**Single Sign On**

**Central Authentication Service**

**Client Configuration Guide**

**Introduction**

This document will help in CASifying a Web Application for Single Sign On with a CAS Server.

**Prerequisites**

|  |  |  |
| --- | --- | --- |
| Web Application Parameters | POC | Production |
| Deployed on HTTPS | Yes | Yes |
| Digital Certificate Generation | Self-Signed | Certificate Authority |
| Import CAS Server Certificate at Client Side Key Store | Yes | Yes |
| Import Client Certificate at CAS Server Side Key Store | No | Yes |

**Steps to CASify a Web Application (POC)**

1. **Generating Self-Signed Certificate**

This step is required only if the Web Application is not deployed on Https currently.

If the Web Application is already deployed on Https please go to **Step 2.**

To generate a Self-Signed Certificate, a Certificate Key Store is need to be created and Certificate is generated from this Key Store.

Below is an example shown for generation of Self-Signed Certificate for your Web Application deployed in a Java based environment.

* Create Java Key Store

The following command will create a Java Key Store using keytool command on windows command line.

**Command**:

keytool -genkey -alias <CERTIFICATE\_ALIAS> -keyalg RSA -validity 10000 -keystore <JAVA\_KEYSTORE\_PATH> -ext san=ip:<IP\_ADDRESS> -storepass <JAVA\_KEYSTORE\_PASS> -keypass <CERTIFICATE\_PASS> -dname <DNAME>

**Example:**

keytool -genkey -alias CAS1 -keyalg RSA -validity 10000 -keystore “D:\cas\_client\jdk7\jre\lib\security\cacerts” -ext san=ip:172.123.123.1 -storepass portalcas1 -keypass portalcas1 -dname "CN=ABC, OU=ABC, O=ABC, L=EDB, ST=EDB, C=SL"

* Create Web Application Key Store

The following command will create a Web Application Key Store using keytool command on windows command line.

**Command**:

keytool -genkey -alias <CERTIFICATE\_ALIAS> -keyalg RSA -validity 10000 -keystore <WEBAPP\_KEYSTORE\_PATH> -ext san=ip:<IP\_ADDRESS> -storepass <WEBAPP\_KEYSTORE\_PASS> -keypass <CERTIFICATE\_PASS> -dname <DNAME>

**Example:**

keytool -genkey -alias CAS1 -keyalg RSA -validity 10000 -keystore “D:\cas\_client\ssl\CAS1.keystore” -ext san=ip:172.123.123.1 -storepass portalcas1 -keypass portalcas1 -dname "CN=ABC, OU=ABC, O=ABC, L=EDB, ST=EDB, C=SL"

* Generate Self-Signed Certificate from Web Application Key Store

The following command will generate a new Self-Signed Certificate for the Web Application from the Web Application Key Store using keytool command on windows command line.

**Command**:

keytool -exportcert -alias <CERTIFICATE\_ALIAS> -keystore <WEBAPP\_KEYSTORE\_PATH> -file <CERTIFICATE\_PATH> -storepass <WEBAPP\_KEYSTORE\_PASS>

**Example:**

keytool -exportcert -alias CAS1 -keystore “D:\cas\_client\ssl\CAS1.keystore”-file “D:\cas\_client\ssl\CAS1.cer” -storepass portalcas1

1. **Import CAS Server Certificate in Key Store**

This step is mandatory.

You will be provided with a Digital Server Certificate used by CAS Server. This certificate needs to be imported in the trust chain of your Certificate Key Store for successful SSL handshake.

Below is an example shown for importing the Server Certificate if your Web Application deployed in a Java based environment.

The following command will import the CAS Server Certificate into Java Key Store using keytool command on windows command line.

**Command**:

keytool -importcert -trustcacerts -keystore <JAVA\_KEYSTORE\_PATH>-file <SERVER\_CER\_PATH> -alias <SERVER\_CER\_ALIAS> -storepass <JAVA\_KEYSTORE\_PASS>

**Example:**

keytool -importcert -trustcacerts -keystore “D:\cas\_client\jdk7\jre\lib\security\cacerts”-file “D:\cas\_client\ssl\CASSERVER.cer” -alias CASSERVER -storepass portalcas1

1. **Set up the Web Application on HTTPS**

This step is required only if your Web Application is not deployed on HTTPS.

If you have already deployed your Web Application on HTTPS then make sure the Web Application Server points to the <WEBAPP\_KEYSTORE\_PATH> and refers to the Certificate with alias <CERTIFICATE\_ALIAS>.

Also make a note of the port on which your Web Application Server is running on HTTPS.

If you are yet to deploy your Web Application on HTTPS please follow the below steps.

Once your Web Application Key Store and Java Key Store are created we need to configure the Web Application server on HTTPS by doing the configurations in the applications Server Configuration.

