HTTPS Configuration

This guideline will show you on how to run Ambience 4.x in HTTPS mode using a self-signed certificate and this can be done by using openSSL keystore.

Prerequisites: Verify that OpenSSL installed on the system. For windows you can get it from https://slproweb.com/products/Win32OpenSSL.html

1/ Below are the three files that need to be add in your "jetty\elx-base\etc" path of your Ambience directory.

- jetty-https.xml
- jetty-ssl.xml
- jetty-ssl-context.xml

jetty-https.xml

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN" "http://www.eclipse.org/jetty/configure_9_3.dtd">

<!-- ============================================================= -->
<!-- Configure a HTTPS connector.                                  -->
<!-- This configuration must be used in conjunction with jetty.xml -->
<!-- and jetty-ssl.xml.                                            -->
<!-- ============================================================= -->
<Configure id="sslConnector" class="org.eclipse.jetty.server.ServerConnector">

  <Call name="addIfAbsentConnectionFactory">
    <Arg>
      <New class="org.eclipse.jetty.server.SslConnectionFactory">
        <Arg name="next">http/1.1</Arg>
        <Arg name="sslContextFactory"><Ref refid="sslContextFactory"/></Arg>
      </New>
    </Arg>
  </Call>

  <Call name="addConnectionFactory">
    <Arg>
      <New class="org.eclipse.jetty.server.HttpConnectionFactory">
        <Arg name="config"><Ref refid="sslHttpConfig" /></Arg>
        <Arg name="compliance"><Call class="org.eclipse.jetty.http.HttpCompliance" name="valueOf"><Arg><Property
name="jetty.http.compliance" default="RFC7230"/></Arg></Call></Arg>
      </New>
    </Arg>
  </Call>

</Configure>
```

jetty-ssl.xml

```
<!-- =========================================================== -->
<!-- Base SSL configuration                                      -->
<!-- This configuration needs to be used together with 1 or more  -->
<!-- of jetty-https.xml or jetty-http2.xml                       -->
<!-- =========================================================== -->
<Configure id="Server" class="org.eclipse.jetty.server.Server">

  <!-- =========================================================== -->
  <!-- Add a SSL Connector with no protocol factories              -->
  <!-- =========================================================== -->
  <Call  name="addConnector">
    <Arg>
      <New id="sslConnector" class="org.eclipse.jetty.server.ServerConnector">
        <Arg name="server"><Ref refid="Server" /></Arg>
        <Arg name="acceptors" type="int"><Property name="jetty.ssl.acceptors"
deprecated="ssl.acceptors"default="-1"/></Arg>
        <Arg name="selectors" type="int"><Property name="jetty.ssl.selectors"
deprecated="ssl.selectors"default="-1"/></Arg>
        <Arg name="factories">
          <Array type="org.eclipse.jetty.server.ConnectionFactory">
            <!-- uncomment to support proxy protocol
            <Item>
              <New class="org.eclipse.jetty.server.ProxyConnectionFactory"/>
            </Item>-->
          </Array>
        </Arg>

        <Set name="host"><Property name="jetty.ssl.host" deprecated="jetty.host" /></Set>
        <Set name="port"><Property name="jetty.ssl.port" deprecated="ssl.port" default="8443" /></Set>
        <Set name="idleTimeout"><Property name="jetty.ssl.idleTimeout" deprecated="ssl.timeout"
default="30000"/></Set>
        <Set name="acceptorPriorityDelta"><Property name="jetty.ssl.acceptorPriorityDelta"
deprecated="ssl.acceptorPriorityDelta" default="0"/></Set>
        <Set name="acceptQueueSize"><Property name="jetty.ssl.acceptQueueSize"
deprecated="ssl.acceptQueueSize" default="0"/></Set>
        <Get name="SelectorManager">
          <Set name="connectTimeout"><Property name="jetty.ssl.connectTimeout" default="15000"/></Set>
        </Get>
      </New>
    </Arg>
  </Call>

  <!-- =========================================================== -->
  <!-- Create a TLS specific HttpConfiguration based on the        -->
  <!-- common HttpConfiguration defined in jetty.xml               -->
  <!-- Add a SecureRequestCustomizer to extract certificate and    -->
  <!-- session information                                         -->
  <!-- =========================================================== -->
```

```
      <New id="sslHttpConfig" class="org.eclipse.jetty.server.HttpConfiguration">
     <Arg><Ref refid="httpConfig"/></Arg>
    <Call name="addCustomizer">
     <Arg>
       <New class="org.eclipse.jetty.server.SecureRequestCustomizer">
         <Arg name="sniHostCheck" type="boolean"><Property name="jetty.ssl.sniHostCheck" default="true"/></Arg>
         <Arg name="stsMaxAgeSeconds" type="int"><Property name="jetty.ssl.stsMaxAgeSeconds"
default="-1"/></Arg>
         <Arg name="stsIncludeSubdomains" type="boolean"><Property name="jetty.ssl.stsIncludeSubdomains"
default="false"/></Arg>
       </New>
     </Arg>
    </Call>
  </New>

