# How to install a new SSL certificate?

January 19, 2018 01:00

## Issue

You want to add a SSL certificate ("certX") for the following cases:
1. non-trusted (self-signed) certificate
2. trusted certificate provided by CA that isn't included in the default JRE `keystore`

For several security features that you want to use over a secure connection. Some examples because you would need to add the mentioned certificate are:

- Connecting to Jenkins a secure service (SSL/TLS). As an example an Active Directory or LDAP
- Accessing Jenkins to a remote HTTPS resource
- Configuring HTTPS for CloudBees Jenkins Enterprise via haproxy

## Environment

- CloudBees Jenkins Team (CJT)
- CloudBees Jenkins Platform - Client Master (CJPCM)
- CloudBees Jenkins Platform - Operations Center
- Jenkins LTS
- haproxy [optional]

## Resolution

### A. Locate "certX" (optional)

In most cases, please reach out to your operations team for the necessary "certX" files.

If configuring HA and you need to download the SSL server certificate (CloudBees Jenkins Operations Center, haproxy virtualmachine, etc), use a tool such as:

- `openssl`

```
> openssl s_client -connect <SERVER_HOSTNAME>:443
```

- `keytool`

```
> keytool -printcert -rfc -sslServer <SERVER_HOSTNAME>
```

- `gnutls-cli`

```
> gnutls-cli --print-cert --insecure <SERVER_HOSTNAME>
```

**Note**: Embedded in the response is the certificate identifiable by the fragment starting with `-----BEGIN CERTIFICATE-----` and ending with `-----END CERTIFICATE-----`. Store the certificate in a file "path/to/.pem", including `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----`.

### Example

```
> openssl s_client -connect www.example.com:443
CONNECTED(00000003)
depth=1 /C=US/O=GeoTrust Inc./CN=GeoTrust SSL CA - G3
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
 0 s:/C=US/ST=California/L=San Francisco/O=Example Technologies, Inc/CN=*.example.com
...
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIE7jCCA9agAwIBAgIQJ85dBpYNN5a56Pa7AA0t6TANBgkqhkiG9w0BAQsFADBE
...
ggLk2IYTdtzZsxYK96maAwmg
-----END CERTIFICATE-----
subject=/C=US/ST=California/L=San Francisco/O=Example Technologies, Inc/CN=*.example.com
```

```
issuer=/C=US/O=GeoTrust Inc./CN=GeoTrust SSL CA - G3
---
No client certificate CA names sent
---
SSL handshake has read 2513 bytes and written 456 bytes
---
New, TLSv1/SSLv3, Cipher is AES256-SHA
...
---
closed
```

Best practice: Download the certificate, transform to an x509 format and then save it to a file

For instance, using `openssl` tool on Unix and saving it into PEM format would be like:

```
openssl s_client -showcerts -connect www.example.com:443 </dev/null 2>/dev/null|openssl x509 -outform PEM > /opt/Labs/resources/certs/exam
```

## B. Adding “certX” to the keystore

To use “certX”, you have several options:

1. Adding it to a fresh `keystore`
2. Adding it to a copy of an existing `keystore`
3. Adding it to the existing `keystore`

By default, Java Applications (as Jenkins) make use the **JVM `keystore`**. If a Java Applications needs to make use of a custom `keystore`, it needs to be configured so.

**Notes:**

- Default password of the JVM `keystore` is `changeit` (or `changeme`).
- Modifications on the `keystore` will be deleted on every JVM update.

**Best practice:** *Option 2.* of the JVM `keystore` for the following reasons:

- Your `keystore` would contains certificates trusted by your JVM.
- Certificates needed by Jenkins would be stored into a single place.
- In case of JVM update, your `keystore` would not be overridden.

