Unimus HTTPS with a CSR

If you want to use Unimus with HTTPS using a CSR to generate a CA-signed certificate, the process is similar to Unimus HTTPS with a self-signed cert.

On Linux:

Set "keytool" path

```
JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
KEYTOOL=$JAVA_HOME/bin/keytool
```

Ensure that *JAVA_HOME* points to the correct Java installation directory for your system. If you are using a different version of Java, adjust the path accordingly.

Create a new keystore, with a corresponding private key

```
cd /opt/unimus
$KEYTOOL -genkeypair -alias UnimusHttpsCert -keyalg RSA -keysize 4092
-keystore keystore.p12 -storetype PKCS12
```

The command above will ask for a keystore password (remember this for later), as well as other parameters of the certificate.
The details provided here will be present in the certificate.

Create a CSR

```
$KEYTOOL -certreq -alias UnimusHttpsCert -keyalg RSA -file unimus-csr.pem
-keystore keystore.p12
```

This will generate the "unimus-csr.pem" CSR file.

Send the CSR for signing

You can now send the "unimus-csr.pem" CSR to your CA for signing.
Once your CA returns a valid certificate, you can continue.

Import cert

Once you have your cert file from the CA, you can run:

```
$KEYTOOL -importcert -alias UnimusHttpsCert -file cert.pem -keystore
keystore.p12
```

Modify Unimus service configuration

After the CA-signed cert was imported into the keystore, you can use it to run Unimus' web server in HTTPS.
This is the same as if using Unimus HTTPS with a self-signed cert, please check that article for more info:

```
vim /etc/default/unimus

# append behind existing config, all on same line
-Dserver.ssl.key-store=/opt/unimus/keystore.p12
-Dserver.ssl.keyStoreType=PKCS12 -Dserver.ssl.keyAlias=UnimusHttpsCert
-Dserver.ssl.key-store-password=[insert password here]
```

Note: When running Unimus in Docker, add the Java parameters to your *docker-compose* file and persist the volume containing the keystore file to ensure the SSL certificate is retained if the container is restarted or recreated:

```
services:
  unimus:
    image: croc/unimus
    environment:
      - 'JAVA_OPTS=-Dserver.ssl.key-store=/opt/unimus/keystore.p12
-Dserver.ssl.keyStoreType=PKCS12 -Dserver.ssl.keyAlias=UnimusHttpsCert
-Dserver.ssl.key-store-password=[insert password here]'
    volumes:
      - './unimus:/opt/unimus'
```

Restart the Unimus service (or container). After startup, Unimus will be available over HTTPS.

On Windows:

On Windows, you can usually generate the CSR using the Windows built-in Certificate Manager. This CSR can then be signed by your Windows CA.

Once the CSR is signed, you will usually import the certificate into your local Trust Store. You can then export both the cert and its key into a .pfx store.
Once you have a .pfx store with both the cert and the key present, you can convert that into the .p12 format Unimus needs using:

```
set JAVA_HOME=C:\Program Files\Unimus\jre17
set KEYTOOL=%JAVA_HOME%\bin\keytool

cd "C:\Program Files\Unimus"
keytool -list -storetype PKCS12 -keystore your-pfx-file-path-here.pfx
-storepass keystore_password
```

Note the **alias** of the certificate you want to use. Then you can run:

```
keytool -import -alias alias_here -file your-pfx-file-path-here.pfx
-keystore unimus.keystore.p12 -storetype PKCS12 -storepass
keystore_password
```

After you generate the .p12 store, you can use it in Unimus' config file like this:

```
-Dserver.ssl.key-store="C:\\Program Files\\Unimus\\unimus.keystore.p12"
-Dserver.ssl.keyStoreType=PKCS12
-Dserver.ssl.keyAlias=[insert alias here]
-Dserver.ssl.key-store-password=[insert password here]
```