**Configuring SSL for Production Environments**

AquaLogic Enterprise Security uses an implementation of the Transport Layer Security (TLS) 1.0 specification, also referred to as SSL. The server (WebLogic Server or Tomcat) hosting ALES supports TLS on a dedicated listening port that defaults to 7010. To establish a secure connection, a client (Web browser or Java application) connects to the Administration Server by supplying the port and the secure address (HTTPS) in the connection URL, e.g., https://myserver:7010. The Administration Server returns a certificate to identify itself to the client.

When you install ALES, demonstration certificates are provided and configured automatically for working in a development environment. However, it is very important that these certificates not be used in a production environment.

Secure Sockets Layer (SSL) is described in the following sections.

* [SSL Basics on page 5-1](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html" \l "wp1074216)

* [Configuring SSL on page 5-4](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html" \l "wp1074264)

* [Keytool Utility on page 5-8](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html" \l "wp1074399)

**SSL Basics**

Basic information about SSL and ALES is contained in the following sections.

* [Private Keys, Digital Certificates, and Trusted Certificate Authorities on page 5-2](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html" \l "wp1074231)

* [One-Way SSL versus Two-Way SSL on page 5-2](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html" \l "wp1074237)

* [How the Administration Server Establishes Trust on page 5-4](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html" \l "wp1074258)

For more information about SSL and WebLogic Server, see [Configuring SSL](https://docs.oracle.com/cd/E13222_01/wls/docs92/secmanage/ssl.html) in *Securing WebLogic Server* in the WebLogic Server documentation set.

**Private Keys, Digital Certificates, and Trusted Certificate Authorities**

Private keys, digital certificates, and trusted certificate authorities establish and verify server identity. SSL uses public key encryption 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 vice versa. The private key is carefully protected so that only the owner can decrypt messages that were encrypted using the public key.

The public key is embedded within digital certificate along 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 is digitally signed with the digital certificate of the certificate authority. Well-known certificate authorities include Verisign and Entrust. The trusted CA certificate establishes trust for a certificate.

Web browsers, servers, and other SSL-enabled applications generally accept as genuine any digital certificate that is signed by a trusted certificate authority and is otherwise valid. For example, a digital certificate can be invalid because it has expired, or the digital certificate of the CA used to sign it expired, or because the host name in the digital certificate of the server does not match the URL specified by the client.

**One-Way SSL versus Two-Way SSL**

You can configure SSL to use either one-way or two-way authentication:

**One-way SSL**

To establish an SSL connection, the server must present a certificate to the client, but the client is not required to present a certificate to the server. To successfully negotiate an SSL connection, the client must authenticate the server, but the server accepts any client into the connection. One-way SSL is common on the Internet where customers want to create secure connections before sharing personal data. Often, clients use SSL to log on so that the server can authenticate them. By default, the Administration Server is configured for one-way SSL using demo certificates.

**Two-Way SSL (SSL with Client Authentication)**

To establish the SSL connection, the server must present a certificate to the client and the client must also present a certificate to the server. ALES can be configured to require clients to submit valid and trusted certificates before completing the SSL connection.

**Keystores**

A keystore is a mechanism designed to create and manage private key/digital certificate pairs and trusted CA certificates.

All private key entries in a keystore are accessed through unique aliases and a password that is specified when creating the private key in the keystore. The default alias for the ALES Administration Server certificates is ales-webserver. **Note:** Aliases are case-insensitive; the aliases Hugo and hugo would refer to the same keystore entry.

ALES explicitly trusts all certificate authorities whose entries are found in the keystore configured as trusted. Although ALES 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.

Upon installation, two keystores are used to establish trust between the Administration Server and clients:

* Webserver.jks— The keystore is located in the Administration Servers ssl directory. It contains:
  + a demonstration private key for the Administration Server.
  + the identity for the Administration Server in a public certificate that is signed by a trusted ALES Demo CA and bound to the server's hostname.
  + a public certificate for the ALES Demo CA itself.
* DemoTrust.jks— This keystore is located in the ssl directory of the SSM or SCM instance. It used by enrollment clients when connecting from an SSM or SCM instance. It contains the public certificate of the same trusted ALES Demo certificate authority that is in webserver.jks. This keystore is used when running enroll.bat/sh (for SSM) or enrolltool.bat/sh (for SCM) with the demo argument. When using the secure argument, the SSM enroller uses $JAVA\_HOME/lib/security/cacerts, while the SCM uses its own trust.jks keystore.

For descriptions of common keytool commands, see [Keytool Utility on page 5-8](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html#wp1074399).

**How the Administration Server Establishes Trust**

The client types connecting to the Administration Server are: (1) Internet Explorer browsers accessing the administration console, and (2) SSM enrollment clients. The method used to establish trust depends on the client type.