### Procedure

For the following steps we assume the following points:

- You have a “certX” located and available
- This `keystore` is custom and it is created by copying the existing JVM `keystore` into a new location within `JENKINS_HOME`, including a new cert (“certX”)
- This `keystore` is not included yet into `JENKINS_JAVA_ARGS`
- `JENKINS_HOME` and `JAVA_HOME` env variables need to be defined

###### 1. Create a custom keystore from the JVM `keystore`

Once you have logged with the `jenkins` user:

For Unix:

```
CUSTOM_KEYSTORE=$JENKINS_HOME/.keystore/
mkdir -p $CUSTOM_KEYSTORE
cp $JAVA_HOME/jre/lib/security/cacerts $CUSTOM_KEYSTORE
```

For Windows:

```
CUSTOM_KEYSTORE=%JENKINS_HOME%\.keystore\
md  %CUSTOM_KEYSTORE%
copy %JAVA_HOME%\jre\lib\security\cacerts %CUSTOM_KEYSTORE%
```

###### 2. Import your certificate:

For Unix:

```
$JAVA_HOME/bin/keytool -keystore $JENKINS_HOME/.keystore/cacerts \
  -import -alias <YOUR_ALIAS_HERE> -file <YOUR_CA_FILE>
```

For Windows:

```
%JAVA_HOME%\bin\keytool -keystore %JENKINS_HOME%\.keystore\cacerts -import -alias <YOUR_ALIAS_HERE> -file <YOUR_CA_FILE>
```

**Note**:
1. At this point, you will be asked for the keystore password.
2. When prompted `Trust this certificate? [no]:` enter yes to confirm the key import:

3. Add the certificate to the Jenkins startup parameters:

The following JAVA properties should be added depending on your OS:

For Unix:

```
-Djavax.net.ssl.trustStore=$JENKINS_HOME/.keystore/cacerts \
-Djavax.net.ssl.trustStorePassword=changeit
```

For Windows:

```
-Djavax.net.ssl.trustStore=%JENKINS_HOME%\.keystore\cacerts
-Djavax.net.ssl.trustStorePassword=changeit
```

Follow instructions on How to add Java arguments to Jenkins for your particular case.

4. You must restart Jenkins for the parameters to take effect.

### Import certificates in Linux OS

In some cases we need also to import the certificate in the OS to use it with CURL, to do that you have to follow these steps based on your Linux distribution.

### Ubuntu

- Obtain the server certificate and the certificates chain need to import **(in PEM format)**
- Copy your certificates in `/usr/share/ca-certificates` directory
- Update your certificates running the command `sudo update-ca-certificates --fresh`

```
sudo openssl s_client -showcerts -connect server.example.com:443 </dev/null 2>/dev/null|openssl x509 -outform PEM > /tmp/server_example_co
sudo cp /tmp/server_example_com.pem /usr/share/ca-certificates/server_example_com.pem
sudo update-ca-certificates
```

### RHLE/CentOS

- Obtain the server certificate and the certificates chain need to import **(in PEM format)**
- Copy your certificates in `/etc/pki/ca-trust/source/anchors/`
- Update your certificates running the command `sudo update-ca-trust extract`
- On RHLE6 you also have to enable the CAs `sudo update-ca-trust enable`

```
sudo openssl s_client -showcerts -connect server.example.com:443 </dev/null 2>/dev/null|openssl x509 -outform PEM > /tmp/server_example_co
sudo cp /tmp/server_example_com.pem /etc/pki/ca-trust/source/anchors/server_example_com.pem
sudo update-ca-trust extract
sudo update-ca-trust enable
```

## Troubleshooting

SSL Certificates Troubleshooting

## References

- Oracle Import the Certificate as a Trusted Certificate
- Oracle Keytool JavaSE 8

- Oracle JavaSE 8 Security Customizing

Was this article helpful? 3 out of 4 found this helpful

Have more questions? Submit a request

2 Comments

**Date** Votes

0

**Matthew Magee** April 08, 2016 07:23

I'm not following these instructions; I'm also very new to CloudBees/Jenkins.

We need to enable SSL between the load balancer and the application.

I've got a .cert, .chain, and a .key file given to me that was created by our centrify server.

Can a working example be given on how to make this work?

1

**Michael Ray** April 18, 2016 11:51

"The correct place for these on debian is in /etc/defaults/jenkins for a Master and /etc/defaults/jenkins-oc for an Operations Center." - James Brown

Please sign in to leave a comment.