</Configure>
```

Note : Please update the value below (as above highlighted in red).

1/ ssl port default

2/ ssl timeout default

3/ ssl acceptor priority delta default

4/ ssl accept queue size default

5/ ssl connection timeout default.

jetty-ssl-context.xml

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN" "http://www.eclipse.org/jetty/configure_9_3.dtd">

<!-- ============================================================= -->
<!-- SSL ContextFactory configuration                              -->
<!-- ============================================================= -->

<!--
  To configure Includes / Excludes for Cipher Suites or Protocols see tweak-ssl.xml example at
     https://www.eclipse.org/jetty/documentation/current/configuring-ssl.html#configuring-sslcontextfactory-cipherSuites
-->

<Configure id="sslContextFactory" class="org.eclipse.jetty.util.ssl.SslContextFactory$Server">
  <Set name="Provider"><Property name="jetty.sslContext.provider"/></Set>
  <Set name="KeyStorePath"><Property name="jetty.base" default="." />/<Property name="jetty.sslContext.keyStorePath"
deprecated="jetty.keystore" default="etc/<!--keystore-name-->"/></Set>
  <Set name="KeyStorePassword"><Property name="jetty.sslContext.keyStorePassword"deprecated="jetty.keystore.password"
default="<!--obfuscated password-->"/></Set>
  <Set name="KeyStoreType"><Property name="jetty.sslContext.keyStoreType" default="<!--keystore type, for e.g.PKCS12--
>"/></Set>
  <Set name="KeyStoreProvider"><Property name="jetty.sslContext.keyStoreProvider"/></Set>
  <Set name="KeyManagerPassword"><Property name="jetty.sslContext.keyManagerPassword"
deprecated="jetty.keymanager.password" default="<!--obfuscated password-->"/></Set>
  <Set name="TrustStorePath"><Property name="jetty.base" default="." />/<Property name="jetty.sslContext.trustStorePath"
deprecated="jetty.truststore" default="etc/<!--keystore-name-->"/></Set>
  <Set name="TrustStorePassword"><Property name="jetty.sslContext.trustStorePassword"
deprecated="jetty.truststore.password"/></Set>
  <Set name="TrustStoreType"><Property name="jetty.sslContext.trustStoreType"/></Set>
  <Set name="TrustStoreProvider"><Property name="jetty.sslContext.trustStoreProvider"/></Set>

<Set name="EndpointIdentificationAlgorithm"><Property name="jetty.sslContext.endpointIdentificationAlgorithm"/></Set>
  <Set name="NeedClientAuth"><Property name="jetty.sslContext.needClientAuth"
deprecated="jetty.ssl.needClientAuth" default="false"/></Set>
  <Set name="WantClientAuth"><Property name="jetty.sslContext.wantClientAuth"
deprecated="jetty.ssl.wantClientAuth" default="false"/></Set>
  <Set name="useCipherSuitesOrder"><Property name="jetty.sslContext.useCipherSuitesOrder"
default="true"/></Set>
  <Set name="sslSessionCacheSize"><Property name="jetty.sslContext.sslSessionCacheSize" default="-1"/></Set>
  <Set name="sslSessionTimeout"><Property name="jetty.sslContext.sslSessionTimeout" default="-1"/></Set>
  <Set name="RenegotiationAllowed"><Property name="jetty.sslContext.renegotiationAllowed" default="true"/></Set>
  <Set name="RenegotiationLimit"><Property name="jetty.sslContext.renegotiationLimit" default="5"/></Set>

        <Set name="ExcludeProtocols">
                <Array type="java.lang.String">
                    <!--  Add in the protocols to be excluded -->
                </Array>
        </Set>

  <Set name="ExcludeCipherSuites">
    <Array type="String">
        <!--  Add in the ciphers to be excluded -->
```

```
      </Array>
    </Set>


  </Configure>
```

Note : Please update the keystore name (more information on the next step), keystore type e.g in this case we are using PKCS12 and obfuscated password

For more information about the obfuscated password, please refer to this link (Page 27)

https://docs.elixirtech.com/Ambience/4.0.0/Install/fo-xep/Elixir%20Ambience%20Installation%20Guide.pdf

Create a Private Key / Certificate Signing Request / Self-Signed Certificate and Keystore

Creating a Private Key

First, we'll create a private key. A private key helps to enable encryption and is the most important component of our certificate.

```
openssl genrsa -des3 -out server.key 2048
```

Creating a Certificate Signing Request

If we want our certificate signed, we need a certificate signing request (CSR)**.** The CSR includes the public key and some additional information (such as organization and country).

Create a CSR (*server.csr*) from our existing private key:

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

Creating a Self-Signed Certificate

A self-signed certificate is a certificate that's signed with its own private key. It can be used to encrypt data just as well as CA-signed certificates, but our users will be shown a warning that says the certificate isn't trusted.

Create a self-signed certificate (server.crt) with our existing private key and CSR:

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

note: The -days option specifies the number of days that the certificate will be valid.

Create a PKCS12 keystore from a private key and certificate.

The private key and certificate must be in Privacy Enhanced Mail (PEM) format. Use the following OpenSSL commands to create a PKCS#12 file from your private key and certificate

1. Open terminal.
2. Execute the following commands.
   note : Make sure that the following commands are executed at the same directory in where “server.key” and “server.crt” resides.

- Create a PEM file.

```
cat server.key > server.pem
cat server.crt >> server.pem
```

- Create .pkcs12 file

```
openssl pkcs12 -export -in server.pem -out keystore.pkcs12
```

Once keystore generated please copy over to **jetty/elx-base/etc** path of your Ambience installation.

Edit the start.ini file found in your “jetty\elx-base” directory and add in below following lines:

- Under the following comment, “# Module: http”:

```
# ---------------------------------------
# Module: http
--module=ssl
--module=https
```

Start up your Ambience server. Open browser and verified the HTTPS port protocol.
For example, **https://localhost:8443**