* Internet Explorer browsers. Browser clients will not have the ALES demo certificate or demo CA certificate in the trusted store, so a security alert window will display when accessing the administration console. The user can use the window to trust the Administration Server's demo certificate. **Note:** The alert window does not display when the Administration Server is configured to use a valid signed certificate.
* SSM and SCM enrollment clients. An enrollment client uses its DemoTrust.jks keystore to establish trust. When the client tries to enroll, the Administration Server presents its public certificate for verification to the enrollment client. The client will trust the certificate, because the DemoTrust.jks keystore that it is using in demo mode has the same ALES Demo CA certificate.

The important thing to remember when updating certificates is that the server and client both trust a common CA.

**Configuring SSL**

To configure SSL for a production environment you must create a keystore to replace Webserver.jks and configure the Administration Server to use it. Then you may configure ALES to use one-way or two-way SSL.

* [Create a Keystore and Load Signed Certificates on page 5-4](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html" \l "wp1074283)

* [Configuring One-Way SSL on page 5-5](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html" \l "wp1074301)

* [Configuring Two-Way SSL on page 5-7](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html" \l "wp1074327)

Procedures described in this section make use of Sun's keytool utility. For information about this tool, see [Keytool Utility](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html#wp1074399).

**Create a Keystore and Load Signed Certificates**

1. Create the keystore and private key as follows:
   1. Create a secureWebserver.jks keystore and generate the private key using keytool utility as follows:

keytool -genkey -alias ales-webserver -keyalg RSA -keystore secureWebserver.jks

* 1. When prompted, enter the keystore password and general information about the certificate, (company, contact name, etc.). This information is displayed to users who attempting to access a secure page in the application.
  2. When prompted for the key password, enter the same password used for the keystore itself. This can be accomplished by pressing ENTER.

1. Create a Certificate Signing Request (CSR) as follows:
   1. Create certreq.csr by entering:

keytool -certreq -alias ales-webserver -keyalg RSA -file certreq.csr -keystore secureWebserver.jks

* 1. Submit certreq.csr to the Certificate Authority.

1. Import the certificate into the keystore as follows:
   1. Download a Chain Certificate from the Certificate Authority. Then import it into the keystore using the following command:

keytool -import -alias cacerts -keystore secureWebserver.jks -trustcacerts -file <filename\_of\_the\_chain\_certificate>

* 1. Import the new certificate using the following command.

keytool -import -alias ales-webserver -keystore secureWebserver.jks -trustcacerts -file <your\_certificate\_filename>

* 1. **For SCM only**, add the CA's certificate chain to trust.jks in the ssl directory.

keytool -import -alias cacerts -keystore trust.jks -trustcacerts -file <filename\_of\_the\_chain\_certificate>

**Configuring One-Way SSL**

The procedure for configuring the new keystore (secureWebserver.jks) for production use on depends on the type of server hosting ALES. This section provides instructions for WebLogic Server and Tomcat.

**Configure One-Way SSL on WebLogic Server**

Perform the following steps to use the secure the keystore when using WebLogic Server.

|  |  |
| --- | --- |
| **Note:** | For information about using the WebLogic Server Administration Console to configure SSL, see [Set up SSL](https://docs.oracle.com/cd/E13222_01/wls/docs92/ConsoleHelp/taskhelp/security/SetUpSSL.html) in the WLS Administration Console online help. |

1. Copy secureWebserver.jks to the ssl directory where the Administration Server is installed (the default is BEA\_HOME/ales25-admin/ssl).
2. Modify the server's configuration file (BEA\_HOME/asiDomain/config.xml) as follows.
   1. Replace every occurrence of webserver.jks with secureWebserver.jks.
   2. Change the ServerPrivateKeyAlias attribute to match the alias that is assigned to the certificate in the secureWebserver.jks keystore. In the example above it was ales-webserver.
   3. Change the ServerPrivateKeyPassPhrase attribute to match the password for the secureWebserver.jks keystore.
3. Restart the Administration Server.

After performing these steps, running enroll.bat/sh (for a SSM) or enrolltool.bat/sh (for a SCM) will pass in secure instead of demo as an argument.

**Configure One-Way SSL on Tomcat**

Perform the following steps to use the secure keystore when using WebLogic Server.

|  |  |
| --- | --- |
| **Note:** | For more information about SSL under Apache Tomcat, see [http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html](http://tomcat.apache.org/tomcat-5.0-doc/ssl-howto.html) . |

1. Copy secureWebserver.jks to the ssl directory where the Administration Server is installed (the default is BEA\_HOME/ales25-admin/ssl).
2. Modify the server's configuration file (TOMCAT\_HOME/config/server.xml) as follows.
   1. Replace every occurrence of webserver.jks with secureWebserver.jks.
   2. Add keystorePass=<your\_password> next to the keystoreFile attribute.
3. Restart the Administration Server.

