, but not required. Depending on your environment, you might require CR checking, or need to enforce behaviors that are specific to particular certificate authorities. Table 12–4 lists and summarizes the types of customizations you can make to CR checking in WebLogic Server and provides links to the sections in which they are explained.

Table 12–4 Customizations You Can Make to the CR Checking Configuration

Customization	Description
CR checking method order	Specifies the order in which the supported CR checking methods are used; that is, OCSP and CRLs. Optionally, you can choose to use only OCSP, or only CRLs. See Section 12.15.3, "Choosing the CR Checking Methods to Be Used by WebLogic Server".
Require certificate revocation status	Specifies that SSL certificate path validation must fail if a certificate's revocation status is unknown or cannot be determined. See Section 12.15.4, "Failing SSL Certificate Path Validation if Revocation Status Cannot Be Determined".
Domain-wide OCSP settings	Customize, domain-wide, one or more of the following OCSP features or behaviors: ■ Use of nonces in OCSP requests and responses ■ OCSP response cache. For example, capacity or refresh period ■ OCSP response timeout interval settings For information, see Section 12.15.5, "Using the Online Certificate Status Protocol".
Domain-wide CRL protocol settings	Customize, domain-wide, one or more of the following CRL features or behaviors: ■ Use of CRL distribution points ■ CRL cache refresh frequency ■ CRL distribution point download timeout interval settings For information, see Section 12.15.6, "Using Certificate Revocation Lists".

Table 12–4 (Cont.) Customizations You Can Make to the CR Checking Configuration

Customization	Description
Certificate authority overrides	Customize the CR checking behavior for certificates issued by a particular CA. For example: ■ Disable revocation checking for those certificates ■ Change the CR checking method order ■ Automatically fail certificate path validation if revocation status is unknown or unavailable ■ Customize OCSP or CRL settings (except for the CRL local cache settings) ■ Designate the OCSP responder URL to use ■ Designate the CRL distribution point URL to use A certificate authority override always takes precedence over domain-wide settings that are in place. For more information, see Section 12.15.7, "Configuring Certificate Authority Overrides".

12.15.3 Choosing the CR Checking Methods to Be Used by WebLogic Server

By default, when checking a certificate's revocation status, WebLogic Server first uses OCSP. If OCSP returns the certificate's status as "unknown," WebLogic Server then uses CRLs. However, you can change the CR checking method used, or the sequence in which the methods are used, to one of the following:

- OCSP only
- CRLs only
- OCSP then CRLs — If the OCSP status for a certificate is returned as `unknown`, CRLs are checked for certificate status.
- CRLs then OCSP — If a certificate's revocation status cannot be determined by checking available CRLs, its OCSP status is checked.

Configuring the CR checking method and order in a WebLogic domain is available through the following MBean attribute:

MBean Attribute	Description	Default Value
`CertRevocMBean.MethodOrder`	Specifies the domain-wide CR checking method.	`OCSP_THEN_CRL`

You can configure a CA override for this MBean attribute. For information, see Section 12.15.7, "Configuring Certificate Authority Overrides".

For information about how to use the WebLogic Server Administration Console to configure the CR checking method and order for a WebLogic domain, see "Enable certificate revocation checking in a domain" in the *Oracle WebLogic Server Administration Console Online Help*.

12.15.4 Failing SSL Certificate Path Validation if Revocation Status Cannot Be Determined

By default, if an X.509 certificate's revocation status cannot be determined by any of the selected checking methods, the certificate can still be accepted if the SSL certificate path validation is otherwise successful. However, for certificates whose revocation

status cannot be determined, you can optionally configure WebLogic Server to fail certificate path validation.

Configuring a WebLogic domain to fail SSL certificate path validation when the revocation status cannot be determined is available through the following MBean attribute:

MBean Attribute	Description	Default Value
`CertRevocMBean.FailOnUnknownRevocStatus`	Specifies on a domain-wide basis whether a certificate's path validation should fail if its revocation status cannot be determined.	`False`

You can configure a CA override for this MBean attribute. For information, see Section 12.15.7, "Configuring Certificate Authority Overrides".

For information about how to configure this MBean attribute using the WebLogic Server Administration Console, see "Enable certificate revocation checking in a domain" in the *Oracle WebLogic Server Administration Console Online Help*.

12.15.5 Using the Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP) is an automated certificate checking network protocol that is defined in RFC 2560. As part of certificate validation, WebLogic Server queries the revocation status of a certificate by issuing an **OCSP request** to an **OCSP responder**. Certificate status is maintained by the OCSP responder. Acceptance of the certificate is suspended until the responder returns an **OCSP response**, indicating whether the certificate is still trusted by the CA that issued it.

OCSP may be used to satisfy some of the operational requirements of providing more timely revocation information than is possible with CRLs and may also be used to obtain additional status information. For more information about OCSP, see the description of RFC 2560 at `http://www.ietf.org/rfc/rfc2560.txt`.

The following sections describe how to configure OCSP in WebLogic Server:

- Section 12.15.5.1, "Using Nonces in OCSP Requests"
- Section 12.15.5.2, "Setting the Response Timeout Interval"
- Section 12.15.5.3, "Enabling and Configuring the OCSP Response Local Cache"

12.15.5.1 Using Nonces in OCSP Requests

A nonce is a random number that, when included in an OCSP request, forces a fresh response; pre-signed responses are rejected. The use of nonces can prevent replay attacks. By default, WebLogic Server does *not* include nonces in OCSP requests.

However, when WebLogic Server is configured to use nonces in OCSP:

1. WebLogic Server generates a nonce for each OCSP request, and includes it in an extension in the request.
2. The signed OCSP response must include the same nonce, which is included in an extension in the response.

You can configure the use of OCSP nonces in a WebLogic domain using the following MBean attribute:

MBean Attribute	Description	Default Value
`CertRevocMBean.OcspNonceEnabled`	Specifies whether nonces are generated for OCSP requests. This setting is domain-wide.	`false`

You can also configure CA overrides for this MBean attribute. For information, see Section 12.15.7.2, "Configuring OCSP Properties in a Certificate Authority Override".

For information about how to use the WebLogic Server Administration Console to configure OCSP nonces, see "Configure domain-wide OCSP settings" in the *Oracle WebLogic Server Administration Console Online Help*.

12.15.5.2 Setting the Response Timeout Interval

The response timeout interval limits the wait time for OCSP responses. Setting a timeout interval helps minimize blocked threads and also reduces the system's vulnerability to denial of service attacks. In addition to setting a response timeout interval, you can configure a time tolerance value for handling clock-skew differences between WebLogic Server and OCSP responders.

The default response timeout interval is 10 seconds, with a zero time tolerance. The response timeout interval and time tolerance value can be set domain-wide and, optionally, set specific to one or more CAs.

You can configure the OCSP response timeout interval and time tolerance value for a WebLogic domain using the following MBean attributes:

MBean Attribute	Description	Default Value
`CertRevocMBean.OcspResponseTimeout`	Specifies the domain-wide timeout interval, in seconds, for OCSP responses. The valid range is between 1 and 300, inclusive.	`10`
`CertRevocMBean.OcspTimeTolerance`	Specifies the domain-wide OCSP time tolerance value, in seconds, for OCSP responses.	`0`

You can also configure CA overrides for these MBean attributes. For information, see Section 12.15.7.2, "Configuring OCSP Properties in a Certificate Authority Override".

For information about how to use the WebLogic Server Administration Console to configure OCSP response timeout interval and time tolerance values, see "Configure domain-wide OCSP settings" in the *Oracle WebLogic Server Administration Console Online Help*.

12.15.5.3 Enabling and Configuring the OCSP Response Local Cache

To optimize performance and reduce network bandwidth, WebLogic Server's OCSP implementation is configured by default to use a local in-memory cache for holding OCSP responses, called the **OCSP response local cache**. Cached entries automatically expire based on the OCSP validity period and other criteria, such as entries least accessed. If nonces are enabled, OCSP responses obtained using a nonce are not cached. This ensures the freshest response is always used with nonces.

You can configure the OCSP response local cache in a WebLogic domain using the following MBean attributes:

MBean Attribute	Description	Default Value
`CertRevocMBean.OcspResponseCache Enabled`	Specifies whether the OCSP response local cache is enabled domain-wide.	`true`
`CertRevocMBean.OcspResponseCache Capacity`	Specifies the maximum number of entries supported by the OCSP response local cache.	`1024`
`CertRevocMBean.OcspResponseCache RefreshPeriodPercent`	Specifies the refresh period for the OCSP response local cache, expressed as a percentage of the validity period of the response. For example, for a validity period of 10 hours, a value of 10% specifies that after one hour, the cached response expires and a fresh response is required.	`100`

You can also configure CA overrides for this MBean attribute. For information, see Section 12.15.7.2, "Configuring OCSP Properties in a Certificate Authority Override".

