# SSL or TLS Configuration for Tomcat
## Oracle FLEXCUBE Universal Banking
### Release 14.3.0.0.0
### [May] [2019]

**ORACLE**
**FINANCIAL SERVICES**

# Table of Contents

**ORACLE**

# 1.   SSL or TLS Configuration

## 1.1   Introduction

This document explains the steps to configure the SSL/TSL for Tomcat 8.0.x.

## 1.2   Reference Sites

http://tomcat.apache.org/tomcat-8.0-doc/ssl-howto.html

## 1.3   Prerequisites

### 1.3.1   SSL and Tomcat

The description below uses the variable name $CATALINA_BASE to refer the base directory against which most relative paths are resolved.

Configuring Tomcat to take advantage of secure sockets is necessary only when running it as a stand-alone web server. When running Tomcat primarily as a Servlet/JSP container behind another web server such as Apache or Microsoft IIS, it is necessary to configure the primary web server to handle the SSL connections from users.

See http://tomcat.apache.org/tomcat-8.0-doc/ssl-howto.html for more information.

### 1.3.2   Configuration

#### 1.3.2.1 Prepare the Certificate Keystore

Tomcat currently operates only on JKS, PKCS11 or PKCS12 format keystores. The JKS format is Java's standard "Java KeyStore" format, and is the format created by the keytool command-line utility.

To create a new keystore from scratch, containing a single self-signed Certificate, execute the following from a terminal command line:

Windows: %JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA

Unix: $JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA

To create a new keystore from scratch, containing a single self-signed Certificate, execute the command specified below, from a terminal command line.

Windows: %JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA -keystore \path\to\my\keystore

Unix: $JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA -keystore /path/to/my/keystore

ORACLE

The location of the keystore to be created is specified via the -keystore parameter, followed by the complete pathname to your keystore file. You will also need to reflect this new location in the server.xml configuration file, as described later.

After executing this command, you will first be prompted for the keystore password. The default password used by Tomcat is "changeit" (all lower case), although you can specify a custom password if you like. You will also need to specify the custom password in the server.xml configuration file, as described later.

Next, you will be prompted for general information about this Certificate, such as company, contact name, and so on. This information will be displayed to users who attempt to access a secure page in your application, so make sure that the information provided here matches what they will expect.

Finally, you will be prompted for the key password, which is the password specifically for this Certificate (as opposed to any other Certificates stored in the same keystore file). You MUST use the same password here as was used for the keystore password itself. (Currently, the keytool prompt will tell you that pressing the ENTER key does this for you automatically.)

If everything was successful, you now have a keystore file with a Certificate that can be used by your server.

🛑 Your private key password and keystore password should be the same. If they differ, you will get an error along the lines of java.io.IOException: Cannot recover key, as documented in Bugzilla issue 38217, which contains further references for this issue.

### 1.3.3  Import Host Cert into Branch Keystore

1. Export main serevr cert and save it as a .cer file

keytool -export -keystore FCUBSMain.jks -alias maincert -file FCUBSMain.cer

2. Import the .cer file into branch keystore

keytool -import -alias maincert -file FCUBSMain.cer –keystore FCUBSBranch.jks

3. Import the .cer file into JAVA truststore.

4. To find the trust store for tomcat,check the **JAVA_HOME** path

5. Go to **JAVA_HOME/ jre/lib/security** look for cacerts, which is the trust store for tomcat.

**6.** keytool -import -alias maincert -file FCUBSMain.cer –keystore JAVA_HOME/ jre/lib/securitycacerts

🛑 maincert is the alias name present in FCUBSMain.jks while creating the keystore.

### 1.3.4  Connector Configuration

You have to configure a single connector for FLEXCUBE UBS:

A connector related to SSL/TLS communication between Browser and Branch which uses one-way Authentication.

**ORACLE**

### 1.3.4.1 Connector Configuration (Related to Browser and Branch Communication)

The final step is to configure your secure socket in the $CATALINA_BASE/conf/server.xml file, where $CATALINA_BASE represents the base directory for the Tomcat 6 instance.

