SSL/TLS Certificate Generation

Last updated: 13/04/2016

Table of contents

1 Introduction

In order to configure Lightstreamer Server in HTTPS/WSS mode, an SSL/TLS certificate is required. This tutorial shows an example of the tasks needed to create an appropriate keystore, using the tools provided by the Java Development Kit.

The demo keystore created through this example, named *myserver.keystore*, is already available in the *conf* directory of Lightstreamer distribution.

2 Procedures

2.1 Creation and Installation

➔ Generate a new keystore, called *myserver.keystore*, where the key pair is identified by the "LS" alias (you are free to change it). In this example the target hostname is "push.mycompany.com":

```
keytool -genkey -alias LS -keystore myserver.keystore -keypass
mypassword -storepass mypassword -keyalg "RSA" -keysize 1024
-validity 365 -dname "CN=push.mycompany.com, O=MYCOMPANY INC.,
L=MyCity, ST=MyState, C=MyCountry"
```

NOTE: "keypass" is the password of the key pair; "storepass" is the password of the keystore. The two passwords must be the same[1].

Of course you are free to change any parameters to meet your requirements in terms of algorithms, validity, and contact details.

At this point, the *myserver.keystore* file can already be used with Lightstreamer Server for test purpose. But since it is a self-signed keystore, it will raise a security alert in any browser. So, for production environment it is necessary to have the certificate signed by a Certification Authority. The following steps show how to accomplish this goal.

➔ Create the Certificate Signing Request (CSR) to be delivered to a Certification Authority (CA):

```
keytool -certreq -alias LS -keystore myserver.keystore -keypass
mypassword -storepass mypassword -file myserver.req
```

The *myserver.req* file is created. It must be sent to a CA to be signed.

NOTE: Some CA may require that in the previous step (i.e. the keystore generation) you have set the "CN" attribute to a proper personal identification key, instead of the target hostname, as a condition for them to accept the CSR. Obviously, you have to supply the target hostname to the CA in some other part of the interaction, as determined by the CA. The CA will take care of providing a signed certificate with the "CN" attribute set to the correct hostname.

➔ When the CA sends the certificate back, it is necessary to import the certificate chain provided by the CA (not including the final certificate) in *myserver.keystore*.

```
keytool -import -alias int1 -keystore myserver.keystore -keypass
mypassword -storepass mypassword -file SuperCertIntermediateCA.crt

keytool -import -alias int2 -keystore myserver.keystore -keypass
mypassword -storepass mypassword -file
VeriSign.Class3.Public.Primary.CA.cer
```

In this example, the certificate provided by the CA is called *push.mycompany.com.crt*, which contains the following certification chain: "Thawte SGC CA" and "VeriSign Class 3 Public Primary CA".

[1] For example see https://search.thawte.com/support/ssl-digital-certificates/index?page=content&id=SO832

The first certificate in the chain is the *SuperCertIntermediateCA.crt* file, the second is the *VeriSign.Class3.Public.Primary.CA.cer* file. To extract any certificate from a chain, open the main .crt file (*push.mycompany.com.crt*) in Windows by double clicking, select an intermediate certificate under the "Certification Path" tab, display its details, then copy it on file. An export wizard will open. Choose the binary X.509 format coded with DER and the file name (e.g. *VeriSign.Class3.Public.Primary.CA.cer*).

Make sure that the returned .crt file (*push.mycompany.com.crt*) includes all the needed intermediate certificates. Otherwise you may need to acquire them in another way; in fact, all intermediate certificates are public and should be made available by the CA also directly.

➔ After importing the intermediate certificates of the chain, the final certificate can be imported too:

```
keytool -import -alias LS -keystore myserver.keystore -keypass
mypassword -storepass mypassword -file push.mycompany.com.crt
-trustcacerts
```

Make sure you use the same alias you used with the "-genkey" command.

The *myserver.keystore* file is now ready to be used with Lightstreamer Server.

2.2 Conversion of an Existing Certificate Chain Available in a Different Format

The keytool utility supports conversions from different keystore formats into the Java format (JKS), through the "-importkeystore" command.

This enables, for instance, the reuse of existing certificate chains stored in the common PKCS12 format. Other formats might be supported by keytool as well.

➔ Case 1: If *mycerts.p12* is a container file in PKCS12 format (the .pfx extension is also commonly used for such files), an equivalent JKS keystore can be created through:

```
keytool -importkeystore -destkeystore myserver.keystore
-deststorepass mypassword -srckeystore mycerts.p12 -srcstoretype
PKCS12
```

The command will prompt you for the password associated to *mycerts.p12*.
In the above command, “mypassword” is the password to be assigned to the new keystore; it should be the same as the password associated to *mycerts.p12*.

The *myserver.keystore* file is now ready to be used with Lightstreamer Server.

➔ Case 2: openssl was used to generate a key pair and the related certification request. So a *MYKEY.key* in PEM format has been produced and a *MYCERT.crt* public key certificate has been received back from the Certification Authority. Furthermore, the CA should have supplied a root and one or more intermediate certificates (in PEM format).
Make sure that all the needed intermediate certificates have been received. Otherwise you may need to acquire them in another way; in fact, all intermediate certificates are public and should be made available by the CA also directly.

Now you could import all of them into a JKS keystore by creating a PKCS12 container file through openssl first.
The openssl command has the form:

```
openssl pkcs12 -export -in MYCERT.crt -inkey MYKEY.key -out
mycerts.p12 -name LS -CAfile MYCACERT.crt -caname myCAname -chain
```

The command will prompt you for a password to be associated to *mycerts.p12*.
Note that *MYCACERT.crt* file must be the chain of the root and all the intermediate certificates of your CA. If you have these in different PEM files you can create the *MYCACERT.crt* file as:

MYCACERT.crt = <concatenation of the encoded public key certificates>

you must order the certificates such that the root certificate is the last certificate in the chain.

With the generated *mycerts.p12* file, you can now fall into the previous case 1.

2.3 Utilities

➔ The certificates included in a keystore can be listed with the following command:

```
keytool -list -keystore myserver.keystore -storepass mypassword
```

➔ Each certificate in the keystore can be extracted in a readable format with the following command:

```
keytool -export -alias int1 -keystore myserver.keystore -storepass
mypassword -file int1.cer
```