For information about how to use the WebLogic Server Administration Console to configure the OCSP response local cache, see "Configure domain-wide OCSP settings" in the *Oracle WebLogic Server Administration Console Online Help*.

12.15.6 Using Certificate Revocation Lists

A certificate revocation list (CRL) is a time-stamped list of digital certificates that have been revoked by the certificate authority (CA) that issued them. Each CRL is signed by a CA and made available in a public repository.

The CRL implementation in WebLogic Server includes support for the following:

- CRL local cache, which enables efficient access for CR checking.
- Automatic import of user supplied CRL files into the CRL cache.
- Use of distribution points from which the CRL cache can optionally be populated and refreshed.

The following sections explain how to configure CRL usage in WebLogic Server:

- Enabling Updates from Distribution Points
- Configuring the CRL Local Cache

12.15.6.1 Enabling Updates from Distribution Points

Updating CRLs from distribution points is enabled by default. If the appropriate CRL for a certificate being validated does not already exist in the local cache, the CRL is downloaded from an available distribution point.

WebLogic Server also allows you to configure a timeout interval for the CRL download from a distribution point. This timeout interval limits the wait time for CRL downloads, and also minimizes the risk of blocked threads and vulnerability to denial of service attacks. Note that if the CRL download times out, the CRL method reports that the revocation status is unknown; however, the CRL download continues in a separate thread until complete and the CRL becomes available for future CRL checking.

You can configure CRL distribution points for a WebLogic domain using the following MBean attributes:

MBean Attribute	Description	Default Value
`CertRevocMBean.CrlDpEnabled`	Specifies whether CRL distribution points are enabled domain-wide.	`true`
`CertRevocMBean.CrlDpDownloadTimeout`	Specifies the overall timeout interval, domain-wide, for the distribution point CRL download, expressed in seconds. The valid range is between 1 and 300, inclusive.	`10`

You can also configure CA overrides for these MBean attributes. For information, see Section 12.15.7.3, "Configuring CRL Properties in a Certificate Authority Override".

For information about how to use the WebLogic Server Administration Console to configure CRL distribution points for a WebLogic domain, see "Configure domain-wide CRL settings" in the *Oracle WebLogic Server Administration Console Online Help*.

12.15.6.2 Configuring the CRL Local Cache

The CRL local cache is automatically enabled in WebLogic Server. Because obtaining CRLs is a time-consuming process, CRLs can be stored, while valid, in local files. In addition, WebLogic Server allows you to configure the refresh interval for the local cache, expressed as a percentage of the validity period of the CRL.

You may supply CRL files to be used by copying them into the following CRL import directory, where *server-name* represents the name of the WebLogic Server instance:

WL_HOME/servers/*server-name*/security/certrevocation/crlcache/import

The CRL files are automatically imported and internally cached. This directory is automatically created, if it does not already exist, when CR checking is enabled and an SSL connection is attempted.

Notes: Note the following:

- After WebLogic Server is started, the import of the CRL file starts automatically when CR checking is enabled and at least one attempt to check a certificate's revocation status has occurred. This minimizes resource usage until necessary.
- After you import CRL files, they are automatically deleted from the import directory.
- The CRL local cache configuration settings are domain-wide. You cannot configure a certificate authority override for the CRL local cache.

You can configure the CRL local cache for a WebLogic domain using the following MBean attributes:

MBean Attribute	Description	Default Value
`CertRevocMBean.CrlCacheRefreshPeriodPercent`	Specifies the refresh period for the CRL local cache, expressed as a percentage of the validity period of the CRL.	`100`

For information about how to use the WebLogic Server Administration Console to configure the CRL local cache for a WebLogic domain, see "Configure domain-wide CRL settings" in the *Oracle WebLogic Server Administration Console Online Help*.

12.15.7 Configuring Certificate Authority Overrides

Configuring certificate authority overrides allows you to specify CR checking behavior that is enforced for certificates issued by a particular CA. A certificate authority override always supersedes the domain-wide CR checking configuration that is enabled. The following sections explain how to configure CR checking CA overrides:

- Section 12.15.7.1, "General Certificate Authority Overrides"
- Section 12.15.7.2, "Configuring OCSP Properties in a Certificate Authority Override"
- Section 12.15.7.3, "Configuring CRL Properties in a Certificate Authority Override"

12.15.7.1 General Certificate Authority Overrides

To create a certificate authority override for a specific CA, complete the following steps:

1. Identify the CA by its distinguished name. This must be the complete issuer distinguished name (defined in RFC 2253) of the certificates for which this override applies.

   For example, the distinguished name of the WebLogic Server DemoTrust CA is `CN=CertGenCAB, OU=FOR TESTING ONLY, O=MyOrganization, L=MyTown, ST=MyState, C=US`.

2. Specify whether CR checking is enabled for certificates issued by this CA, if necessary.
3. Specify the CR checking methods and order performed for certificates issued by this CA.
4. Specify whether SSL certificate path validation should fail if the revocation status of certificates issued by this CA cannot be determined.
5. Optionally, specify additional OCSP or CRL customizations, as explained in the following sections:
   - Configuring OCSP Properties in a Certificate Authority Override
   - Configuring CRL Properties in a Certificate Authority Override

You can configure general certificate authority overrides for a CA by using the following MBean attributes:

MBean Attribute	Description	Default Value
`CertRevocCaMBean.DistinguishedName`	Specifies the distinguished name (DN) of the CA subject.	None (required field)
`CertRevocCaMBean.CheckingDisabled`	For this CA, specifies whether CR checking is disabled.	`false`
`CertRevocCaMBean.FailOnUnknownRevocStatus`	For this CA, specifies whether SSL certificate path checking should fail if the certificate revocation status cannot be determined from any of the available methods.	Same as current setting of `CertRevocMBean.FailOnUnknownRevocStatus`.

MBean Attribute	Description	Default Value
`CertRevocCaMBean.MethodOrder`	Specifies the certificate revocation checking method order when checking certificates issued by this CA.	Same as current setting of `CertRevocMBean.MethodOrder`.

For information about how to use the WebLogic Server Administration Console to configure certificate authority overrides, see "Configure certificate authority overrides" in the *Oracle WebLogic Server Administration Console Online Help*

12.15.7.2 Configuring OCSP Properties in a Certificate Authority Override

WebLogic Server tries the following trust models in its OCSP implementation:

- Delegated Trust Model (DTM) — The OCSP response is signed by an OCSP responder that has been delegated by the CA to sign responses on its behalf.
- Explicit Trust Model (ETM) — If neither the CA nor an authority to which OCSP responsibilities have been delegated has signed the OCSP response, an explicitly trusted signer may be specified. ETM is used when you can supply an additional trusted certificate that may be used to verify the OCSP response signature. This can be any certificate, including one unrelated to the CA corresponding to the override. ETM may be used for OCSP responders which are trusted, but are not authorized to sign OCSP responses on behalf of issuers. Explicitly trusted public certificates for OCSP responders may be suitable if the OCSP server is internally maintained within your enterprise.
- CA-signed Trust Model — The OCSP response is presumed to be signed by the same CA that issued the certificate for which the revocation status is being requested.

When you create a certificate authority override, WebLogic Server allows you to configure the OCSP properties that are described in Table 12–5. This table also identifies the MBean attributes you can use to configure these override properties.

Table 12–5 OCSP Properties That Can Be Specified in a Certificate Authority Override

Override	Description	MBean Attribute
OCSP responder URL	Specifies the URL to be used for either: ■ Failover, if the OCSP responder URI from the certificate AIA value is not available or not acceptable ■ Override, to be always used as the responder URL instead of the responder URI from the certificate AIA. For more information, see Section 12.15.7.2.1, "Identifying the OCSP Responder URL".	`CertRevocCaMBean.OcspResponderUrl` The default value is none.
How the OCSP responder URL is used	Specifies how the OCSP responder URL is to be used: for failover or override.	`CertRevocCaMBean.OcspResponderUrlUsage` The default value is `FAILOVER`.
OCSP responder certificate subject name	For this CA, specifies the explicitly trusted OCSP responder certificate subject name. For example, `CN=OCSP Responder, O=XYZ Corp`. This must correspond to the subject distinguished name of a certificate in the configured WebLogic Server trust keystore. In cases where the subject name alone is not sufficient to uniquely identify the certificate, both the `CertRevocCaMBean.OcspResponderCertIssuerName` and `CertRevocCaMBean.OcspResponderCertSerialNumber` are used instead.	`CertRevocCaMBean.OcspResponderCertSubjectName` The default value is `NONE`.
OCSP responder certificate issuer name	For this CA, specifies the explicitly trusted OCSP responder certificate issuer name. For example, `CN=Enterprise CA, O=XYZ Corp`. This must correspond to the issuer distinguished name of a certificate in the configured WebLogic Server trust keystore. When this attribute is set, the `CertRevocCaMBean.OcspResponderCertSerialNumber` must also be set.	`CertRevocCaMBean.OcspResponderCertIssuerName` The default value is `NONE`.
OCSP responder certificate serial number	For this CA, specifies the explicitly trusted OCSP responder certificate serial number. For example, `2A:FF:00`. This must correspond to the serial number of a certificate in the configured WebLogic Server trust keystore. When this attribute is set, the `CertRevocCaMBean.OcspResponderCertIssuerName` attribute must also be set.	`CertRevocCaMBean.OcspResponderCertSerialNumber` The default value is `NONE`.

