

# How to configure SSL

## How to configure SSL

This is an overview of how to configure SSL for Jetty, which uses Sun's reference implementation for the Java Secure Sockets Extension (JSSE).

Configuring SSL can be a confusing experience of keys, certificates, protocols and formats, thus it helps to have a reasonable understanding of the basics. The following links provide some good starting points:

- Certificates:
  - [SSL Certificates HOWTO](#)
  - [Mindprod Java Glossary definition](#)
- \*Keytool:
  - [Keytool for Unix](#)
  - [Keytool for Windows](#)
- Other tools:
  - [IBM Keyman](#)
- Open SSL:
  - [OpenSSL HOWTO](#)
  - [FAQ](#)

## The Basics

The following steps are required to configure Jetty for SSL:

[Step 1](#): Generate or obtain a public/private key pair and x509 certificate.

[Step 2](#): Optionally obtain a certificate from a known certificate authority.

[Step 3](#): Load the keys and the certificates into a JSSE Keystore.

[Step 4](#): Configure a JsseListener with the location and passwords for the keystore.

## OpenSSL Versus Keytool

For testing, keytool is probably the simplest way to generate the key and certificate you will need. However, IBMs [keyman](#) is also pretty good and provides a GUI rather than a command line.

The OpenSSL tools can also be used to generate keys and certificates or to convert ones that have been used with Apache or other servers.

The OpenSSL tool suite is commonly used by other servers such as Apache to generate manipulate keys and certificates. So you may already have some keys and certificates created by openssl, or openssl may be more trusted than keytool or some certificate authorities for step 2 may also prefer the formats produced by ssl.

If you want the option of using the same certificate with Jetty or a web server such as Apache not written in Java, you may prefer to generate your private key and certificate with openssl. The Java keytool does not provide options for exporting private keys, and Apache needs the private key. If you create the key and certificate with openssl your non-Java web server will have ready access to it.

## Step 1: Keys and Certificates

The simplest way generate keys and certificates is to use the keytool application that comes with the JDK, as it generates keys and certificates directly into the keystore. See [Step 1a](#).

If you already have keys and certificates, please goto [Step 3](#) to load them into a JSSE key store.

If you have a renewal certificate to replace one that is expiring, take a look at [#renewals](#).

The commands below only generate minimal keys and certificates. You should read the full manuals of the tools you are using if you wish to specify:

- Key size.
- Certificate expiry.
- Alternate security providers.

## Step 1a: Generating a certificate with JDK keytool

The following command will generate a key pair and certificate directly into a keystore:

```
keytool -keystore keystore -alias jetty  
-genkey -keyalg RSA
```

**Note:** DSA key algorithm certificate produces an error after several loading of pages. In browser, it gives you a message "Could not establish an encrypted connection because certificate presented by localhost has an invalid signature." See more details in [troubleshooting](#) page.

This command will prompt for information about the certificate and for passwords to protect both the keystore and the keys within it. The only mandatory response is to provide the fully qualified host name of the server at the "first and last name" prompt. For example:

```
keytool -keystore keystore -alias jetty
-genkey -keyalg RSA
Enter keystore password: password
What is your first and last name?
  [Unknown]: jetty.mortbay.org
What is the name of your organizational
unit?
  [Unknown]: Jetty
What is the name of your organization?
  [Unknown]: Mort Bay Consulting Pty. Ltd.
What is the name of your City or Locality?
  [Unknown]:
What is the name of your State or Province?
  [Unknown]:
What is the two-letter country code for this
unit?
  [Unknown]:
Is CN=jetty.mortbay.org, OU=Jetty, O=Mort
Bay Consulting Pty. Ltd.,
L=Unknown, ST=Unknown, C=Unknown correct?
  [no]: yes

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

You now have the minimal requirements to run an SSL connection and could proceed directly to [Step 4](#) to configure an SSL connector.

However the certificate you have generated will not be trusted by the browser and the user will be prompted to this effect. This is often sufficient for testing, but most public site will need to [Step 2a](#) to obtain a certificate trusted by

most popular clients.

## Step 1b: Keys and Certificates with openssl

The following command generates a key pair in the file jetty.key:

```
openssl genrsa -des3 -out jetty.key
```

You may also wish to use the -rand file argument to provide an arbitrary file to help seed the random number generator.

The following command generates a certificate for the key into the file jetty.crt:

```
openssl req -new -x509 -key jetty.key -out  
jetty.crt
```

This command will prompt for information about the certificate and for passwords to protect both the keystore and the keys within it. The only mandatory response is to provide the fully qualified host name of the server at the "Common Name" prompt. For example:

```
openssl genrsa -des3 -out jetty.key  
Generating RSA private key, 512 bit long  
modulus  
.....++++++  
..++++++  
e is 65537 (0x10001)  
Enter pass phrase for jetty.key:  
Verifying - Enter pass phrase for jetty.key:  
  
