Oct. 12, 2020

Java PKI Certificate Based Authentication: Coding Examples

Sample Java Code for making a PJM browserless call with PKI Certificates:

PJMBrowserless.java

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.Security;

import javax.net.ssl.SSLContext;

import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.config.Registry;
import org.apache.http.config.RegistryBuilder;
import org.apache.http.conn.socket.ConnectionSocketFactory;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.conn.ssl.TrustSelfSignedStrategy;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.impl.conn.BasicHttpClientConnectionManager;
import org.apache.http.ssl.SSLContexts;
import org.apache.http.util.EntityUtils;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
```

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Sample code for certificate based authentication
 * <p>
 * - login to SSO with PKI certificate to retrive token
 * <p>
 * - Make a call to application with token
 * <p>
 * - Logout from SSO with token
 * <p>
 * This code is build using Apache-httpclient4 , bouncycastle, json-google-tool-kit
 */
public class PJMBrowserless {
   private static final Logger log = LoggerFactory.getLogger(PJMBrowserless.class);

   /**
    * main method
    */
   public static void main(String[] args) throws Exception {

      PJMBrowserless browserless = new PJMBrowserless();

      String tokenId = browserless.loginWithCertificate();
      log.info("created sso token {}", tokenId);

      String appResponse = browserless.getRestCall(tokenId);
      log.info("Application response {}", appResponse);

      browserless.logout(tokenId);
```

```
      log.info("Sucessfully logged out from sso");
   }

   /**
    * Method makes a call to SSO with credentials and certificate to retrieve
    * SSO token
    *
    * @return sso tokenid
    * @throws Exception
    */
   private String loginWithCertificate() throws Exception {

      String pjmUserName = "*****"; //replace with username
      String pjmPassword = "*****"; //replace with password
      String pfxlocation = "C:\\certpath\\certificate.pfx"; //replace with certificate location
      String certPassword = "****"; //replace with certificate password
      String tokenId = null;
      String ssourl = "https://ssotrain.pjm.com/access/authenticate/pjmauthcert"; //
sso.pjm.com for Prod
      CloseableHttpClient httpClient = null;
      InputStream inputStream = null;

      try {

         File f = new File(pfxlocation);

         // below code establishes 2-way SSL connection on sso.pjm.com with client certificate

         if (Security.getProvider("BC") == null) {
            Security.addProvider(new BouncyCastleProvider());
         }

         KeyStore keystore = KeyStore.getInstance("PKCS12", "BC");
         inputStream = new FileInputStream(f);
```

```
        keystore.load(inputStream, certPassword.toCharArray());

        SSLContext sslContext = SSLContexts.custom().loadKeyMaterial(keystore,
certPassword.toCharArray())
                .loadTrustMaterial(null, new TrustSelfSignedStrategy()).build();
        SSLConnectionSocketFactory sslConnectionFactory = new
SSLConnectionSocketFactory(sslContext);

        Registry<ConnectionSocketFactory> registry = RegistryBuilder.<ConnectionSocketFactory>
create()
                .register("https", sslConnectionFactory).build();

        BasicHttpClientConnectionManager connManager = new
BasicHttpClientConnectionManager(registry);

        httpClient =
HttpClients.custom().setConnectionManager(connManager).setSSLSocketFactory(sslConnectionFactory
)
                .build();

        HttpPost httpPost = new HttpPost(ssourl);
        httpPost.addHeader("X-OpenAM-Username", pjmUserName);
        httpPost.addHeader("X-OpenAM-Password", pjmPassword);
        HttpResponse httpResponse = httpClient.execute(httpPost);

        int statuscode = httpResponse.getStatusLine().getStatusCode();
        String json = EntityUtils.toString(httpResponse.getEntity());

        if (statuscode != 200) {
           log.error("error on sso login, status is {} and response is {} ", statuscode,
json);
           throw new RuntimeException("error on sso login status code: " + statuscode);
        }

        JSONParser parser = new JSONParser();
        Object resultObject = parser.parse(json);
```

```
            JSONObject obj = (JSONObject) resultObject;
            tokenId = (String) obj.get("tokenId");

        } catch (ClientProtocolException e) {
            log.error("unable to create sso token", e);
            throw new RuntimeException("unable to create sso token ", e);
        } catch (IOException e) {
            log.error("unable to create sso token", e);
            throw new RuntimeException("unable to create sso token ", e);
        } catch (ParseException e) {
            log.error("unable to create sso token", e);
            throw new RuntimeException("unable to create sso token ", e);
        } finally {
            if (httpClient != null)
                httpClient.close();
            if (inputStream != null)
                inputStream.close();
        }
        return tokenId;

    }

    /**
     * Call to Application REST API with token (*No changes with PKI)
     *
     * @param sso tokenId
     * @return Application response in String
     */
    private String getRestCall(String tokenId) {

        String resturl =
"https://exscheduletrain.pjm.com/exschedule/rest/secure/download/xml/schedules";
        String appResponse = "";
```

```
      try (CloseableHttpClient httpClient =
HttpClientBuilder.create().disableRedirectHandling().build()) {
         HttpGet httpget = new HttpGet(resturl);
         httpget.addHeader("Cookie", "pjmauthtrain=" + tokenId); // pjmauth for production

         HttpResponse httpResponse = httpClient.execute(httpget);
         int statuscode = httpResponse.getStatusLine().getStatusCode();
         appResponse = EntityUtils.toString(httpResponse.getEntity());

         if (statuscode != 200) {
            log.error("error on applicaton rest call, status is {} and response is {} ",
statuscode, appResponse);
            throw new RuntimeException("error on applicaton rest call, status is " +
statuscode);
         }

      } catch (IOException e) {
         throw new RuntimeException("unable to retrieve data", e);
      }
      return appResponse;

   }

   /**
    * Logout from SSO (*No changes with PKI)
    *
    * @param tokenId sso tokenId
    */
   private void logout(String tokenId) {

      String ssoLogoutUrl = "https://ssotrain.pjm.com/access/logout"; // sso.pjm.com for
production

      try (CloseableHttpClient client =
HttpClientBuilder.create().disableRedirectHandling().build()) {
```

```
        HttpPost httpPost = new HttpPost(ssoLogoutUrl);
        httpPost.addHeader("Cookie", "pjmauthtrain=" + tokenId);

        HttpResponse httpResponse = client.execute(httpPost);

        String appResponse = EntityUtils.toString(httpResponse.getEntity());

        int statuscode = httpResponse.getStatusLine().getStatusCode();

        if (statuscode != 200) {
           log.error("error on sso logout, status is {} and response is {} ", statuscode,
appResponse);
           throw new RuntimeException("error on sso logout, status is " + statuscode);
        }

     } catch (ClientProtocolException e) {
        throw new RuntimeException("error on sso logout ", e);
     } catch (IOException e) {
        throw new RuntimeException("error on sso logout ", e);
     }
  }
}
```