Table 12–5 (Cont.) OCSP Properties That Can Be Specified in a Certificate Authority Override

Override	Description	MBean Attribute
OCSP responder Explicit Trust Method	For this CA, specifies whether the OCSP Explicit Trust model is enabled and how a trusted certificate in the Weblogic Server trust keystore is specified. The following values can be specified: ■ `NONE` specifies that Explicit Trust is disabled. ■ `USE_SUBJECT` specifies that the trusted certificate is identified using the subject DN that is specified in the `CertRevocCaMBean.OcspResponderCertSubjectName` attribute. ■ `USE_ISSUER_SERIAL_NUMBER` specifies that the trusted certificate is identified using the issuer DN and certificate serial number that are specified in the `CertRevocCaMBean.OcspResponderCertIssuerName` and `CertRevocCaMBean.OcspResponderCertSerialNumber` attributes, respectively.	`CertRevocCaMBean.OcspResponderExplicitTrustMethod` The default value is `NONE`.
Nonce enabled	For this CA, specifies whether nonces are sent with OCSP requests, which forces a fresh (not pre-signed) response.	`CertRevocCaMBean.OcspNonceEnabled` The default value is the same as the current setting for `CertRevocMBean.OcspNonceEnabled`.
OCSP response local cache	For this CA, specifies whether the OCSP response local cache is enabled.	`CertRevocCaMBean.OcspResponseCacheEnabled` The default value is the same as the current setting for `CertRevocMBean.OcspResponseCacheEnabled`.
OCSP response timeout	For this CA, specifies the timeout interval for the OCSP response, expressed in seconds. The valid range is between 1 and 300, inclusive. For more information, see Section 12.15.5.2, "Setting the Response Timeout Interval".	`CertRevocCaMBean.OcspResponseTimeout` The default value is the same as the current setting for `CertRevocMBean.OcspResponseTimeout`.
OCSP time tolerance	For this CA, specifies the time tolerance value for handling clock-skew differences between WebLogic Server and responders, expressed in seconds. The valid range is between 0 and 900, inclusive. The validity period of the response is extended both into the future and into the past by the specified amount of time, effectively widening the validity interval.	`CertRevocCaMBean.OcspTimeTolerance` The default value is the same as the current setting for `CertRevocMBean.OcspTimeTolerance`.

For information about how use the WebLogic Server Administration Console to configure OCSP settings in a certificate authority override, see "Configure certificate authority overrides" in the *Oracle WebLogic Server Administration Console Online Help*.

12.15.7.2.1 Identifying the OCSP Responder URL To validate a certificate using an OCSP responder lookup, WebLogic Server uses the following methods to determine the OCSP responder URL:

- Authority Information Access (AIA) value in the certificate, which indicates how to access information and services for the issuer of the certificate. For example, the AIA contains the URI for the OCSP responder.
- Default OCSP responder failover or override — If the OCSP responder URI is not available from the certificate AIA value, or is not acceptable, a default OCSP responder URL can be configured on a per-CA basis.

  Additionally, the default OCSP responder URL per CA can be specified selectively for either failover, or for override. When specified for override, this URL always overrides the value obtained from the certificate AIA extension.

For information about how to use the WebLogic Server Administration Console to set the OCSP responder URL in a certificate authority override, see "Configure certificate authority overrides" in the *Oracle WebLogic Server Administration Console Online Help*.

12.15.7.3 Configuring CRL Properties in a Certificate Authority Override

When you configure a certificate authority override, WebLogic Server allows you to configure the CRL properties listed and described in Table 12–6. This table also identifies the MBean attributes you can use to configure these properties.

Table 12–6 CRL Properties That Can Be Specified in a Certificate Authority Override

Override	Description	MBean Attribute
Use of distribution point to update local CRL cache	For this CA, specifies whether CRL distribution point processing to update the local CRL cache is enabled.	`CertRevocCaMBean.CrlDpEnabled` The default value is the same as the current setting for `CertRevocMBean.CrlDpEnabled`.
Distribution point URL	For this CA, specifies the CRL distribution point URL to be used for either: ■ Failover, if the URL from the CRLDistributionPoints extension in the certificate is unavailable ■ Override, to be always used as the CRL distribution point URL instead of the CRLDistributionPoints extension in the certificate	`CertRevocCaMBean.CrlDpUrl` The default value is `null`.
How the distribution point URL is used	Specifies how the distribution point URL is to be used: for failover or override.	`CertRevocCaMBean.CrlDpUrlUsage` The default value is `FAILOVER`.
Distribution point CRL download timeout	For this CA, specifies the overall timeout interval for the distribution point CRL download, expressed in seconds. The valid range is between 1 and 300, inclusive.	`CertRevocCaMBean.CrlDpDownloadTimeout` The default value is the same as the current setting for `CertRevocMBean.CrlDpDownloadTimeout`.

For information about how to use the WebLogic Server Administration Console to customize the CRL settings in a certificate authority override, see "Configure certificate authority overrides" in the *Oracle WebLogic Server Administration Console Online Help*.

13

Configuring Oracle OPSS Keystore Service

This chapter describes how to configure the Oracle OPSS Keystore Service for use with WebLogic Server.

Chapter 11, "Configuring Identity and Trust" describes how to configure identity and trust for WebLogic Server with the default JKS keystore type.

As described in "Managing Keys and Certificates with the Keystore Service" in *Application Security Guide*, the OPSS Keystore Service provides an alternate mechanism to manage keys and certificates for message security. The OPSS Keystore Service makes using certificates and keys easier by providing central management and storage of keys and certificates for all servers in a domain. You use the OPSS Keystore Service to create and maintain keystores of type KSS.

This section includes the following sections:

- Prerequisites for Using the OPSS Keystore Service
- Where is the OPSS Keystore Service Documented?
- Configuring the OPSS Keystore Service for Demo Identity and Trust: Main Steps
- Configuring the OPSS Keystore Service for Custom Identity and Trust: Main Steps

This section assumes that you are familiar with a basic overview of the OPSS Keystore Service, as described in "Managing Keys and Certificates with the Keystore Service".

13.1 Prerequisites for Using the OPSS Keystore Service

You can use the OPSS Keystore Service with WebLogic Server only if you have installed the Oracle JRF template on the WebLogic Server system and used this template to create the domain.

The OPSS Keystore Service is available only with the JRF template and is not available with the default WebLogic Server configuration.

13.2 Where is the OPSS Keystore Service Documented?

The OPSS Keystore Service is documented in "Managing Keys and Certificates with the Keystore Service" in *Application Security Guide*. In particular, "Managing Keys and Certificates with the Keystore Service" describes how you create the KSS keystore, how to manage it, and what tools and commands are available.

This section briefly summarizes the steps you follow to configure the OPSS Keystore Service, but "Managing Keys and Certificates with the Keystore Service" in *Application Security Guide* is the definitive source.

13.3 Configuring the OPSS Keystore Service for Demo Identity and Trust: Main Steps

You can perform the OPSS Keystore Service operations using either Fusion Middleware Control or the Keystore Service commands with WLST. This section demonstrates the Fusion Middleware Control steps, but "Managing Keys and Certificates with the Keystore Service" describes both options.

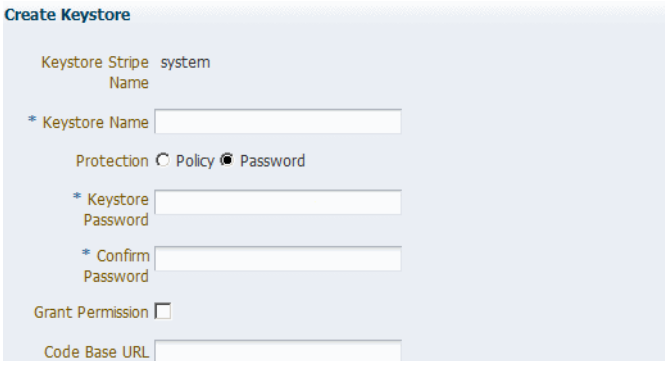
You must configure the OPSS Keystore Service before you can use it for demo identity and trust with WebLogic Server.

