Setup a Real SSL Certificate for Tomcat

> This page is the form of quick notes and needs to be rewritten as an article.

Create Local Certificate

In this example we are generating a key called **<your_cert_alias>** and storing it in a brand new keystore called **mywebservices.bin**.

```
su - serveradmin
cd /opt/jre1.6.0_12/bin/
# Create a local Certificate
keytool -genkey -alias <your_cert_alias> -keyalg RSA -keystore
mywebservices.bin
```

This information depends on your company. Note that when creating a cert for a website the first and last name will be the website url.

```
Enter keystore password:  mypassword
What is your first and last name?
  [Unknown]:  mywebservice.myapp.mycompany.com
What is the name of your organizational unit?
  [Unknown]:  My Unit
What is the name of your organization?
  [Unknown]:  My Organization
What is the name of your City or Locality?
  [Unknown]:  My City
What is the name of your State or Province?
  [Unknown]:  My State
What is the two-letter country code for this unit?
  [Unknown]:  CA
```

As a result, a brand new keystore file is generated. You can confirm this,

```
keytool -keystore mywebservices.bin -list

Enter keystore password:  ******

Keystore type: jks
Keystore provider: SUN

Your keystore contains 1 entry

mywebservices, Oct 8, 2009, keyEntry,
Certificate fingerprint (MD5):
02:70:28:DE:A6:BC:0B:5E:3C:FB:BF:B3:68:8F:0F:32
```

The results show 1 entry with the alias name **mywebservices** which contains a single element, a self-signed certificate which is soley there to generate the CSR.

Generate the CSR (Certificate Service Request)

In this step we generate the request for the SSL certificate.

```
# Generate the CSR (Certificate Service Request)
keytool -certreq -keyalg RSA -alias mywebservices -file mywebservices.csr
-keystore mywebservices.bin
# It is important that the cert files be in the webapps directory
mv mywebservices.* /opt/apache/tomcat.0/webapps/
```

Now submit your **mywebservices**.csr to the CA (Certificate Authority).

> There is usally a web form to fill out in this step as you upload the contents. One note here is that if you are creating a public website you should choose a cert type that supports an alias. As such your cert will then be able to support the base domain name and the base domain prefixed by www. For example, you generate the certificate with the name "mywebservice.myapp.mycompany.com" and as you fill in the form you should use "www.mywebservice.myapp.mycompany.com".

Your CA will return a certificate (your keystore file signed by the CA). Depending on the signer they may use different names so it can be confusing. For example the Entrust CA calls your certificate the "Webserver Certificate" even though in this example we are putting this on an application server. In our example we are provided the CA signed certificate as plain text which we then store in a text file calling it mywebservices-signe.cer. Here is what the contents of the plain text might look like,

> To fill in.

Import The CA (Certificate Authority) Certificate(s)

At this point how many CA certificates need to be imported depend on the CA you use. Also, the CA certificates themselves are usually obtained from the respective Certificate Authority.

In a **simple scenario**, one certificate is returned which authenticates the CA's public key. This is a self-signed certificate (that is, a certificate from the CA authenticating its own public key). For example, here is what the chain looks like this from top to bottom.

1. CA Self-Signed Certificate (often called the root certificate)
2. Your Certificate Signed by the CA

In the case that multiple certs are returned, the CA is returning a chain of certificates which might look like this from top to bottom,

1. CA (Root) Self-Signed Certificate (often called the root certificate)
2. CA (A) Key (certificate signed by a CA (A), authenticating the public key of Key of B below in the chain)
3. CA (B) Key (certificate signed by a CA (B), authenticating the public key of Key of C below in the chain)
4. CA (C) Key (certificate signed by a CA (C), authenticating the public key of Key one below in the chain)
5. .... (there can be more)....
6. Your Certificate Signed by the CA

Here is an example of what the Entrust CA returned chain looks like from top to bottom,

1. CA (Root) Entrust Self-Signed Certificate (often called the root certificate)
2. CA (LB1) Key (certificate signed by a CA (Root), authenticating the public key of Key of 1 below in the chain) - Entrust calls this the Cross Certificate
3. Your Certificate Signed by the CA

The key concept of the chain is that you must import all certs in the chain in order from top to bottom. You also should provide a unique alias per cert. This will allow you to update specific certs in the chain when they expire. Using the example of the Entrust CA we would need to import as follows.

```
keytool -import -alias entrust-2048-root -keystore mywebservices.bin
-trustcacerts -file <filename_of_the_chain_certificate>
Certificate was added to keystore # This is the expected response
keytool -import -alias entrust-L1B -keystore mywebservices.bin
-trustcacerts -file <filename_of_the_chain_certificate>
Certificate was added to keystore
```

Take a look inside your keystore and you should see the newly added certs with their unique alias.

```
keytool -keystore mywebservices.bin -list
entrust-l1b, Oct 9, 2009, trustedCertEntry,
Certificate fingerprint (MD5):
C2:DF:86:BD:E4:8B:FF:26:4D:AE:6A:26:1D:7A:70:D9
entrust-2048-root, Oct 9, 2009, trustedCertEntry,
Certificate fingerprint (MD5):
CC:23:87:09:9B:09:3A:6F:5E:62:EB:F4:73:54:E9:28
pkiwebservices, Oct 8, 2009, keyEntry,
Certificate fingerprint (MD5):
02:70:28:DE:A6:BC:0B:5E:3C:FB:BF:B3:68:8F:0F:32
```

Import Your Certificate (Signed Keystore File)

Finally you can import your new Certificate making sure to use the **same** alias on the initial generation, **pkiwebservices** as you are replacing the self-signed cert already in the keystore.

```
keytool -import -alias mywebservices -keystore mywebservices.bin
-trustcacerts -file <filename_of_the_chain_certificate>
Certificate reply was installed in keystore # This is the expected response
```

This action replaces the self-signed certificate with a proper signed certificate. This is considered the first and bottom chain in a chain of certificates.

Configure Tomcat to Use the SSL Certificate

The final step is to configure Tomcat to use SSL.

Resources

This is an abridged and modified version of the following articles,

http://techtracer.com/2007/09/12/setting-up-ssl-on-tomcat-in-3-easy-steps/

http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html

http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.htmlhttp://www.manpagez.com/man/1/keytool/ - section on Certificate Chains was helpful