



CHAPTER 17

Enabling Resource Manager to Use Secure Sockets Layer Connections on a JBoss Application Server

Cisco Unified Videoconferencing Manager uses the JBoss application server platform. The JBoss application server installs with Cisco Unified Videoconferencing Manager automatically.

- [Introduction, page 17-1](#)
- [Prerequisites, page 17-2](#)
- [Using Keytool to Generate a Certificate, page 17-2](#)
- [Configuring JBoss to use SSL, page 17-5](#)
- [Accessing Resource Manager Using HTTPS, page 17-6](#)

Introduction

Secure Sockets Layer (SSL) connections rely on the existence of digital certificates. A digital certificate reveals information about its owner, including the owner's identity.

During the initialization of an SSL connection, the server must present its certificate to the client for the client to determine the server identity. The client can also present the server with its own certificate for the server to determine the client identity. SSL is therefore, a means of propagating identity between components.

A client can trust the contents of a certificate if that certificate is digitally signed by a trusted third party. A Certificate Authority (CA) acts as a trusted third party and signs certificates on the basis of its knowledge of the certificate requestor.

There are two options for creating a new certificate.

- Request that a CA generates the certificate on your behalf.

The CA creates a new certificate, digitally signs it, and delivers it to the requester. Popular web browsers are preconfigured to trust certificates that are signed by certain CAs. No further client configuration is necessary for a client to connect to the server through an SSL connection. Therefore, CA signed certificates are useful where configuration for each and every client that accesses the server is impractical.

- Generate a self-signed certificate.

This option is quicker and requires fewer details to create the certificate, but the certificate is not signed by a CA. Any client that connects to this server over an SSL connection needs configuration to trust the signer of this certificate. Therefore, self-signed certificates are only useful when you can configure each of the clients to trust the certificate. It is possible in some cases to present a self-signed certificate to an untrusting client. In some web browsers, when the certificate is received and does not match any of those listed in the client trust file, a prompt appears asking if the certificate should be trusted for the connection and added to the trust file.

Prerequisites

To use SSL with JBoss, the following conditions must be met:

- You have a certificate.
- You configure JBoss to use this certificate.
- You store the certificate in a JKS keystore.

Using Keytool to Generate a Certificate

Keytool is the command line Java utility. This section describes how to use keytool to create a private and public self-signed certificate key pair.

Procedure

-
- Step 1** Open a DOS window and set the path to point to the JDK or JRE *bin* directory. For example
- ```
D:\>set path= D:\jdk1.5.0\bin
```
- Step 2** Create a self-signed certificate key pair. For example
- ```
D:\>keytool -genkey -keyalg RSA
-dname "cn=scheduler,ou=users,ou=yourcountry,
DC=yourcompany,DC=com"
-alias scheduler -keypass yourcompany -keystore
scheduler.keystore
-storepass yourcompany
```
- Step 3** Specify RSA as the private key to ensure that the MD5 with RSA signature algorithm is used. Not all web browsers support the DSA cryptograph algorithm, which is the default when RSA is not specified.
- Step 4** Set a password of at least six characters to protect the private key.
- Step 5** Specify the keystore file and keystore password (the option is storepass).



Note Type each string on a single line.

If you do not wish to send a certificate signing request, skip to [“Configuring JBoss to use SSL” section on page 17-5](#).

- Step 6** Generate the certificate signing request. For example


```

gQB0mSnorCvcm8tluJ2S3hGzbDIRfxli61iLPX7x3aZhQbMjteK8Sm+w4xo2G0yG
n3Sw0cSrf+4YH0ioXgtAIyTDE6aXE+vRINH0TEpq5xwRLY/7WH+EyYwaMs7EhRqC
gzlFicEqfBea/he5maogsW0OVptY5IH6erPOHawfszkUITCCApkwggICoAMCAQIC
AQAwDQYJKoZIhvcNAQEEBQAwwYcxCzAJBgNVBAYTAlpBMSIwIAYDVQQQIEExIGT1Ilg
VEVTVEI0RyBQVVJQT1NFUyBPTkxZMR0wGwYDVQQKEXRUAzGF3dGUgQ2VydGlmaWNh
dGlvbjEXMBUGA1UECXMOMOVEVTVCBURVNUIFRFU1QxHDAaBgNVBAMTE1RoYXd0ZSBU
ZXN0IENBIFJvb3QwHhcNOTYwODAxMDAwMDAwWhcNMjAxMjMxMjE1OTU5WjCBhzEL
MAkGA1UEBhMCWkExIjAgBgNVBAgTGUZPUiBURVNUSU5HIFBVU1BPU0VTIE9OTFkx
HTAbBgNVBAoTFFRoYXd0ZSBBDZXJ0aWZpY2F0aW9uMRcwFQYDVQQLEw5URVNUIFRF
U1QgVEVTVDEcMBoGA1UEAxMTVGhhd3RIIFRlc3QgQ0EgUm9vdDCBnzANBgkqhkiG
9w0BAQEFAAOBjQAwgYkCgYEAAtX2Qb46zrH8M6Gb60pRB/NUxYaET3mwWYS2QwxNf
ZmLifqLoG/OhF4nmePO3UsVyyq7gRST0mLbR6Aop+apHOZAUsv7WeJJ18kRQTwp
jjRCgj33bvQ9OsuM15oxwKX4BiX6QP5EvkG24opTc+6thywKyt7ppO+MxyqlMk4Z
5I8CAwEAAAMTMBEwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQQFAAOBgQCC
jOPsGQP07yyLZXS/4Rn+3sR/P8sLzLqYPsgFTOIfln/nfKYYAYu3KFce9f5QgTpR
ZseujcipWSuZjm4L3dnZZ7AD/p50rcoVRxSKSst57AzuCaqyIBIQRHaYpEqW7XaN
fi1Lm1es9ZEO9Qq0ULW4v1rXasUBBLYGQiTtpT6CFjEA
-----END PKCS #7 SIGNED DATA-----