Perform the following steps to configure an OPSS Keystore Service for demo identity and trust:

1. Launch Fusion Middleware Control.
2. From the **WebLogic Domain** menu, select **Security** then **Keystore**.
3. Create a keystore named `demoidentity` in the `system` stripe. (See "Creating a Keystore with Fusion Middleware Control" for more information.)
   a. Select the `system` stripe and click **Create Keystore**.

      The Create Keystore page is shown in Figure 13–1.

Figure 13–1 Create Keystore

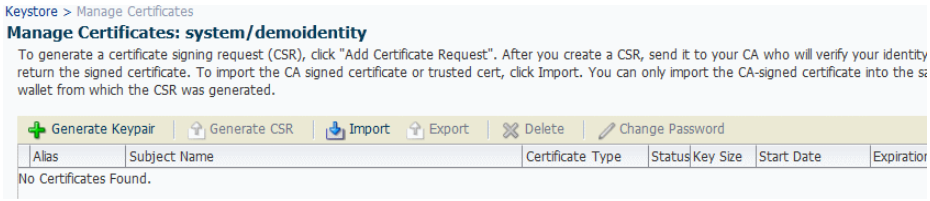


   b. Name this keystore `demoidentity`.
   c. Set the protection type to Password.
   d. Set the password to `DemoIdentityKeyStorePassPhrase`, and confirm.
   e. Uncheck the Grant Permission check box.
   f. Do not specify a code base URL.
4. Select the `demoidentity` keystore you just created and click **Manage**.

   Enter the `DemoIdentityKeyStorePassPhrase` password.

   The Manage Certificates screen shown in Figure 13–2 appears.

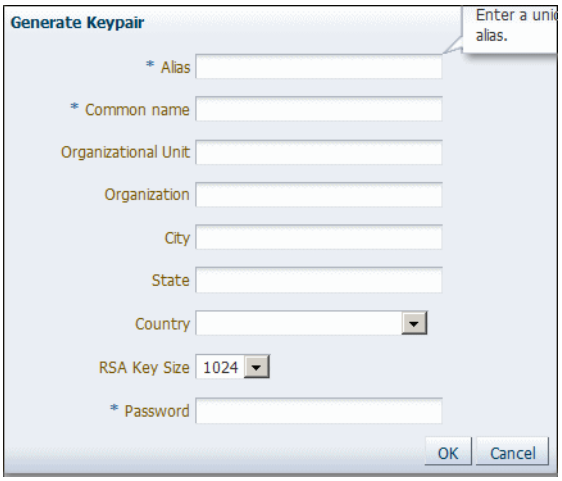
Figure 13–2 Manage Certificates



5. Click **Generate Keypair** to generate a private/public key pair.

   The Generate Keypair screen is shown in Figure 13–3.

Figure 13–3 Generate Keypair



   a. Specify `DemoIdentity` as the alias for the key pair.

   b. Specify the Common Name as `DemoCertFor_<WLS Domain Name>`, where `DemoCertFor_` is a required constant and `<WLS Domain Name>` is the WebLogic Server domain name. For Example: `DemoCertFor_base_domain`.

   Note: The WebLogic Server DefaultHostnameVerifier has been modified to accept this non-standard hostname format when you set the "Use KSS For Demo" flag in the security configuration for the Weblogic Server domain. Other hostname verifiers may not support this format.

   c. Specify other site-specific information as appropriate.

   d. You can accept the default RSA key size if appropriate for your environment. Oracle requires a key length of 1024 bits or larger.

   e. Specify the password as `DemoIdentityPassPhrase`.

   f. Click **OK**.

6. From the WebLogic Server Administration Console, navigate to the Domain -> Security -> Advanced page, and enable the "Use KSS For Demo" check box.

7. Configure the WebLogic Server instance to use Demo Identity and Demo Trust, as described in Configure keystores.

8. Configure SSL for the WebLogic Server instance, as described in Set Up SSL.

Remember that the WebLogic Server DefaultHostnameVerifier has been modified to accept the non-standard `DemoCertFor_<WLS Domain Name>` hostname format. Other hostname verifiers may not support this format.

9. Restart WebLogic Server.

13.4 Configuring the OPSS Keystore Service for Custom Identity and Trust: Main Steps

You must configure the OPSS Keystore Service before you can use it for custom identity and trust with WebLogic Server.

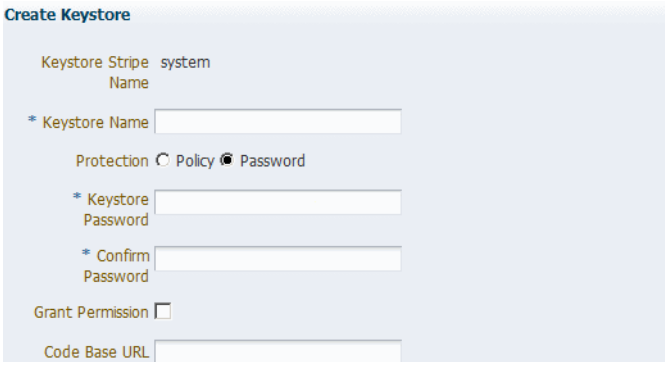
You can perform the OPSS Keystore Service operations using either Fusion Middleware Control or the Keystore Service commands with WLST. This section demonstrates the Fusion Middleware Control steps, but "Managing Keys and Certificates with the Keystore Service" describes both options.

Perform the following steps to configure an OPSS Keystore Service for custom identity and trust:

1. Launch Fusion Middleware Control.
2. From the **WebLogic Domain** menu, select **Security** then **Keystore**.
3. Create a keystore in the `system` stripe. (See "Creating a Keystore with Fusion Middleware Control" for more information.)
   a. Select the `system` stripe and click **Create Keystore**.

      The Create Keystore page is shown in Figure 13–1.

Figure 13–4 Create Keystore

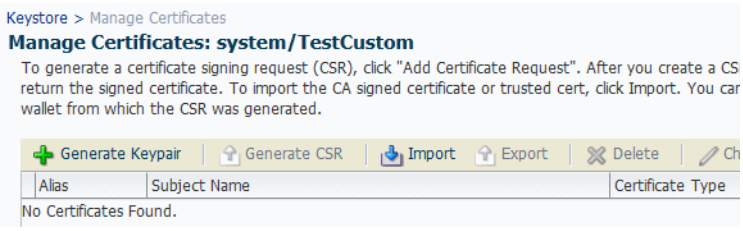


   b. Name this keystore.
   c. Set the protection type to Password.
   d. Set the password.
   e. Uncheck the Grant Permission check box.
   f. Do not specify a code base URL.
4. Select the keystore you just created and click **Manage**.

   Enter the password.

   The Manage Certificates screen shown in Figure 13–2 appears.

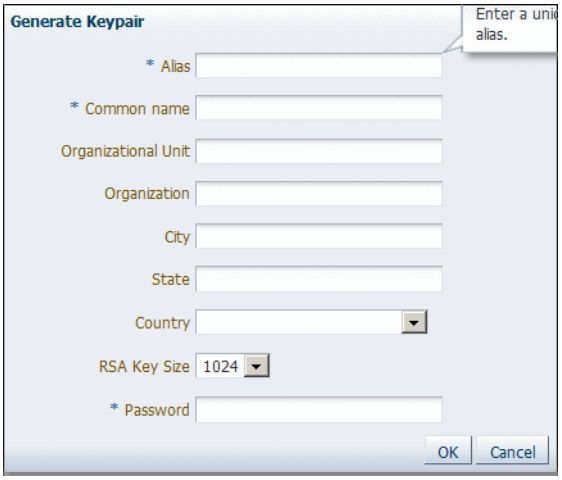
Figure 13–5 Manage Certificates



5. Click **Generate Keypair** to generate a private/public key pair.

   The Generate Keypair screen is shown in Figure 13–3.

Figure 13–6 Generate Keypair



   a. Specify the alias for the key pair.

   b. Specify site-specific information as appropriate.

   c. You can accept the default RSA key size if appropriate for your environment. Oracle requires a key length of 1024 bits or larger.

   d. Specify the password.

   e. Click **OK**.

6. You have the option to use this KSS Demo CA-signed key pair as-is, or to obtain a signed certificate from a reputable vendor such as Entrust, Verisign, and so forth.

   To obtain the signed certificate from a reputable vendor, select the alias for the key pair and click **Generate CSR**. After you create a CSR, send it to your CA, which will authenticate the certificate request and create a digital certificate based on the request.

   See "Importing a Certificate or Trusted Certificate with Fusion Middleware Control" in *Application Security Guide* for instructions on how to import the CA-signed certificate.