Below is an example shown for configuring the Apache Tomcat Server assuming the Web Application is deployed on it.

Locate the Define a SSL HTTP/1.1 Connector on port in server.xml file of the Tomcat Server and uncomment the block if commented.

Make below mentioned changes in this section:

**Changes:**

<Connector port="<HTTPS\_PORT\_NO>" protocol="HTTP/1.1" keystorePass="<WEBAPP\_KEYSTORE\_PASS>" sslProtocol="TLS" secure="true" scheme="https" maxThreads="150" keystoreFile="<WEBAPP\_KEYSTORE\_PATH>" clientAuth="false" alias="<CERTIFICATE\_PATH>" SSLEnabled="true"/>

**Example:**

<Connector port="8443" protocol="HTTP/1.1" keystorePass="portalcas1" sslProtocol="TLS" secure="true" scheme="https" maxThreads="150" keystoreFile=" D:\cas\_client\ssl\CAS1.keystore" clientAuth="false" alias="CAS1" SSLEnabled="true"/>

1. **Download the Client Libraries**

Different CAS Client libraries are available for specific type of Web Applications.

These libraries can be downloaded from [here.](http://www.jasig.org/cas/client-integration)

The Client libraries specifically for Java based Web Application are as follows:

cas-client-core-3.1.3.jar

cas-client-integration-atlassian-3.1.3.jar

commons-logging-1.1.jar

servlet-api-2.3.jar

xmlsec-1.3.0.jar

These libraries can be found [here.](http://downloads.jasig.org/cas-clients/)

1. **Configuration in Web Application for Single Sign On with CAS Server (Redirection/Call Back Approach)**

CAS Client Web Application configuration for Single Sign On with CAS Server differs from framework to framework. The detailed description about these configurations can be found [here.](http://www.jasig.org/cas/client-integration)

The sample configurations required to be done in a Java Based Web Application are as follows:

**web.xml**

****

As per the sample configurations mentioned in the web.xml file:

Whenever any page on the Client Web Application which is kept under /samlAttrServlet is requested by a user, the page will first be redirected to the CAS Server Login Page if the user is not logged in.

The user when Logs in on this CAS Server Login page, will be redirect back to the requested page.

**Example**

If a below page is requested by a User:

<https://CAS_CLIENT_HOST:HTTPS_PORT_NO/samlAttrServlet/>abc.jsp

First the user if not logged in is redirected to the below page:

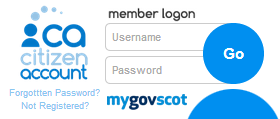
<https://CAS_SERVER_HOST:HTTPS_PORT_NO/login>

Once the user logs in on that page, page will be redirected back to:

<https://CAS_CLIENT_HOST:HTTPS_PORT_NO/samlAttrServlet/>abc.jsp

1. **Configuration in Web Application to use Login Plugin**

Authentication with the CAS Server can also be done via including a plugin component in a Client Web Application page.



Instead of redirection to the CAS Server Login page, this component can be plugged in to the Client Web Application homepage in an iFrame.

The plugin component consists of some HTML code and JavaScript code which is compatible with any web page. The codes for the same are attached here.

The javascripts attached are required to be included in any Web Page of the Web Application.



The CAS\_SERVER\_HOST:PORT\_NO needs to be replaced in **message.js** with the actual url.

Also add following code in the Web Page of the Web Application.

<script type="text/javascript" src="js/porthole.js"></script>

<script type="text/javascript" src="js/json2.js"></script>

<script type="text/javascript" src="js/message.js"></script>

And

<iframe id="loginFrame" name="loginFrame" width="280px" height="120px" frameborder="0" src="https://<CAS\_SERVER\_HOST:PORT\_NO>/login?service=<YOUR\_URL\_WHERE\_LOGGED\_IN\_USER\_WILL\_BE\_REDIRECTED>&ViewType=pluginView" style="overflow: hidden;padding:0; "></iframe>

The code contains:

<YOUR\_URL\_WHERE\_LOGGED\_IN\_USER\_WILL\_BE\_REDIRECTED>

This string needs to be replaced by the URL containing where you want to redirect the user on your Web Application Page after logging in.

**Example**

The string will be replaced by:

http://www.councilwebsite.com/myloggedinaccount

1. **Configuration in Web Application to Retrieve Custom Attributes**

The CAS Server can pass some of the configured custom attributes of the logged in user to the CAS Client upon successful authentication while redirecting back to the requested Client Web Application Page.

It is important to note that these attributes can be retrieved at Client Web Application only if they are configured in CAS Server.

Below is the sample code to retrieve these custom attributes as Key Value pair in Java Web Application:

String username= request.getRemoteUser();

AttributePrincipal principal = (AttributePrincipal) request.getUserPrincipal();

Map attributes = principal.getAttributes();