Use either of the following Connector definitions to configure the SSL/TSL communication port which will be use for one way authentication (which will be use to serve the browser requests). The first definition utilizes the blocking connector, whereas the second uses the non-blocking connector to service requests.

```
<!-- Define a blocking Java SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector protocol="org.apache.coyote.http11.Http11Protocol"
    port="8443" minSpareThreads="5" maxSpareThreads="75"
    enableLookups="true" disableUploadTimeout="true"
    acceptCount="100" maxThreads="200"
    scheme="https" secure="true" SSLEnabled="true"
    keystoreFile="${user.home}/.keystore" keystorePass="changeit"
    clientAuth="false" sslProtocol="TLS"
    sslEnabledProtocols="TLSv1,TLSv1.1,TLSv1.2,SSLv2Hello"/>
```

```
<!-- Define a non-blocking Java SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector protocol="org.apache.coyote.http11.Http11NioProtocol"
    port="8443" minSpareThreads="5" maxSpareThreads="75"
    enableLookups="true" disableUploadTimeout="true"
    acceptCount="100" maxThreads="200"
    scheme="https" secure="true" SSLEnabled="true"
    keystoreFile="${user.home}/.keystore" keystorePass="changeit"
    clientAuth="false" sslProtocol="TLS"
    sslEnabledProtocols="TLSv1,TLSv1.1,TLSv1.2,SSLv2Hello"/>
```

**<Provide instructions on how to populate the keystoreFile and keystorePass>**

🛑 Note the following:

- If the de-centralized application WAR file was built with the SSL enabled, Tomcat will attempt to automatically redirect users accessing the application on the HTTP port, to the HTTPS port. If you change the port number in the SSL connector configuration, you should also change the value specified for the redirectPort attribute on the default non-SSL connector. This allows Tomcat to automatically redirect users who attempt to access the FLEXCUBE UBS de-centralized application on a non-SSL port to the SSL port.

**ORACLE**

- clientAuth must be false in case of one way authentication (for the port related connector to which Browser will connect).

# 1.4   <u>Two-Way SSL Connection</u>

A two-way SSL is used when the server needs to authenticate the client. In a two-way SSL connection, the client verifies the identity of the server and then passes its identity certificate to the server. The server then validates the identity certificate of the client before completing the SSL handshake.

This section discusses the two-way SSL connection between Oracle Weblogic server (Host) and Tomcat server (Branch) in which the branch acts as the server and host as a client (i.e. host and browser sends HTTPS request to the branch).

## 1.4.1  <u>Preparing Certificate</u>

In order to establish a two-way SSL connection, you need to have two certificates, one for the server and the other for client.

The steps mentioned in this section are for creating self signed certificates.

*For more information on the key tool commands and for creating trusted certificates issued by a CA, refer to the chapter 'Obtaining the Identity Store' in the installation manual 'SSL Configuration on Weblogic'.*

### 1.4.1.1 <u>Preparing Server Certificate</u>

Follow the steps given below:

1. Create java keystore certificate. This can be used as keystore in server.

   keytool -genkey -v -alias server -keyalg RSA -validity 3650 -keystore D:\keystore\ssl\server.jks -dname "CN=Branch, OU=tomcat, O=Oracle, L=EN, ST=KA, C=IN" -storepass changeit -keypass changeit

2. Export as .cer file so that it can be imported to other certificates.

   keytool -export -alias server -keystore D:\keystore\ssl\server.jks -storepass changeit -rfc -file D:\keystore\ssl\server.cer

### 1.4.1.2 <u>Preparing Client Certificate</u>

Follow the steps given below:

1. Create PKCS12 certificate. This can be used in client browser.

   keytool -genkey -v -alias client -keyalg RSA -storetype PKCS12 -keystore D:\keystore\ssl\client.p12 -dname "CN=Client, OU=Developer, O=Oracle, L=EN, ST=KA, C=IN" -storepass changeit -keypass changeit

2. Export as .cer file so that it can be imported to other certificates.

   keytool -export -alias client -keystore D:\keystore\ssl\client.p12 -storetype PKCS12 -storepass changeit -rfc -file D:\keystore\ssl\client.cer