7. If you do not use the preconfigured OPSS Keystore Service trust store `kss://system/trust`, you must create your own.

Note: Oracle recommends you use the preconfigured OPSS Keystore Service trust store.

To create your own trust store, create another OPSS Keystore Service keystore, and import trusted certificates. See "Importing a Certificate or Trusted Certificate with Fusion Middleware Control" in *Application Security Guide* for instructions on how to import trusted certificates.

8. Configure the WebLogic Server instance to use KSS for Custom Identity and Trust, as described in Configure keystores. You specify the fully-qualified path to the keystore as the URI in the form `kss://system/`*`keystore-name`*. The keystore type is KSS.
9. Configure SSL for the WebLogic Server instance, as described in Set Up SSL.

All the server SSL attributes are dynamic; when modified via the Console, they cause the corresponding SSL server or channel SSL server to restart and use the new settings for new connections. Old connections will continue to run with the old configuration. To ensure that all the SSL connections exist according to the specified configuration, you must reboot WebLogic Server.

14

Configuring Security for a WebLogic Domain

This chapter describes how to set security configuration options for a WebLogic domain.

This chapter includes the following sections:

- Important Information Regarding Cross-Domain Security Support
- Enabling Trust Between WebLogic Server Domains
- Using Connection Filters
- Using the Java Authorization Contract for Containers
- Viewing MBean Attributes
- How Passwords Are Protected in WebLogic Server
- Protecting User Accounts
- Configuring a Domain to Use JAAS Authorization

> **Note:** These sections apply to WebLogic Server deployments using the security features in this release of WebLogic Server as well as deployments using Compatibility Security.

14.1 Important Information Regarding Cross-Domain Security Support

This section describes important information regarding support for the cross-domain security solution.

As described in Section 14.2, "Enabling Trust Between WebLogic Server Domains", cross-domain security establishes trust between domains such that principals in a subject from one WebLogic domain can make calls in another domain. WebLogic Server establishes a security role for cross-domain users, and uses the WebLogic Credential Mapping security provider in each domain to store the credentials to be used by the cross-domain users.

In this release of WebLogic Server, subsystems such as JMS, JTA, MDB, and WAN replication implement cross-domain security. These subsystems can authenticate and send the required credentials across domains. However, the EJB container does not implement the solution for cross-domain security.

14.2 Enabling Trust Between WebLogic Server Domains

Trust between domains is established so that principals in a Subject from one WebLogic domain can make calls in another domain. In previous releases of WebLogic

Server, there was only one type of domain trust that is now referred to as Global Trust. WebLogic Server now supports a type of domain trust that is referred to as Cross Domain Security. The following sections explain how to configure each domain trust type:

- Section 14.2.1, "Enabling Cross Domain Security Between WebLogic Server Domains"
- Section 14.2.2, "Enabling Global Trust"

14.2.1 Enabling Cross Domain Security Between WebLogic Server Domains

> **Note:** Please see Section 14.1, "Important Information Regarding Cross-Domain Security Support" before enabling cross domain security.

Cross Domain Security establishes trust between two WebLogic domain pairs by using a credential mapper to configure communication between these WebLogic domains. Configuration and use of cross-domain security is described in the following sections:

- Section 14.2.1.1, "Configuring Cross-Domain Security"
- Section 14.2.1.2, "Configuring a Cross-Domain User"
- Section 14.2.1.3, "Configure a Credential Mapping for Cross-Domain Security"

In addition to the approach that uses a Credential Mapping security provider for cross-domain security, WebLogic Server also enables a different approach, under which global trust is established between two or more domains by using the same domain credential in each domain. If you enable global trust between two or more domains, the trust relationship is transitive and symmetric. In other words, if Domain A trusts Domain B and Domain B trusts Domain C, then Domain A will also trust Domain C and Domain B and Domain C will both trust Domain A. In most cases, the Cross Domain Security approach is preferable to the global trust approach, because its use of a specific user group and role for cross-domain actions allows for finer grained security. For information about the global trust approach in WebLogic Server, see Section 14.2.2, "Enabling Global Trust".

14.2.1.1 Configuring Cross-Domain Security

To configure cross-domain security in a WebLogic domain, set the `SecurityConfigurationMBean.CrossDomainSecurityEnabled` attribute to `true`. To do this in the WebLogic Server Administration Console:

1. Click the name of the domain in the Domain Configurations section of the Home page.
2. Select **Security** > **General**.
3. Check **Cross Domain Security Enabled**.

If you maintain any WebLogic domains that have not enabled cross-domain security, you need to add their domain names to the list of excluded domains, in the `SecurityConfigurationMBean.ExcludedDomainNames` attributes. To do this in the WebLogic Server Administration Console:

1. Click the name of the domain in the Domain Configuration section of the Home page.
2. Select **Security** > **General**.

3. In the **Excluded Domain Names** field, enter the names of any domains that do not have cross-domain security enabled. Enter the names of these domains separated either by semicolons or line breaks.

14.2.1.2 Configuring a Cross-Domain User

Cross-domain security in WebLogic Server uses a global security role named `CrossDomainConnector` with resource type remote and a group named `CrossDomainConnectors`, which is assigned the `CrossDomainConnector` role. Invocation requests from remote domains are expected to be from users with the `CrossDomainConnector` role. By default, the `CrossDomainConnectors` group has no users as members. You need to create one or more users and add them to the group `CrossDomainConnectors`. Typically, such a user will be a virtual system user and preferably should have no privileges other than those granted by the `CrossDomainConnector` security role.

14.2.1.3 Configure a Credential Mapping for Cross-Domain Security

> **Note:** The Credential Mapper identifies domains by their names. Therefore, it is important that the domains involved have unique names.

In each WebLogic domain, you need to specify a credential to be used by each user on each remote domain that needs to be trusted. Do this by configuring credential mappings for each domain in the connection. Each credential mapping needs to specify:

- The resource protocol, which is named `cross-domain-protocol`
- The name of the remote domain that needs to interact with the local domain
- The name of the user in the remote domain that will be authorized to interact with the local domain
- The password of the user in the remote domain that will be authorized to interact with the local domain

To configure a cross-domain security credential mapping in the WebLogic Server Administration Console, click **Security Realms** in the left panel.

1. Click the name of your security realm (default is `myrealm`).
2. Select **Credential Mappings** > **Default**, and click **New**.
3. On the Creating the Remote Resource for the Security Credential Mapping page:
   - Select **Use cross-domain protocol**.
   - In the **Remote Domain** field, enter the name of the remote domain that needs to interact with the local domain.
4. Click **Next**.
5. On the Create a New Security Credential Map Entry page, enter the following:
   - **Local User**: `cross-domain`
   - **Remote User**: User configured in the Remote Domain that is authorized to interact with the Local Domain.
   - **Password**: The password for the Remote User.

6. Click **Finish**.

See "Create a Cross-Domain Security Credential Mapping" in the *Oracle WebLogic Server Administration Console Online Help*.

14.2.2 Enabling Global Trust

> **Caution:** Enabling Global Trust between WebLogic domains has the potential to open the servers up to man-in-the-middle attacks. Great care should be taken when enabling trust in a production environment. Oracle recommends having strong network security such as a dedicated communication channel or protection by a strong firewall.

WebLogic Server enables you to establish global trust between two or more domains. You do this by specifying the same Domain Credential for each of the domains. By default, the Domain Credential is randomly generated and therefore, no two domains will have the same Domain Credential. If you want two WebLogic domains to interoperate, you need to replace the generated credential with a credential you select, and set the same credential in each of the domains. For configuration information, see "Enable global trust between domains" in the *Oracle WebLogic Server Administration Console Online Help*.

If you enable global trust between two domains, the trust relationship is transitive and symmetric. In other words, if Domain A trusts Domain B and Domain B trusts Domain C, then Domain A will also trust Domain C and Domain B and Domain C will both trust Domain A. In most cases, the credential mapper approach, described in Section 14.2.1, "Enabling Cross Domain Security Between WebLogic Server Domains", is preferable to the global trust approach, because it is provides closer control over access.

Global trust between domains is established so that principals in a Subject from one WebLogic domain are accepted as principals in another domain. When this feature is enabled, identity is passed between WebLogic domains over an RMI connection without requiring authentication in the second domain (for example: log in to Domain 1 as Joe, make an RMI call to Domain 2 and Joe is still authenticated). WebLogic Server signs Principals with the Domain Credential as Principals are created. When a Subject is received from a remote source, its Principals are validated (the signature is recreated and if it matches, the remote domain has the same Domain Credential). If validation fails, an error is generated. If validation succeeds, the Principals are trusted as if they were created locally.

