Configuring Java CAPS for SSL Support

Generating a KeyStore and TrustStore

The following sections explain how to create both a KeyStore and a TrustStore (or import a certificate into an existing TrustStore such as the default Logical Host TrustStore in the location:

```
<c:\JavaCAPS>\appserver\domains\<MyDomain>\config\cacerts.jks
```

where `<c:\JavaCAPS>` is the directory where Java CAPS is installed and `<MyDomain>` is the name of your domain. The primary tool used is **keytool**, but **openssl** is also used as a reference for generating **pkcs12** KeyStores.

For more information on **openssl** and available downloads, visit the following web site:

`http://www.openssl.org` .

Creating a KeyStore in JKS Format

This section explains how to create a KeyStore using the JKS format as the database format for both the private key, and the associated certificate or certificate chain. By default, as specified in the java.security file, **keytool** uses JKS as the format of the key and certificate databases (KeyStore and TrustStores). A CA must sign the certificate signing request (CSR). The CA is therefore trusted by the server-side application to which the Adapter is connected.

Note –

It is recommended to use the default KeyStore

```
<c:\JavaCAPS>\appserver\domains\<MyDomain>\config\keystore.jks
```

where `<c:\JavaCAPS>` is the directory where Java CAPS is installed and `<MyDomain>` is the name of your domain.

To Generate a KeyStore

1. Perform the following command.

```
keytool -keystore clientkeystore -genkey -alias client
```

2. Once prompted, enter the information required to generate a CSR. A sample key generation section follows.

```
Enter keystore password: javacaps
What is your first and last name?
[Unknown]: development.sun.com
What is the name of your organizational unit?
[Unknown]: Development
what is the name of your organization?
[Unknown]: Sun
What is the name of your City or Locality?
[Unknown]: Monrovia
What is the name of your State or Province?
[Unknown]: California
What is the two-letter country code for this unit?
[Unknown]: US
Is<CN=development.sun.com, OU=Development, O=Sun, L=Monrovia, ST=California,
C=US> correct?
[no]: yes

Enter key password for <client>
    (RETURN if same as keystore password):
```

   If the KeyStore password is specified, then the password must be provided for the adapter.

3. Press RETURN when prompted for the key password (this action makes the key password the same as the KeyStore password).

   This operation creates a KeyStore file **clientkeystore** in the current working directory. You must specify a fully qualified domain for the “first and last name” question. The reason for this use is that some CAs such as VeriSign expect this properties to be a fully qualified domain name.

   There are CAs that do not require the fully qualified domain, but it is recommended to use the fully qualified domain name for the sake of portability. All the other information given must be valid. If the information cannot be validated, a CA such as VeriSign does not sign a generated CSR for this entry.

   This KeyStore contains an entry with an alias of **client**. This entry consists of the generated private key and information needed for generating a CSR as follows:

```
keytool -keystore clientkeystore -certreq -alias client -keyalg rsa -file client.csr
```

   This command generates a certificate signing request which can be provided to a CA for a certificate request. The file **client.csr** contains the CSR in PEM format.

   Some CA (one trusted by the web server to which the adapter is connecting) must sign the CSR. The CA generates a certificate for the corresponding CSR and signs the certificate with its private key. For more information, visit the following web sites:

http://www.thawte.com

or

http://www.verisign.com

If the certificate is chained with the CA's certificate, perform step 4; otherwise, perform step 5 in the following list:

4. Perform the following command.

   ```
   keytool -import -keystore clientkeystore -file client.cer -alias client
   ```

   The command imports the certificate and assumes the client certificate is in the file **client.cer** and the CA's certificate is in the file **CARoot.cer**.

5. Perform the following command to import the CA's certificate into the KeyStore for chaining with the client's certificate.

   ```
   keytool -import -keystore clientkeystore -file CARoot.cer -alias theCARoot
   ```

6. Perform the following command to import the client's certificate signed by the CA whose certificate was imported in the preceding step.

   ```
   keytool -import -keystore clientkeystore -file client.cer -alias client
   ```

   The generated file **clientkeystore** contains the client's private key and the associated certificate chain used for client authentication and signing. The KeyStore and/or **clientkeystore**, can then be used as the adapter's KeyStore.

Creating a KeyStore in PKCS12 Format

This section explains how to create a PKCS12 KeyStore to work with JSSE. In a real working environment, a customer could already have an existing private key and certificate (signed by a known CA). In this case, JKS format cannot be used, because it does not allow the user to import/export the private key through **keytool**. It is necessary to generate a PKCS12 database consisting of the private key and its certificate.

The generated PKCS12 database can then be used as the Adapter's KeyStore. The **keytool** utility is currently lacking the ability to write to a PKCS12 database. However, it can read from a PKCS12 database.

Note –

There are additional third-party tools available for generating PKCS12 certificates, if you want to use a different tool.

For the following example, **openssl** is used to generate the PKCS12 KeyStore:

```
cat mykey.pem.txt mycertificate.pem.txt>mykeycertificate.pem.txt
```

The existing key is in the file **mykey.pem.txt** in PEM format. The certificate is in **mycertificate.pem.txt**, which is also in PEM format. A text file must be created which contains the key followed by the certificate as follows:

```
openssl pkcs12 -export -in mykeycertificate.pem.txt -out mykeystore.pkcs12
-name myAlias -noiter -nomaciter
```

This command prompts the user for a password. The password is required. The KeyStore fails to work with JSSE without a password. This password must also be supplied as the password for the Adapter's KeyStore password.

This command also uses the **openssl pkcs12** command to generate a PKCS12 KeyStore with the private key and certificate. The generated KeyStore is **mykeystore.pkcs12** with an entry specified by the **myAlias** alias. This entry contains the private key and the certificate provided by the **-in** argument. The **noiter** and **nomaciter** options must be specified to allow the generated KeyStore to be recognized properly by JSSE.

Creating a TrustStore

For demonstration purposes, suppose you have the following CAs that you trust: **firstCA.cert, secondCA.cert, thirdCA.cert**, located in the directory **C:\cascerts**. You can create a new TrustStore consisting of these three trusted certificates.

To Create a New TrustStore

1. Perform the following command.

   ```
   keytool -import -file C:\cascerts\firstCA.cert -alias firstCA -keystore myTrustStore
   ```

2. Enter this command two more times, but for the second and third entries, substitute **secondCA** and **thirdCA** for **firstCA**. Each of these command entries has the following purposes:

   - The first entry creates a KeyStore file named **myTrustStore** in the current working directory and imports the **firstCA** certificate into the TrustStore with an alias of **firstCA**. The format of **myTrustStore** is JKS.
   - For the second entry, substitute **secondCA** to import the **secondCA** certificate into the TrustStore, **myTrustStore**.
   - For the third entry, substitute **thirdCA** to import the **thirdCA** certificate into the TrustStore.

Once completed, myTrustStore is available to be used as the TrustStore for the adapter.