```

Step 9 Import the CA trusted root certificate into the keystore. For example

```

D:\>keytool -import -alias "Provider Test CA Root" -file "Provider Test Root.cer"
-keystore sceduler.keystore -storepass yourcompany

```

where

- Provider Test CA Root is the directory containing the test CA root binary and text files.
- Provider Test Root.cer is the test CA root binary file.

When the command is successfully executed, the following output displays:

```

Owner: CN=Provider Test CA Root, OU=TEST TEST TEST, O=Provider Certification, ST=FOR
TESTING PURPOSES ONLY, C=ZA

```

```

Issuer: CN=Provider Test CA Root, OU=TEST TEST TEST, O=Provider Certification, ST=FOR
TESTING PURPOSES ONLY, C=ZA

```

```

Serial number: 0

```

```

Valid from: Thu Aug 01 08:00:00 CST 1996 until: Fri Jan 01 05:59:59 CST 2021

```

```

Certificate fingerprints:

```

```

MD5: 5E:E0:0E:1D:17:B7:CA:A5:7D:36:D6:02:DF:4D:26:A4

```

```

SHA1: 39:C6:9D:27:AF:DC:EB:47:D6:33:36:6A:B2:05:F1:47:A9:B4

```

```

Trust this certificate? [no]: y

```

```

Certificate was added to keystore

```

Step 10 Import the certificate responses from the CA into the keystore file using the same alias name that was first given to the self-signed certificates.

In this example, the alias name is scheduler. Using an alternative alias name generates a new signed certificate and not a personal certificate chain.

```
D:\>keytool -import -trustcacerts -alias scheduler -file scheduler.cer
        -keystore scheduler.keystore -storepass yourcompany
```

When the command is successfully executed, the following output displays:

```
Certificate reply was installed in keystore
```

```
We have now created a keystore file that stores a valid certificate for use.
```

Configuring JBoss to use SSL

This section describes how to configure the JBoss application server for use with SSL.

Procedure

-
- Step 1** Copy the scheduler.keystore file to
<Resource Manager installation directory>\jboss\server\default\conf
 - Step 2** Open the server.xml file located in jboss\server\default\deploy\jbossweb-tomcat50.sar
 - Step 3** Locate the section beginning with the line
<!-- SSL/TLS Connector configuration using the admin devl guide keystore
 - Step 4** Remove the comment indicators and make the following changes:
 - a. Uncomment out the SSL/TLS connector.
 - b. Change the keystore file from **chap8.keystore** to **scheduler.keystore**.
 - c. Change the keystorePass from **rmi+ssi** to **yourcompany**.
 - d. We recommend that you change the port from 8443 to 443 so that the user does not need to type the port when accessing Resource Manager. Like port 80, port 443 is a known HTTPS port.

The amended text appears as follows:

```
<!-- A HTTP/1.1 Connector on port 8080 or 80 -->
<Connector port="8080" address="{jboss.bind.address}"
  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" redirectPort="443" acceptCount="100"
  connectionTimeout="20000" disableUploadTimeout="true"/>

<!-- A AJP 1.3 Connector on port 8009 -->
<Connector port="8009" address="{jboss.bind.address}"
  enableLookups="false" redirectPort="443" debug="0"
  protocol="AJP/1.3"/>

<!-- SSL/TLS Connector configuration using the admin devl guide keystore -->
<Connector port="443" address="{jboss.bind.address}"
  maxThreads="100" minSpareThreads="5" maxSpareThreads="15"
  scheme="https" secure="true" clientAuth="false"
  keystoreFile="{jboss.server.home.dir}/conf/
  scheduler.keystore"
  keystorePass="yourcompany" sslProtocol = "TLS" />
<!-- -->
```

Step 5 Restart JBoss.

Accessing Resource Manager Using HTTPS

This section describes how to access Resource Manager via an HTTPS connection.

Procedure

- Step 1** Type a URL of the format `https://localhost`, or `https://localhost:8443` (if port 8443 is used instead of 443). If the certificate in use is a test root certificate or a self-signed certificate that is not trusted by Internet Explorer, a security alert appears.
- Step 2** Click **Yes** to access Resource Manager.
- Step 3** To avoid this message in future logins, click **View Certificate**:
- Step 4** Click **Install Certificate**.
- Step 5** After the certificate is installed, the user will not see the security alert on subsequent logins.
-