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

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

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

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

To avoid these issues, Oracle recommends that rather than enabling global trust between two domains, you should instead configure users with the `CrossDomainConnector` role and use the credential mapping approach described in Section 14.2.1, "Enabling Cross Domain Security Between WebLogic Server Domains".

14.3 Using Connection Filters

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

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

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

To configure a connection filter:

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

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

For more information:

- See "Configure connection filtering" in the *Oracle WebLogic Server Administration Console Online Help*.
- For information about connection filter rules and writing a custom connection filter, see "Using Network Connection Filters" and "Developing Custom Connection Filters" in *Developing Applications with the WebLogic Security Service*.
- You can also use the WebLogic Scripting Tool or Java Management Extensions (JMX) APIs to create a new security configuration.

14.4 Using the Java Authorization Contract for Containers

As of version 12.1.1, WebLogic Server supports the Java Authorization Contract for Containers (JACC) Standard, Version 1.4. JACC can replace the EJB and Servlet container deployment and authorization provided by WebLogic Server. When you configure a WebLogic domain to use JACC, EJB and servlet authorization decisions are made by the classes in the JACC framework. All other authorization decisions within WebLogic Server are still determined by the WebLogic Security Framework. For information about the WebLogic JACC provider, see "Using the Java Authorization Contract for Containers" in *Developing Applications with the WebLogic Security Service*.

You configure WebLogic Server to use JACC with a command line start option. For more information, see the description of the `-Djava.security.manager` option in the "weblogic.Server Command-Line Reference" in *Command Reference for Oracle WebLogic Server*.

Note that an Administration Server and all Managed Servers in a domain need to have the same JACC configuration. If you change the JACC setting on the Administration Server, you should shut down the Managed Server and reboot them with the same settings as the Administration Server to avoid creating a security vulnerability. Otherwise, it may appear that EJBs and servlets in your domain are protected by WebLogic Security Framework roles and policies, when in fact the Managed Servers are still operating under JACC.

14.5 Viewing MBean Attributes

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

To verify the setting of the **Anonymous Admin Lookup Enabled** option in the WebLogic Server Administration Console, select **Domain** > **Security** > **General**, or view the `SecurityConfigurationMBean.AnonymousAdminLookupEnabled` attribute.

14.6 How Passwords Are Protected in WebLogic Server

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

If the `SerializedSystemIni.dat` file is destroyed or corrupted, you must reconfigure the WebLogic 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.

14.7 Protecting User Accounts

WebLogic Server defines a set of configuration options to protect user accounts from intruders. In the default security configuration, these options are set for maximum protection. You can use the Administration Console to modify these options on the **Configuration** > **User Lockout** page.

As a system administrator, you have the option of turning off all the configuration options, increasing the number of login attempts before a user account is locked, increasing the time period in which invalid login attempts are made before locking the user account, and changing the amount of time a user account is locked. Remember that changing the configuration options lessens security and leaves user accounts vulnerable to security attacks. See "Set user lockout attributes" in the *Oracle WebLogic Server Administration Console Online Help*.

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

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

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

For information about unlocking a locked user account, see "Unlock user accounts" in the *Oracle WebLogic Server Administration Console Online Help*. Unlocking a locked user account can be done through either the WebLogic Administration Console or the `clearLockout` attribute on the `UserLockoutManagerRuntimeMBean`.

14.8 Configuring a Domain to Use JAAS Authorization

The security configuration in a WebLogic domain can be modified to use JAAS authorization, which interprets Subjects differently from the way in which the WebLogic Security Service does. For example, when a principal requests access to a resource that is protected by the Java policy provider in Oracle Platform Security Services (OPSS), the principal is compared to another principal that is built from a name contained in a policy store. (This comparison occurs when the `Principal.equals()` method is invoked.) If the appropriate attributes of the two principal objects match, access is granted.

Principal comparison is not used by the WebLogic Security Service to determine access decisions to protected resources. However, when principal comparison is performed in a default WebLogic domain, the comparison of principal names is case sensitive, and only the names of the principals are compared. To use JAAS authorization, the security configuration of a WebLogic domain can be modified to accommodate the following principal comparison behavior:

- The comparison of principal names is case insensitive
- The GUID and DN data in WebLogic principal objects are included in the comparison

To modify the security configuration of a WebLogic domain so that principal objects can be used with JAAS authorization, the following MBean attributes settings are available:

```
SecurityConfigurationMBean.PrincipalEqualsCaseInsensitive="true"
SecurityConfigurationMBean.PrincipalEqualsCompareDnAndGuid="true"
```

To set these attributes in the WebLogic Server Administration Console:

1. In the left pane of the Console, under **Domain Structure**, select the domain name.
2. Select **Configuration** > **Security** and click **Advanced**.
3. Select the check box next to each of the following entries:
   - **Principal Equals Case Insensitive**
   - **Principal Equals Compare DN and GUID**

> **Note:** If a domain is configured to use the GUID and DN data in principals, there may be an impact when interoperating with other WebLogic domains, particularly older domains, resulting from changes made to the way identity is passed.

For information about principal comparison in the Oracle Platform Security Service, see "Principal Name Comparison Logic" in *Application Security Guide*. For information about passing identity to a WebLogic domain, see *Developing Stand-alone Clients for Oracle WebLogic Server*.

15 Configuring JASPIC Security

This chapter describes how to configure the Java Authentication Service Provider Interface for Containers (JASPIC).

The Java Authentication Service Provider Interface for Containers (JASPIC) specification (http://www.jcp.org/en/jsr/detail?id=196) defines a service provider interface (SPI) by which authentication providers that implement message authentication mechanisms can be integrated in server Web application message processing containers or runtimes.

This section includes the following sections:

This section assumes that you are familiar with a basic overview of JASPIC, as described in *Understanding Security for Oracle WebLogic Server*.

15.1 JASPIC Mechanisms Override WebLogic Server Defaults

If you configure an Authentication Configuration Provider for a Web application, it is used instead of the WLS authentication mechanism for that Web Application. The JASPIC authentication provider assumes responsibility for authenticating the user credentials and returning a Subject.

You should therefore exercise care when you specify an Authentication Configuration Provider to make sure that it satisfies your security authentication needs.

15.2 Prerequisites for Configuring JASPIC

This section describes prerequisites for configuring JASPIC in your environment, including how to make your own or third party server authentication module (SAM) or Authentication Configuration Providers available to WebLogic Server.

The JASPIC programming model is described in the Java Authentication Service Provider Interface for Containers (JASPIC) specification (http://www.jcp.org/en/jsr/detail?id=196).

A sample SAM implementation is described in Adding Authentication Mechanisms to the GlassFish Servlet Container. Although written from the GlassFish Server perspective, the tips for writing a SAM, and the sample SAM itself, are instructive.

15.2.1 Server Authentication Module Must Be in Classpath

If you plan to configure a WebLogic Server Authentication Configuration Provider, you must add the jar for your SAM to the system classpath via the startup scripts or the command line used to start the WebLogic Server instance. If you do not do this, WebLogic Server is not able to find the appropriate classes.

15.2.2 Custom Authentication Configuration Providers Must Be in Classpath

If you plan to configure a custom Authentication Configuration Provider, you must add the jar for your custom Authentication Configuration Provider to the system classpath via the startup scripts or the command line used to start the WebLogic Server instance. If you do not do this, WebLogic Server is not able to find the appropriate classes.

15.3 Location of Configuration Data

You can use either the Administration Console or the WebLogic Scripting Tool (WLST) to configure JASPIC and the Authentication Configuration Providers.

After you configure JASPIC and the Authentication Configuration Providers, the domain-wide Authentication Configuration Provider configuration data is kept in the domain `config.xml` file in the `<jaspic>` element. For example:

```
<jaspic>
      <auth-config-provider xsi:type="wls-auth-config-providerType">
        <name>WLSAuthConfigProvider-0</name>
      </auth-config-provider>
    </jaspic>
```

When you configure an Authentication Configuration Provider for a deployed Web application, the Administration console (or WLST) updates the deployment plan (plan.xml) for the Web application with the application-specific Authentication Configuration Provider configuration. For example:

```
<variable>
      <name>JASPICProvider_AuthConfigProviderName_13210476440805</name>
      <value>WLSAuthConfigProvider-0</value>
</variable>
:
<variable-assignment>
   <name>JASPICProvider_AuthConfigProviderName_13210476440805</name>
   <xpath>/weblogic-web-app/jaspic-provider/auth-config-provider-name</xpath>
</variable-assignment>
```

15.4 Configuring JASPIC for a Domain

By default, JASPIC is enabled for a domain. This means that you can configure JASPIC properties for the domain, and JASPIC is available for any Web applications for which you have specified an Authentication Configuration Provider.

See "Configure Web applications for JASPIC" in *Oracle WebLogic Server Administration Console Online Help* for the specific steps to follow to configure JASPIC in the Administration Console.