3. Import .cer to java client java keystore file. This can be used in Oracle Weblogic server and browser.

ORACLE

```
keytool -importkeystore -srckeystore D:\keystore\ssl\client.p12 -srcstoretype PKCS12 -srcalias client -
destkeystore D:\keystore\ssl\client.jks -deststoretype jks -deststorepass changeit -destalias client
```

## 1.4.2 Configuring Connector

For Oracle FLEXCUBE Universal Banking Solutions, you need to configure a single connector. This connector is related to SSL/TLS communication between host or browser and the branch which uses two-way authentication.

## 1.4.3 Configuring Connector related to Host and Branch Communication

### 1.4.3.1 Branch or Tomcat Configuration

You need to configure your secure socket in the '$CATALINA_BASE/conf/server.xml' file, where '$CATALINA_BASE' represents the base directory for the Tomcat 6 instance.

Use either of the following connector definitions to configure the SSL/TSL communication port which will be used for two-way authentication.

- This definition utilizes the blocking connector to service requests.

```
<!-- Define a blocking Java SSL Coyote HTTP/1.1 Connector on port 8443 -->

<Connector protocol="org.apache.coyote.http11.Http11Protocol"

    port="8443" minSpareThreads="5" maxSpareThreads="75"

    enableLookups="true" disableUploadTimeout="true"

    acceptCount="100" maxThreads="200"

    scheme="https" secure="true" SSLEnabled="true"

    keystoreFile="D:\keystore\ssl\server.jks" keystorePass="changeit"

    truststoreFile="D:\software\JDK\jdk1.6.0_18\jre\lib\security\cacerts" truststorePass="changeit"

    clientAuth="false" sslProtocol="TLS"

    sslEnabledProtocols="TLSv1,TLSv1.1,TLSv1.2,SSLv2Hello"/>
```

- This definition uses the non-blocking connector to service requests.

```
<!-- Define a non-blocking Java SSL Coyote HTTP/1.1 Connector on port 8443 -->

<Connector protocol="org.apache.coyote.http11.Http11NioProtocol "

    port="8443" minSpareThreads="5" maxSpareThreads="75"

    enableLookups="true" disableUploadTimeout="true"

    acceptCount="100" maxThreads="200"
```

ORACLE

```
scheme="https" secure="true" SSLEnabled="true"

keystoreFile="D:\keystore\ssl\server.jks" keystorePass="changeit"

truststoreFile="D:\software\JDK\jdk1.6.0_18\jre\lib\security\cacerts" truststorePass="changeit"

clientAuth="false" sslProtocol="TLS"

sslEnabledProtocols="TLSv1,TLSv1.1,TLSv1.2,SSLv2Hello"/>
```

🛑 Note the following:

- If the de-centralized application WAR file was built with SSL enabled, Tomcat attempts to redirect the users accessing the application on the HTTP port automatically to the HTTPS port. If you change the port number in the SSL connector configuration, you should also change the value specified for the attribute 'redirectPort' on the default non-SSL connector. This allows Tomcat to automatically redirect the users who attempt to access the FLEXCUBE UBS decentralized application on a non-SSL port to the SSL port.

- In case of one way authentication 'clientAuth' must be true for the port related connector to which the Browser or Host will connect.

- Trust store should import the client certificate as mentioned below.

- keytool -import -trustcacerts -v -file D:\keystore\ssl\client.cer -alias client -keystore D:\software\JDK\jdk1.6.0_18\jre\lib\security\cacerts -storepass changeit

### 1.4.3.2 Host or Oracle Weblogic Configuration

As part of this configuration, you need to configure the identity and trust stores and set the SSL attributes for private key alias and password.

**Configuring the Identity and Trust Stores**

Follow the steps given below:

1. Login to Oracle WebLogic Server Admin Console.

2. Go to 'Change Center' of the Administration Console and click 'Lock & Edit'.

3. Expand the 'Servers' node.

4. Select the name of the server for which you want to configure the keystores. For example, 'exampleserver'.

5. Under 'Configuration', select the 'Keystores' tab.

6. In the 'Keystores' field, select the method for storing and managing private keys/digital certificate pairs and trusted CA certificates. This choice should match with the choice made during SSL configuration.