After performing these steps, running enroll.bat/sh (for a SSM) or enrolltool.bat/sh (for a SCM) will pass in secure instead of demo as an argument.

**Configuring Two-Way SSL**

The procedure for configuring the new keystore (secureWebserver.jks) for two-way SSL (SSL with client authentication) depends on the type of server hosting ALES. This section provides instructions for WebLogic Server and Tomcat.

**Configure Two-Way SSL on WebLogic Server**

To configure the Administration Server for two-way SSL on WebLogic server:

1. Configure one-way SSL as described in [Configuring One-Way SSL on page 5-5](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html#wp1074301).
2. Log in to the WebLogic Server Administration Console.
3. Expand the Servers node and select name adminserver.
4. Select the Configuration-->Keystores and SSL tab.
5. Click the Show link under Advanced Options.
6. In the Server attributes section of the window, set the Two-Way Client Cert Behavior attribute. The available options are shown in [Table 5-1](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html#wp1074349).

|  |  |
| --- | --- |
| Table 5-1 **Two Way SSL Cert Behavior Options** | |
| **Option** | **Description** |
| Client Certs Not Requested | The default (meaning one-way SSL). |
| Client Certs Requested But Not Enforced | Requires a client to present a certificate. If a certificate is not presented, the SSL connection continues. |
| Client Certs Requested And Enforced | Requires a client to present a certificate. If a certificate is not presented or if the certificate is not trusted, the SSL connection is terminated. |

1. Click Apply.

After performing these steps, running enroll.bat/sh (for a SSM) or enrolltool.bat/sh (for a SCM) will pass in secure instead of demo as an argument.

**Configure Two-Way SSL on Apache Tomcat**

To configure the Administration Server for two-way SSL on WebLogic server:

1. Configure one-way SSL as described in [Configuring One-Way SSL on page 5-5](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html#wp1074301).
2. Open TOMCAT\_HOME/config/server.xml in a text editor and set the value of clientAuth as follows.

|  |  |
| --- | --- |
| Value | Description |
| false | Tomcat will NOT require all SSL clients to present a client Certificate in order to use this socket. (1-way SSL) |
| want | Tomcat will request a client Certificate, but not fail if one isn't presented. (Optional 2-way SSL) |
| true | Tomcat will require all SSL clients to present a client Certificate in order to use this socket. (Mandatory 2-way SSL) |

After performing these steps, running enroll.bat/sh (for a SSM) or enrolltool.bat/sh (for a SCM) will pass in secure instead of demo as an argument.

**Keytool Utility**

Sun Microsystem's keytool utility is included in JDK installations. For complete information about this tool, consult the Sun Microsystems web site. See also detailed command usage options for Windows ( <http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html>) and Solaris/Linux ( <http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/keytool.html>).

When using the keytool utility, observe the following:

* + The keytool utility does not allow you to import existing private keys into the keystore.
  + When using the keytool utility, the default key pair generation algorithm is DSA. Specify another key pair generation such as RSA algorithm when using ALES.
  + ALES currently operates only on JKS keystores. The JKS format is Java's standard keystore format and is the format created by the keytool command-line utility.

[Table 5-2](https://docs.oracle.com/cd/E13169_01/ales/docs25/admindeployguide/ssl.html" \l "wp1074414) shows the keytool commands to use when creating and using JKS keystores.

|  |  |
| --- | --- |
| Table 5-2 Common Keytool Commands | |
| **Command** | **Description** |
| keytool -genkey -alias aliasforprivatekey -keystore keystorename -storepass keystorepassword | Generates a new private key entry and self-signed digital certificate in a keystore. If the keystore does not exist, it is created. |
| keytool -import -alias aliasforprivatekey -file certificatefilename.pem -keypass privatekeypassword -keystore keystorename -storepass keystorepassword | Updates the self-signed digital certificate with one signed by a trusted CA. |
| keytool -import -alias aliasfortrustedca -trustcacerts -file trustedcafilename.pem -keystore keystorename -storepass keystorepassword | Loads a trusted X.509 CA certificate or PKCS#7 certificate chain into a keystore. If the keystore does not exist, it is created. |
| keytool -certreq -alias aliasforprivatekey -sigalg RSA -file certreq\_file -keypass privatekeypassword -storetype keystoretype -keystore keystorename -storepass keystorepassword | Generates a CSR (using the PKCS#10 format) to be sent to a trusted CA. The trusted CA authenticates the certificate requestor and returns a digital certificate, which replaces the existing self-signed digital certificate in the keystore. |
| keytool -list -keystore keystorename | Displays what is in the keystore. |
| keytool -delete -alias aliasforprivatekey -keystore keystorename -storepass keystorepassword -alias privatekeyalias | Delete a private key/digital certifcate pair for the specified alias from the keystore. |
| keytool -help | Provides online help for keytool. |