If you disable JASPIC for a domain, JASPIC is then disabled for all Web applications in that domain, regardless of their configuration.

To configure JASPIC for a domain:

1. In the left pane, select the name of the domain for which you want to configure JASPIC.
2. Select Security > JASPIC > General.

   The JASPIC general page appears.
3. Ensure that the Enable JASPIC control is set for this domain.
4. Click Save.
5. Select Security > JASPIC > Authentication Configuration Providers.

   The JASPIC Authentication Configuration Providers page for the domain appears.
6. Click New.
7. From the drop-down list, select Create a New WLS Authentication Configuration Provider or Create a Custom WLS Authentication Configuration Provider.
8. On the Create a New WLS Authentication Configuration Provider page, set the desired values on the Name and Server Authentication Module (SAM) Class Name fields.

   You can accept the suggested name of WLSAuthConfigProvider-0, or use another name of your choice. The Server Authentication Module (SAM) Class Name identifies the Java class name of the SAM this Authentication Configuration Provider uses.
9. On the Create a New Custom Authentication Configuration Provider page, set the desired values on the Name and Class Name fields.

   You can accept the suggested name of CustomAuthConfigProvider-0, or use another name of your choice. The Class Name is dependent on the implementation of your custom Authentication Configuration Provider.
10. Enter the configuration properties for the Authentication Configuration Provider in the Configuration Properties text box.

    Each property must be on a separate line. For example: property1=value1.
11. Click Finish.
12. Restart WebLogic Server.

15.5 Displaying Authentication Configuration Providers

To display the Authentication Configuration Providers for a domain:

1. In the left pane, select the name of the domain for which you want to display the Authentication Configuration Providers.
2. Select Security > JASPIC > Authentication Configuration Providers.

   The JASPIC Authentication Configuration Providers page for the domain appears.

3. Select an existing Authentication Configuration Provider for which you want to display the configuration properties.

   The Settings page for this Authentication Configuration Provider appears.

4. Optionally, click the Notes page and enter any site-specific configuration information you want to capture.
5. If you made changes, click Save.
6. If you made changes, restart WebLogic Server.

15.6 Configuring JASPIC for a Web Application

You can specify which, if any, Authentication Configuration Provider is to apply to a specific Web application.

Before you can do this, you must first perform the following steps, as described in Section 15.4, "Configuring JASPIC for a Domain".

1. Enable JASPIC in the domain.
2. Configure a WebLogic Server Authentication Configuration Provider. Or,
3. Configure a Custom Authentication Configuration Provider.

To configure JASPIC properties for this Web application:

1. In the left pane of the Console, select Deployments.

   A table that lists the deployments currently installed on WebLogic Server appears in the right pane. The Type column specifies whether a deployment is an Enterprise application, a Web application, or an EJB module.

2. In the right pane, click the name of the Web application you want to configure.
3. Select Security > JASPIC to view and change the JASPIC properties.

   By default, JASPIC is disabled for Web applications. To enable JASPIC for this Web application, select one of the existing Authentication Configuration Providers from the drop-down list.

4. Click Save to save any changes.
5. Save the changes to the deployment plan, as prompted.
6. Redeploy the Web application.
7. Restart WebLogic Server.

15.7 Configuring JASPIC with WLST

This section describes how to use WebLogic Scripting Tool (WLST) to configure JASPIC. See *Understanding the WebLogic Scripting Tool* for information on using WLST.

This section requires you to configure the following MBeans via WLST:

- JASPICMBean
- CustomAuthConfigProviderMBean
- WLSAuthConfigProviderMBean

See *MBean Reference for Oracle WebLogic Server* for additional MBean information.

15.7.1 Creating a WLS Authentication Configuration Provider

Example 15–1 creates a WLS Authentication Configuration Provider, sets the class name of the SAM, and sets a configuration property.

After you run this example, restart WebLogic Server.

Example 15–1 Create a WLS Authentication Configuration Provider

```
connect('weblogic', 'password')
edit()
startEdit()
cd('SecurityConfiguration')
cd('mydomain')
jaspic = cmo.getJASPIC()
wacp = jaspic.createWLSAuthConfigProvider('wacp')
am = wacp.getAuthModule()
am.setClassName('com.my.auth.module.Classname')
props = Properties()
props.setProperty('property', 'value')
am.setProperties(props)
save()
activate()
```

15.7.2 Creating a Custom Authentication Configuration Provider

Example 15–2 creates a custom Authentication Configuration Provider, sets the class name of this Authentication Configuration Provider, and sets a configuration property.

After you run this example, restart WebLogic Server.

Example 15–2 Create a Custom Authentication Configuration Provider

```
connect('weblogic', 'password')
edit()
startEdit()
cd('SecurityConfiguration')
cd('mydomain')
jaspic = cmo.getJASPIC()
acp = jaspic.createCustomAuthConfigProvider('cacp')
acp.setClassName('com.my.acp.Classname')
props = Properties()
props.setProperty('property', 'value')
acp.setProperties(props)
save()
activate()
```

15.7.3 Listing All WLS and Custom Authentication Configuration Providers

Example 15–3 shows how to list all Authentication Configuration Providers for a domain.

Example 15–3 List All Authentication Configuration Providers

```
connect('weblogic', 'password')
edit()
startEdit()
cd('SecurityConfiguration')
cd('mydomain')
jaspic = cmo.getJASPIC()
jaspic.getAuthConfigProviders()
```

15.7.4 Enabling JASPIC for a Domain

Example 15–4 shows how to enable JASPIC for a domain.

After you run this example, restart WebLogic Server.

Example 15–4 Enable JASPIC for a Domain

```
connect('weblogic', 'password')
edit()
startEdit()
cd('SecurityConfiguration')
cd('mydomain')
jaspic = cmo.getJASPIC()
jaspic.setEnabled(false)
save()
activate()
```

15.7.5 Disabling JASPIC for a Domain

Example 15–5 shows how to disable JASPIC for a domain.

After you run this example, restart WebLogic Server.

Example 15–5 Disable JASPIC for a Domain

```
connect('weblogic', 'password')
edit()
startEdit()
cd('SecurityConfiguration')
cd('mydomain')
jaspic = cmo.getJASPIC()
jaspic.setEnabled(false)
save()
activate()
```

16 Using Compatibility Security

This chapter describes Compatibility security, which is the capability to run security configurations developed with 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.

This chapter includes the following sections:

> **Note:** Compatibility security is deprecated in this release of WebLogic Server and will not be supported in future major releases. Oracle strongly recommends upgrading your WebLogic Server deployment to the security features in this release of WebLogic Server. You should only use Compatibility security pending such an upgrade.

16.1 Running Compatibility Security: Main Steps

To set up Compatibility security:

1. Make a backup copy of your 6.x WebLogic domain (including your `config.xml` file) before using Compatibility security.
2. Add the following to the 6.x `config.xml` file if it does not exist, replacing the values with the actual names of your domain, security realm, and `FileRealm`:

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

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

To verify whether you are running Compatibility security correctly, open the new WebLogic Server Administration Console. If you are running Compatibility security, a Compatibility Security node is displayed on the left in the Domain Structure pane.

16.2 Limited Visibility of Compatibility Security MBeans

All Compatibility security MBeans are marked excluded and therefore have limited visibility in the WebLogic Scripting Tool. For example, the following command lists the attributes of the `DomainMBean`, excluding Compatibility security attributes such as `FileRealmMBean`:

```
java weblogic.WLST
connect()
ls()
```

However, if you address a Compatibility MBean directly, you can access it as in the following command:

```
java weblogic.WLST
connect()
cmo.getFileRealms()
```

16.3 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 used 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 "Define users" and "Define groups" topics of the Compatibility security section of the *Oracle WebLogic Server Administration Console Online Help*.
  - If you are using an alternate security realm (LDAP, Windows NT, RDBMS, or custom), you must use the administration tools provided by that realm to manage users and groups.

  For information about configuring a Realm Adapter Authentication provider, see Section 16.4, "Configuring a Realm Adapter Authentication Provider".

You can use implementations of the `weblogic.security.acl.CertAuthenticator` class in Compatibility security by configuring the Identity Assertion provider in the Realm Adapter Authentication provider. For more information, see Section 16.5, "Configuring the Identity Assertion Provider in the Realm Adapter Authentication Provider".

