Creating a Client Certificate for Mutual Authentication

If you have a certificate signed by a trusted Certificate Authority (CA) such as Verisign, and the Application Server `cacerts.jks` file already contains a certificate verified by that CA, you do not need to complete this step. You only need to install your certificate in the Application Server certificate file when your certificate is self-signed.

From the directory where you want to create the client certificate, run `keytool` as outlined here. When you press Enter, `keytool` prompts you to enter the server name, organizational unit, organization, locality, state, and country code.

You must enter the **server name** in response to `keytool` 's first prompt, in which it asks for first and last names. For testing purposes, this can be `localhost` . The host specified in the keystore must match the host identified in the `javee.server.host` variable specified in your *`tut-install`*`/javaeetutorial5/examples/bp-project/build.properties` file. If this example is to verify mutual authentication and you receive a runtime error stating that the HTTPS host name is wrong, re-create the client certificate, being sure to use the same host name that you will use when running the example. For example, if your machine name is `duke` , then enter `duke` as the certificate CN or when prompted for first and last names. When accessing the application, enter a URL that points to the same location (for example, `https://duke:8181/mutualauth/hello`). This is necessary because during SSL handshake, the server verifies the client certificate by comparing the certificate name and the host name from which it originates.

To create a keystore named `client_keystore.jks` that contains a client certificate named `client.cer` , follow these steps:

1. Create a backup copy of the server truststore file. To do this,

   A. Change to the directory containing the server's keystore and truststore files, *`as-install`*`\domains\domain1\config` .

   B. Copy `cacerts.jks` to `cacerts.backup.jks` .

   C. Copy `keystore.jks` to `keystore.backup.jks` .

   Do not put client certificates in the `cacerts.jks` file. Any certificate you add to the `cacerts` file effectively means it can be a trusted root for any and all certificate chains. After you have completed development, delete the development version of the `cacerts` file and replace it with the original copy.

2. Generate the client certificate. Enter the following command from the directory where you want to generate the client certificate:

```
 java-home\bin\keytool -genkey -alias client-alias -keyalg RSA -keypass changeit
-storepass changeit -keystore client_keystore.jks
```

3. Export the generated client certificate into the file `client.cer` .

```
 java-home\bin\keytool -export -alias client-alias -storepass changeit
-file client.cer -keystore client_keystore.jks
```

4. Add the certificate to the truststore file *`domain-dir`*`/config/cacerts.jks` . Run `keytool` from the directory where you created the keystore and client certificate. Use the following parameters:

```
 java-home\bin\keytool -import -v -trustcacerts -alias client-alias
-file client.cer -keystore domain-dir/config/cacerts.jks -keypass changeit
-storepass changeit
```

   The `keytool` utility returns a message like this one:

```
Owner: CN=localhost, OU=Java EE, O=Sun, L=Santa Clara, ST=CA, C=US
Issuer: CN=localhost, OU=Java EE, O=Sun, L=Santa Clara, ST=CA, C=US
Serial number: 3e39e66a
Valid from: Thu Jan 30 18:58:50 PST 2005 until: Wed Apr 3019:58:50 PDT 2005
Certificate fingerprints:
    MD5: 5A:B0:4C:88:4E:F8:EF:E9:E5:8B:53:BD:D0:AA:8E:5A
    SHA1:90:00:36:5B:E0:A7:A2:BD:67:DB:EA:37:B9:61:3E:26:B3:89:46:32
Trust this certificate? [no]: yes
Certificate was added to keystore
```

5. Restart the Application Server.