   *Refer to the chapter 'Choosing the Identity and Trust Stores' in the installation manual 'SSL Configuration on Weblogic'.*

7. Provide the following details under 'Identity' section:

   - Custom Identity Keystore File Name: The fully qualified path to the identity keystore. For example, you may specify 'D:\keystore\ssl\client.jks'.

ORACLE

- Custom Identity Keystore Type: Set this attribute to JKS, which is the type of the keystore. If you leave it blank, it will be defaulted to JKS (Java KeyStore).
- Custom Identity Keystore PassPhrase: This is the password that you will enter when reading or writing to the keystore. Depending on the type of keystore, you can decide whether to define this passphrase or not. A passphrase is required in order to write to any keystore. However, passphrase is not required to read from certain keystores. WebLogic server reads from the keystore, but does not write to it. Hence depending on the requirements of the keystore, you can decide whether to define a passphrase or not.

8. Provide the following details under 'Trust' section:

- If you choose 'Java Standard Trust', specify the password to access the trust store.
- If you choose 'Custom Trust', provide the following attributes:
  - ➢ Custom Trust Keystore: The fully qualified path to the trust keystore.
  - ➢ Custom Trust Keystore Type: Set this attribute to JKS, which is the type of the keystore. If you leave this blank, it is defaulted to JKS (Java KeyStore).
  - ➢ Custom Trust Keystore Passphrase: This is the password that you will enter when reading or writing to the keystore. Depending on the type of keystore, you can decide whether to define this passphrase or not. A passphrase is required in order to write to any keystore. However, passphrase is not required to read from certain keystores. WebLogic server reads from the keystore, but does not write to it. Hence depending on the requirements of the keystore, you can decide whether to define a passphrase or not.

🛑 When identity and trust stores are of JKS format, passphrases are not required.

9. Import the branch certificate into Java Standard Trust or Custom Trust based on the choice made in previous step. Branch certificate can be imported as mentioned below.

   keytool -import -trustcacerts -v -file D:\keystore\ssl\server.cer -alias server -keystore C:\Oracle\Middleware\jdk160_14_R27.6.5-32\jre\lib\security\cacerts

**Setting SSL Attributes for Private Key Alias and Password**

To configure the private key alias and password, follow the steps given below:

1. Go to 'Change Center' of the Administration Console and click 'Lock & Edit'.

2. Expand the 'Servers' node.

3. Select the name of the server for which you want to configure the keystores. For example, 'exampleserver'.

4. Under 'Configuration', select the 'Keystores' tab.

**ORACLE**
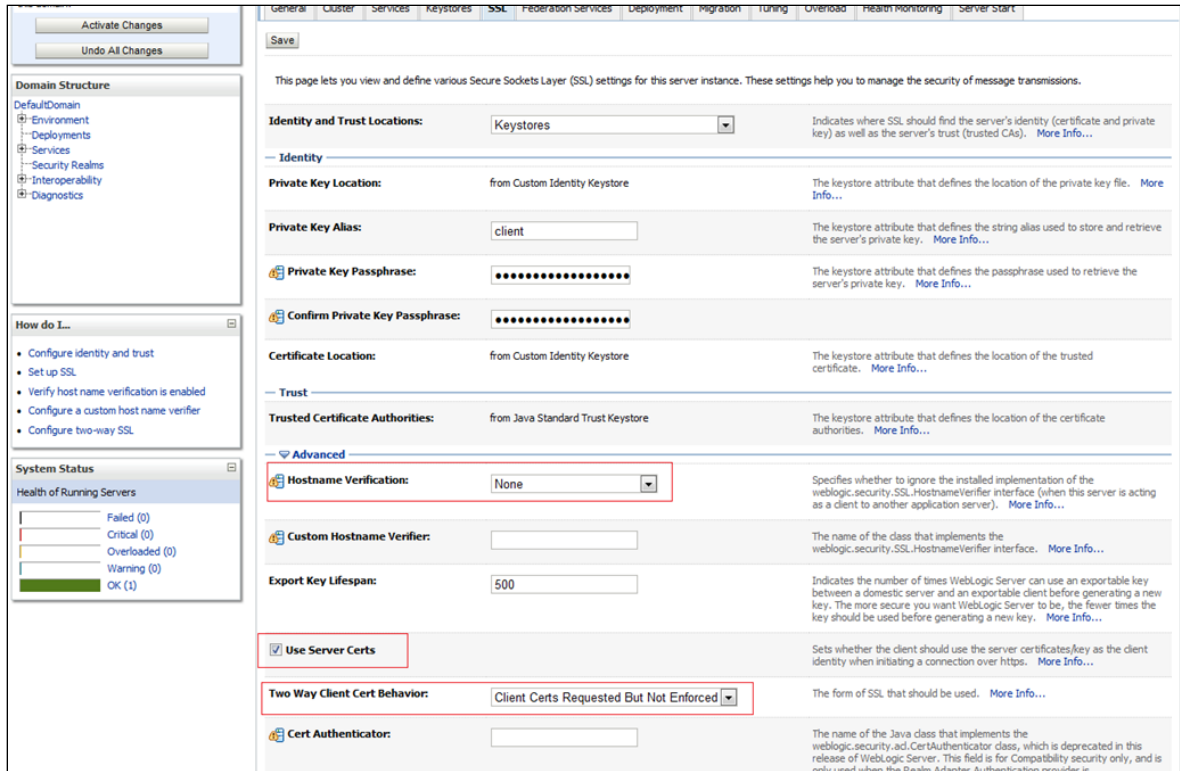
5. Specify the following details:

**Identity and Trust Locations**

Select 'Keystores'.

## Identity

### Private Key Alias

Specify the private key alias. This is the alias name defined for the key pair when creating the key pair in the identity keystore.

### Private Key Passphrase

Specify the private key passphrase. This is the password defined for the key pair (alias_password) at the time of its creation. Confirm the password by entering it again.

## Advanced

### Use Server Certs

Check this box.

### Hostname Verification

Select 'None' from the drop-down list.

ORACLE

**Two Way Client Cert Behavior**

Select 'Client Certs Requested But Not Enforced' from the drop-down list.

6. Click 'Save' to save the details.

7. Under 'Change Center', click 'Activate Changes'.

8. Go to 'Controls' tab, check the appropriate server and click Restart SSL. Click 'Confirm' when you are prompted to confirm.

### 1.4.3.3 <u>Configuring Connector related to Browser and Branch</u>

The browser communicates to the branch through two-way SSL. In order to establish the communication, the browser needs to have its own identity or certificate.

You need to import the 'PKCS12' certificate that was created earlier.

*Refer to the section 'Preparing Client Certificate' in this chapter for details.*

For importing the client certificate into the browser, follow the steps given below.