- Access Control Lists (ACLs) in the 6.x security realm are used to populate the Realm Adapter Authorization provider.
- The Realm Adapter Adjudication provider enables the use of both ACLs and security roles and security policies in Compatibility security. The Realm Adapter Adjudication provider can be used only with the Realm Adapter Authentication provider and the WebLogic Authorization provider. It resolves access decision conflicts between ACLs and new security policies set through the Administration Console. The Realm Adapter Adjudication provider permits access if the one authorization provider votes PERMIT and the other authorization provider votes DENY.
- The WebLogic Credential Mapping provider allows the use of credential maps in Compatibility security. For more information, see *Developing Resource Adapters for Oracle WebLogic Server*.
- You can add a Realm Adapter Auditing provider to access implementations of the `weblogic.security.audit.AuditProvider` class from the `CompatibilityRealm`. For more information, see "Configure a Realm Adapter Auditing Provider" in the *Oracle WebLogic Server Administration Console Online Help*.

16.4 Configuring a Realm Adapter Authentication Provider

When using Compatibility security, a Realm Adapter Authentication provider is by default configured for the `CompatibilityRealm`. For information about using the Realm Adapter Authentication provider in the `CompatibilityRealm`, see Section 16.3, "The Default Security Configuration in the CompatibilityRealm".

The Realm Adapter Authentication provider also allows use of implementations of the `weblogic.security.acl.CertAuthenticator` class with this release of WebLogic Server. The Realm Adapter Authentication provider includes an Identity Assertion provider that asserts identity based on X.509 tokens. For information about using a CertAuthenticator with WebLogic Server, Section 16.5, "Configuring the Identity Assertion Provider in the Realm Adapter Authentication Provider".

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

Note: The subjects produced by the Realm Adapter Authentication provider do not contain principals for the groups to which a user belongs. Use the `weblogic.security.SubjectUtils.isUserInGroup()` method to determine whether a user is in a group. When you use subjects produced by the Realm Adapter Authentication provider, you cannot iterate the complete set of groups to which a user belongs.

16.5 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 deprecated `weblogic.security.acl.CertAuthenticator` class. The identity assertion is performed on X.509 tokens. By default, the Identity Assertion provider is not enabled in the Realm Adapter Authentication provider.

For information about how to enable the Identity Assertion provider, see "Enable the Identity Assertion provider" in the *Oracle WebLogic Server Administration Console Online Help*.

16.6 Configuring a Realm Adapter Auditing Provider

The Realm Adapter Auditing provider allows you to use implementations of the `weblogic.security.audit.AuditProvider` interface when using Compatibility security. In order for the Realm Adapter Auditing provider to work properly, the implementation of the AuditProvider interface must have been defined. You can define the `AuditProvider` class using the Administration Console by selecting, in the **Audit Provider Class** field available from the **Domain: Compatibility Security** > **General** page.

For information, see "Configure a Realm Adapter Auditing provider" in the *Oracle WebLogic Server Administration Console Online Help*.

16.7 Protecting User Accounts in Compatibility Security

Password guessing is a common type of security attack. In this type of attack, a hacker attempts to log in to a computer using various combinations of usernames and passwords. WebLogic Server provides a set of lockout configuration options to protect user accounts from this kind of attack. By default, these options are set for maximum protection. As a system administrator, you have the option of turning off all the options, increasing the number of login attempts before a user account is locked, increasing the time period in which invalid login attempts are made before locking the user account, and changing the amount of time a user account is locked. Remember that changing the configuration options lessens security and leaves user accounts vulnerable to security attacks.

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

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

For information, see "Protect user accounts" and "Unlock user accounts" in the *Oracle WebLogic Server Administration Console Online Help*.

16.8 Accessing 6.x Security from Compatibility Security

Using Compatibility security assumes that 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 domain. WebLogic Server 6.x security management tasks such as configuring a security realm or defining ACLs should not be required and therefore those management tasks are not described in this section. However, if you corrupt an existing 6.x security realm and have no choice but to restore it, the following 6.x security management tasks are described in the Compatibility Security topic of the *Oracle WebLogic Server Administration Console Online Help*:

- "Configure LDAP V1 security realms"
- "Configure LDAP V2 security realms"
- "Configure RDBMS security realms"
- "Configure Windows NT security realms"
- "Configure wlauth for UNIX security realms"
- "Configure UNIX security realms"
- "Configure Custom security realms"
- "Configure Caching realms"
- "Configure the File realm"
- "Define ACLs"
- "Define groups"
- "Delete groups"
- "Define users"
- "Delete users"
- "Change user passwords"
- "Change the system password"
- "Disable the Guest user"

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

17

Security Configuration MBeans

This chapter describes the MBeans used in configuring the WebLogic Security Framework. Each MBean attribute is marked either dynamic, meaning that the attribute value can be changed without requiring a server restart, or non-dynamic, meaning that if you change the attribute value, you need to restart the server for the change to take effect. Note also that if an edit is made to a non-dynamic attribute, no edits to dynamic attributes will take effect until after restart. This is to assure that a batch of updates having a combination of dynamic and non-dynamic attribute edits will not be partially activated.

This chapter includes the following sections:

Any security MBeans not listed are completely non-dynamic (create or destroy MBean, change any attribute).

For more information about WebLogic Security MBeans, see:

- "Managing Security Realms with JMX" in *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*
- "Security MBeans" in the *MBean Reference for Oracle WebLogic Server*

17.1 SSLMBean

Creating or destroying this bean is dynamic.

Dynamic attributes:

Enabled, TwoWaySSLEnabled, ClientCertificateEnforced, ListenPort

Ciphersuites, ExportKeyLifespan, SSLRejectionLoggingEnabled, LoginTimeoutMillis

ServerCertificateChainFileName, ServerKeyFileName, ServerCertificateFileName, TrustedCAFileName

ServerPrivateKeyAlias, ServerPrivateKeyPassPhrase

IdentityAndTrustLocations

InboundCertificateValidation, OutboundCertificateValidation

All other attributes are non-dynamic.

17.2 ServerMBean

Creating or destroying this bean is dynamic.

Dynamic attributes:

KeyStores

CustomIdentityKeyStoreFileName, CustomIdentityKeyStoreType, CustomIdentityKeyStorePassPhrase

CustomTrustKeyStoreFileName, CustomTrustKeyStoreType, CustomTrustKeyStorePassPhrase

JavaStandardTrustKeyStorePassPhrase

All other attributes are non-dynamic.

17.3 EmbeddedLDAPMBean

Dynamic attributes:

Credential

All other attributes are non-dynamic

17.4 RDBMSSecurityStoreMBean

Creating or destroying this MBean is non-dynamic.

All attributes are non-dynamic.

17.5 SecurityMBean

Dynamic attributes:

ConnectionFilterRules

ConnectionLoggerEnabled

All other attributes are non-dynamic

17.6 SecurityConfigurationMBean

Dynamic attributes:

Credential

ConnectionFilterRules, ConnectionLoggerEnabled, CompatibilityConnectionFiltersEnabled

NodeManagerUsername, NodeManagerPassword

All other attributes are non-dynamic.

17.7 RealmMBean

Creating or destroying this MBean is non-dynamic.

Dynamic attributes:

DeployRoleIgnored, DeployPolicyIgnored, DeployCredentialMappingIgnored

FullyDelegateAuthorization

ValidateDDSecurityData, SecurityDDModel

CombinedRoleMappingEnabled

All other attributes are non-dynamic

17.8 WindowsNTAuthenticatorMBean

Creating or destroying this MBean is non-dynamic.

Dynamic attributes:

BadDomainControllerRetryInterval

MapUPNNames, LogonType

MapNTDomainName

All other attributes are non-dynamic.

17.9 CustomDBMSAuthenticatorMBean

Creating or destroying this MBean is non-dynamic. The ControlFlag and read-only provider attributes (such as ProviderClassName and Description) are non-dynamic. All other attributes are dynamic.

17.10 ReadonlySQLAuthenticatorMBean

Creating or destroying this MBean is non-dynamic.

The ControlFlag and read-only provider attributes (such as ProviderClassName and Description) are non-dynamic. All other attributes are dynamic.

17.11 SQLAuthenticatorMBean

Creating or destroying this MBean is non-dynamic.

The ControlFlag and read-only provider attributes (such as ProviderClassName and Description) are non-dynamic. All other attributes are dynamic.

17.12 DefaultAuditorMBean

Creating or destroying this MBean is non-dynamic.

Dynamic attributes:

Severity

All other attributes are non-dynamic

17.13 Compatibility Security MBeans

All MBeans used for Compatibility security are completely non-dynamic (create or destroy MBean, change any attribute). These MBeans include:

- RealmMBean
- FileRealmMBean
- BasicRealmMBean
- CachingRealmMBean
- PasswordPolicyMBean
- CustomRealmMBean
- LDAPRealmMBean
- NTRealmMBean
- RDBMSRealmMBean
- UnixRealmMBean

17.14 UserLockoutManagerMBean

This MBean is completely non-dynamic (create or destroy MBean, change any attribute).

17.15 Other Security Provider MBeans

All other security MBeans are completely non-dynamic (create or destroy MBean, change any attribute).