# openssl req -new -x509 -key jetty.key -out  
jetty.crt  
Enter pass phrase for jetty.key:  
You are about to be asked to enter
```

information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [AU]:.

State or Province Name (full name)

[Some-State]:.

Locality Name (eg, city) []:.

Organization Name (eg, company) [Internet Widgets Pty Ltd]:Mort Bay Consulting Pty. Ltd.

Organizational Unit Name (eg, section)

[:Jetty

Common Name (eg, YOUR name)

[:jetty.mortbay.org

Email Address []:

#

You now have the minimal requirements to run an SSL connection and could proceed directly to [Step 3](#) to load these keys and certificates into a JSSE keystore. However the certificate you have generated will not be trusted by the browser and the user will be prompted to this effect. This is often sufficient for testing, but most public site will need to [Step 2b](#) to obtain a certificate trusted by most popular clients.

## Step 1c: Keys and Certificates from other sources

If you have keys and certificates from other sources, then you can proceed directly to [Step 3](#).

## Step 2: Request a trusted certificate

The keys and certificates generated in steps 1a and 1b are sufficient to run an SSL connector. However the certificate you have generated will not be trusted by the browser and the user will be prompted to this effect.

To obtain a certificate that will be trusted by most common browsers, you need to request a well known certificate authority (CA) to sign your key/certificate. Such trusted CAs include: AddTrust, Entrust, GeoTrust, RSA Data Security, Thawte, VISA, ValiCert, Verisign, beTRUSTed, among others.

Each CA will have their own instructions which should be followed (look for JSSE or openssl sections), but all will involve a step to generate a certificate signing request (CSR).

### Step 2a: CSR from keytool

The following commands generate the file jetty.csr using keytool for a key/cert already in the keystore:

```
keytool -certreq -alias jetty -keystore
keystore -file jetty.csr
```

### Step 2b: CSR from openssl

The following commands generate the file jetty.csr using openssl for a key in the file jetty.key:

```
openssl req -new -key jetty.key -out  
jetty.csr
```

Note that this command only uses the existing key from `jetty.key` file and not a certificate in `jetty.crt` generated by step 1b. The details for the certificate need to be entered again.

### Step 3: Loading Keys and Certificates

Once a CA has sent you a certificate, or if you generated your own certificate without keytool, then it will need to be loaded into a JSSE keystore. If you did not use keytool to generate the key, then it will also need to be loaded into the keystore.

#### Combined Private Key and Certificate

You need both the private key and the certificate in the keystore. So the certificate should be loaded into the keystore used to generate the CSR (step 2a). If your key pair is not in a keystore (eg if generated as step 1b), then you will need to use the PKCS12 format to load both key and certificate as in step 3b.

#### Step 3a: Loading Certificates with keytool

A certificate in PEM form may be directly loaded into a keystore with keytool. The PEM format is a text encoding of certificates and is produced by openssl (as in step 1b) and is returned by some CAs. An example PEM file is:

jetty.crt

-----BEGIN CERTIFICATE-----

MIICSDCCAfKgAwIBAgIBADANBgkqhkiG9w0BAQQFADBUMSYwJAYDVQQKEEx1Nb3J0

IEJheSBDb25zdWx0aW5nIFB0eS4gTHRkLjEOMAwGA1UECzMFSmV0dHkxGjAYBgNV

BAMTEWpldHR5Lm1vcnRiYXkub3JnMB4XDTAzMdQwNjEzMTk1MFoXDTAzMDUwNjEz

MTk1MFowVDEmMCQGA1UEChMdTW9ydCBCYXkgQ29uc3Vs dGluZyBQdHkuIEEx0ZC4x

DjAMBgNVBAsTBUpIdHR5MR0wGAYDVQQDEXFqZXR0eS5t b3J0YmF5Lm9yZzBcMA0G

CSqGSIB3DQEBAQUAA0sAMEgCQQC5V4oZeVdhhdhHqa9L2 /ZnKySPWUqqy81riNfAJ

7uALW0kEv/Lt1G34d00cVvt/PK8/bU4dlolnJx1SpiMZ bKsFAGMBAAGjga4wgasw

HQYDVR0OBByEFFV1gbB1XRvUx1UofmifQJS/MCYwMHwG A1UdIwR1MHOAFFV1gbB1

XRvUx1UofmifQJS/MCYwoVikVjBUMSYwJAYDVQQKEEx1N b3J0IEJheSBDb25zdWx0

aW5nIFB0eS4gTHRkLjEOMAwGA1UECzMFSmV0dHkxGjAY BgNVBAMTEWpldHR5Lm1v

cnRiYXkub3JnggEAMAwGA1UdEwQFMAMBAf8wDQYJKoZI hvcNAQEEBQADQQA6NkaV

OtXzP4ayzBcgK/qSCmF44jdcARmrXhiXUCXzjxsLjsJe YPJoJhUdC2LQKy+p4ki8

Rcz6oCRvCGCe5kDB

-----END CERTIFICATE-----

The following command will load a PEM encoded certificate in the jetty.crt file into a JSSE keystore:

```
keytool -keystore keystore -import -alias  
jetty -file jetty.crt -trustcacerts
```

Depending on the situation you may not require the `-trustcacerts` option. Try the operation without it if you like. If the certificate you receive from the CA is not in a format that keytool understands, then the openssl command can be used to convert formats:

```
openssl x509 -in jetty.der -inform DER  
-outform PEM -out jetty.crt
```

### Step 3b: Loading keys and certificates via PKCS12

If you have a key and certificate in separate files, they need to be combined into a PKCS12 format file to be loaded into a new keystore. The certificate can be one you generated yourself or one that has been returned from a CA in response to your CSR.

The following openssl command will combine the keys in jetty.key and the certificate in the jetty.crt file into the jetty.pkcs12 file:

```
openssl pkcs12 -inkey jetty.key -in  
jetty.crt -export -out jetty.pkcs12
```

If you have a chain of certificates, because your CA is an intermediary, build the pkcs12 file like this:

```
# cat example.crt intermediate.crt  
[intermediate2.crt]... rootCA.crt >  
cert-chain.txt  
# openssl pkcs12 -export -inkey example.key  
-in cert-chain.txt -out example.pkcs12
```

The order of certificates must be from server to rootCA, as per RFC2246 section 7.4.2.

OpenSSL is going to ask you for an "export password". A non-empty password seems to be required to make the next step work. The resulting PKCS12 file may be loaded into a JSSE keystore with the following jetty utility class:

```
java -classpath
$JETTY_HOME/lib/jetty-util-6.1-SNAPSHOT.jar:
$JETTY_HOME/lib/jetty-6.1-SNAPSHOT.jar
org.mortbay.jetty.security.PKCS12Import
jetty.pkcs12 keystore
```

This asks for two passphrases. Give the password from the last step as the input passphrase and you are set. The "output passphrase" will need to appear in your jetty.xml config file as both the Password and KeyPassword of the SunJsseListener that will use the certificate.

We may also use keytool (starting from jdk1.6) to import pkcs12 file with the following command:

```
keytool -importkeystore -srckeystore
jetty.pkcs12 -srcstoretype PKCS12
-destkeystore keystore
```

## Step 4: Configure Jetty

```

<Call name="addConnector">
  <Arg>
    <New
class="org.mortbay.jetty.security.SslSocketC
onconnector">
      <Set name="Port">8443</Set>
      <Set name="maxIdleTime">30000</Set>
      <Set name="keystore"><SystemProperty
name="jetty.home" default="."
/>/etc/keystore</Set>
      <Set
name="password">OBF:1vny1zlo1x8e1vnw1vn61x8g
1zlu1vn4</Set>
      <Set
name="keyPassword">OBF:1u2u1wml1z7s1z7a1wn1l
u2g</Set>
      <Set
name="truststore"><SystemProperty
name="jetty.home" default="."
/>/etc/keystore</Set>
      <Set
name="trustPassword">OBF:1vny1zlo1x8e1vnw1vn
61x8g1zlu1vn4</Set>
    </New>
  </Arg>
</Call>

```

Remember that the default port for https is 443 not 80, so change 8443 to 443 if you want to be able to use URL's without explicit port numbers. For a production site it normally makes sense to have a HttpListener on port 80 and a SunJsseListener on port 443. Note that as these are privileged ports, you may want to use a redirection mechanism

to map port 80 to eg 8080 and 443 to eg 8443. For details on this, see the [Running Jetty as a non-root user](#). The keystore file in this example is given relative to the jetty home directory. For production, choose a private directory with restricted access to keep your keystore in. Even though it has a password on it, the password may be configured into the runtime environment so is vulnerable to theft.

Jetty can now be started the normal way (make sure that jcert.jar, jnet.jar and jsse.jar are on your classpath) and SSL can be used with a URL like:

```
https://localhost:8443/
```

**Note:** The most common mistake at this point is to try to access port 8443 with http rather than https. If CONFIDENTIAL or INTEGRAL security constraints are being used, then you should also configure the normal HTTP connector with which port to use for SSL:

```
<Call name="addConnector">
  <Arg>
    <New
      class="org.mortbay.jetty.nio.SelectChannelCo
      nconnector">
        <Set name="port">8080</Set>
        <Set
          name="maxIdleTime">30000</Set>
        <Set name="Acceptors">2</Set>
        <Set
          name="confidentialPort">8443</Set>
        </New>
      </Arg>
    </Call>
```

## Password Issues

If the passwords are not provided in the configuration, they may be provided as java properties (jetty.ssl.password and jetty.ssl.keypassword) else they will be prompted for.

Remember that putting your password on the command line is a security risk. They can also be set as properties within the config file, but this risks accidental discovery by developers.

If jetty is given a password that begins with "OBF:" it is treated as an obfuscated password. Passwords can be obfuscated by running [org.mortbay.jetty.security.Password](#) as a main class. This can protect passwords from casual observation.

## Renewing Certificates

If you are updating your configuration to use a newer certificate, as when the old one is expiring, just do [Step 3](#). If you imported the key and certificate originally using the PKCS 12 method, use an alias of "1" rather than "jetty", because that is the alias the PKCS12 process enters into the keystore.