Configuring Apache Tomcat to use secure https protocol

Summary: this page explains how to modify your Apache Tomcat application server to enable the use of the **https** protocol.

Introduction

When installed "out of the box", the Apache Tomcat application server will be configured to use the "**http**" protocol on **port 8080**, as shown in the following extract from the **CATALINA_HOME\conf\server.xml** file:

```
<!-- A "Connector" represents an endpoint by which requests are received
     and responses are returned. Documentation at :
     Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
     Java AJP  Connector: /docs/config/ajp.html
     APR (HTTP/AJP) Connector: /docs/apr.html
     Define a non-SSL HTTP/1.1 Connector on port 8080
-->

<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />

<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
           port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
-->
```

If your organization requires the use of the "**https**" protocol on **port 443** (or another port) when interacting with the CAST dashboards, then there are various steps that need to be completed with regard to the Apache Tomcat installation. You can find out more information about the changes that are required by following the official Apache Tomcat documentation here: **https://tomcat.apache.org/tomcat-8.5-doc/ssl-howto.html**.

SSL certificates

CAST highly recommends the use of a trusted **CA (Certificate Authority) SSL certificate** when configuring Apache Tomcat for secure https access. You can use a **self-signed SSL certificate**, however, it is not recommended since unpredictable results may occur when using CAST dashboards with this type of certificate.

Advanced security configuration options

If you have configured Apache Tomcat for **secure https access**, CAST highly recommends that you ALSO configure the following options to further secure your installation.

"secure" attribute

If you intend to serve both http and https from your Apache Tomcat application server, the "secure" attribute" should NOT be added.

CAST recommends that you add the "**secure**" attribute to your SSL connector and set it to "**true**" - this attribute forces Apache Tomcat to specify whether the request was made using a secure channel, such as **https**. To do so:

- Edit the **CATALINA_HOME\conf\server.xml** file with a text editor
- Find your existing SSL connector in the file
- For a **JSSE SSL** implementation, the connector will look something like this:

```
<Connector port="8443"
    SSLEnabled="true"
    scheme="https"
    clientAuth="false"
    SSLProtocol="TSL"
    protocol="HTTP/1.1"
    keystoreFile="path/to/keystore"
    keystorePass="passwordOfKeystore"/>
```

- For an **APR SSL** implementation, the connector will look something like this:

```
<Connector port="8443"
        SSLEnabled="true"
        scheme="https"
        clientAuth="false"
        SSLProtocol="SSLv3+TLSv1"
        protocol="HTTP/1.1"
        SSLCertificateFile="path/to/server.crt"
        SSLCertificateKeyFile="path/to/server.pem"/>
```

- You now need to add the **secure="true"** attribute to your SSL connector as follows:
- For a **JSSE SSL** implementation

```
<Connector port="8443"
        secure="true"
    SSLEnabled="true"
    scheme="https"
    clientAuth="false"
    SSLProtocol="TSL"
    protocol="HTTP/1.1"
    keystoreFile="path/to/keystore"
    keystorePass="passwordOfKeystore"/>
```

- For an **APR SSL** implementation:

```
<Connector port="8443"
        secure="true"
    SSLEnabled="true"
    scheme="https"
    clientAuth="false"
    SSLProtocol="SSLv3+TLSv1"
    protocol="HTTP/1.1"
    SSLCertificateFile="path/to/server.crt"
    SSLCertificateKeyFile="path/to/server.pem"/>
```

Following any changes you make, **save the CATALINA_HOME\conf\server.xml** file and then **restart** your application server so that the changes are taken into account.

You can find out more information about the "secure" attribute here: https://tomcat.apache.org/tomcat-8.0-doc/config/http.html

"useHttpOnly" attribute

CAST recommends that you add the "**useHttpOnly**" attribute to your context and set it to "**true**" - this attribute forces an HttpOnly flag be set on session cookies to prevent **client side script** from accessing the session ID. To do so:

- Edit the **CATALINA_HOME\conf\context.xml** file with a text editor
- You will find the **<context>** element as shown below:

```
<Context>
    <!-- Default set of monitored resources -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->
    <!-- Uncomment this to enable Comet connection tacking (provides events
         on session expiration as well as webapp lifecycle) -->
    <!--
    <Valve className="org.apache.catalina.valves.CometConnectionManagerValve" />
    -->
</Context>
```

- Add the **useHttpOnly**" attribute to the opening **<context>** tag and set it to "**true**":

```
<Context useHttpOnly="true">
    <!-- Default set of monitored resources -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->
    <!-- Uncomment this to enable Comet connection tacking (provides events
         on session expiration as well as webapp lifecycle) -->
    <!--
    <Valve className="org.apache.catalina.valves.CometConnectionManagerValve" />
    -->
</Context>
```

Following any changes you make, **save the CATALINA_HOME\conf\context.xml** file and then **restart** your application server so that the changes are taken into account.

Note:

- You can find out more information about the "useHttpOnly" attribute here: https://tomcat.apache.org/tomcat-8.0-doc/config/context.html.
- Adding the "useHttpOnly" attribute will activate it for ALL web applications running in Apache Tomcat.