1. Double click 'client.p12'. The following window is displayed.
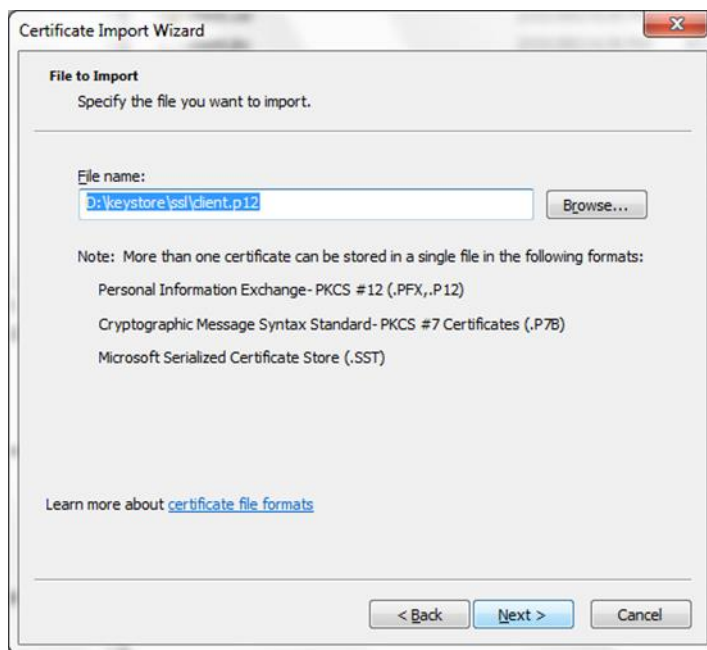


2. Click 'Next'. The following window is displayed.

ORACLE
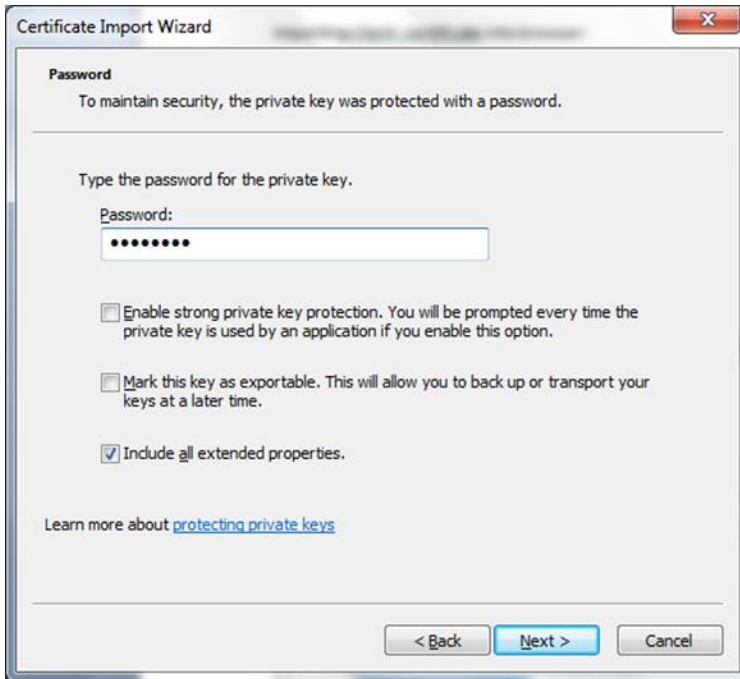
3. Specify the password and click 'Next'. The following window is displayed.
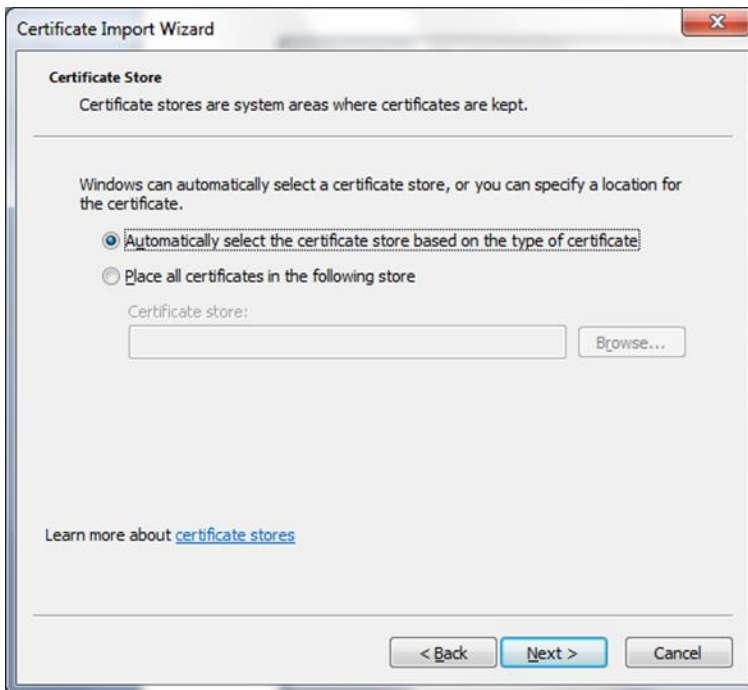


4. Select 'Automatically select the certificate store based on the type of certificate'. Click 'Next'.

5. Click 'Finish' button.