BI 4.2 Custom Elements installation and configuration

Note that I am running SAP BI 4.2 Service Pack 5, on a Windows Server 2016 Datacentre

Preparation

Leaflet: https://leafletjs.com/download.html

You will need to download and unpack the Leaflet application. You will require the following files. I have listed those that I have placed in our environment.
leaflet.css, leaflet.js, leaflet.markercluster.js, leaflet.markercluster-src.js, leaflet-heat.js, leaflet-src.js
MarkerCluster.css, MarkerCluster.Default.css and screen.css

You can put the leaflet application inside the BI 4.2 Tomcat\webapps\ folder. For example, our leaflet is placed in C:\<BOE Install>\tomcat\webapps\leaflet

Images:
Create yourself an image folder on the BI server to house map icons. Ideally you want to use PNG and the background of the image is transparent.

I have placed ours in C:\<BOE Install>\tomcat\webapps\leaflet\map_icons\

HTTPS or HTTP
If your BI 4.2 server sits under HTTPS then understand that environment and hopefully you will know all about HTTPS configuration, for me it was a pain and took many attempts to get it to work, and I hope the instructions I have listed below is sufficient to assist you and work, should you struggle like I did. I am not an expert in this field either so I may not be able to answer any high-level technical questions. If you are running BI 4.2 server as HTTPS also configure the NodeJS file to run as HTTPS.

Allowing custom JS files to run

I think you may also need to amend one of BI properties files, I have made so many configuration changes to get custom elements working and other projects, I never kept track of everything I did.

Backup file global.properties located here C:\<BOE Install>\tomcat\webapps\BOE\WEB-INF\internal\

Amend the original;

Locate the following line

 # Enable or disable BOE Trust Guard for CSRF protection
boe.trustguard.enable=false

Amend to true

 # Enable or disable BOE Trust Guard for CSRF protection
boe.trustguard.enable=true

It might also be this line but currently ours is set to false and custom elements still works.

 # Use precompiled JSPs?
precompiled.jsp.files.use=false

Setting up NodeJS custom element on a server

DO NOT install onto the same server running BI 4.2, just in case.

Download version “node-v10.16.0-x64” from https://nodejs.org/en/ and then install onto computer 1.
Create a folder to house your custom element area workspace. Such as C:\MyCustomElements.
Open Windows CMD prompt as an Administrator and navigate to your workspace folder.

Enter the following commands and press enter.

npm install pm2 -g
npm install

If you struggle due to Administrative restrictions placed on the server (computer 1) you are installing NodeJS onto then install onto a machine you have full rights to (computer 2) and prepare to copy the following folder structures and paste into an identical location onto computer 1 that will eventually be running your custom element.

From computer 2
1. Copy the NODEJS installation folder to a flash drive or network location that both computers have access to.
2. Copy the npm and npm-cache from the following location: C:\Users\<username>\AppData\Roaming\

Rename the NodeJS installation folder on computer 1 and replace with item 1 above.
Copy item 2 above into an identical location on computer 1.

With NodeJS, NPM and PM2 installed we can place our custom element JS file in our workspace and edit it in preparation for running.

The CE_Map.js file

Credit goes to https://wiki.scn.sap.com/wiki/display/BOBJ/Custom+Elements+Sample+-+SAP+Web+Intelligence+4.2

Many failed attempts using the example given on the above website, and I have taken there JS file and amended it using what little experience I have in programming Javascript and HTML to get maps working, some of the code within the I am assuming may not be required and I have left it in, but I will leave that to those more qualified than I am to confirm my assumption.

Configure your Leaflet and BI 4.2 server URL's

```
var leafletHomeURL = "https://<BI 4.2 server>/ leaflet/";
var leafletMapURL = "https://{s}.tile.osm.org/{z}/{x}/{y}.png"; // Open Street Maps
var leafletImagesURL = "https:// <BI 4.2 server>/ leaflet/map_icons/";
```

Test the URL's before committing the changes, just in case, took me a while before I had the correct URL syntax.

You may want to Increase amount of available RAM within the JS file of your custom element. 30Mb is more than enough for 50,000 data points.

```
app.use(bodyParser.json({limit: '50mb', extended: true}));
app.use(bodyParser.urlencoded({limit: '50mb', extended: true}));
```

Set the port that the custom element will be using (NOT HTTPS), in the example CE_Maps.js file I have provided I have remarked this area out, if you are using HTTP then unremark

```
app.listen(<port>);
app.get("/", function(req, res) {
res.send("Server Up and Running");
});
```

If you are placing custom element securely in HTTPS you will need a certificate, I placed mine in the same location as the javascript JS file.

```
var options = {
 key: fs.readFileSync('E:\\certs\\key-20190821-085009.pem'),
 cert: fs.readFileSync('E:\\certs\\cert-20190821-085009.crt'),
 hostname : 'vmnodejs',  // replace with your server hostname housing your nodeJS
 path : "/",
 method : "GET",
};

// Create an HTTP service
http.createServer(app).listen(8080);

// Create an HTTPS service identical to the HTTP service.
https.createServer(options, app).listen(13000);
```

To run your custom element, you will need to open a Windows CMD prompt as Administrator on the NodeJS server and navigate to the folder housing your JS file and run it using the PM2 commands, location example in previous step, C:\MyCustomElements

PM2 commands, ***do not run unless you have already amended the CE_Maps.js file for your environment.***

pm2 start "CE_Maps.js" – Run the customer element
pm2 list – List all nodejs processes running, each process will have an ID, starting with 0
pm2 stop 0 or ***pm2 stop all*** – stop process with id of 0, ***all*** will stop all running processes.
pm2 delete 0 or ***pm2 delete al***l – Delete process with id 0 or delete all. It only deletes the running process not the JS file.

Check server is running.
http://<hostname>:<port> or use https

Check list of visualisation inside the custom element
http://<hostname>:<port>/api/visualizations

Should you get any errors with the JS file please refer to NodeJS online support, I have checked and rechecked the JS file several times and it does not fail, you may not have one or more of the "requires" as seen in the initialisation area of the CE_Maps.js file. Therefore you will need to install it on the NodeJS server using npm install syntax.

BI 4.2 Server configuration to enable HTTPS.

A certificate will need to be created, if your IT department is unable to create you a certificate you can use CreateCertGUI.exe to generate one, this can be run on any machine. Follow instructions below to run CreateCertGUI.exe. Search Google for the CreateCertGUI.exe, I found out about this file by accident via this YouTube tutorial https://www.youtube.com/watch?v=wC2_vG6IbkM

Place the certificate into the Tomcat\conf folder, it does not need to be placed here I have placed here for convenience.

Stop TOMCAT and delete the CATALINA folder cache.
Navigate to the Tomcat\conf\ folder and backup both server.xml and web.xml files.
Edit web.xml and add the following to the end of the file.
Make sure it is placed before the closing of </web-app>.

```
  <security-constraint>
          <web-resource-collection>
             <web-resource-name>HTTPSOnly</web-resource-name>
            <url-pattern>/*</url-pattern>
           </web-resource-collection>
         <user-data-constraint>
          <transport-guarantee>CONFIDENTIAL</transport-guarantee>
         </user-data-constraint>
    </security-constraint>
</web-app>
```

Edit the server.xml file and add the following connector, replacing the original connector or remark out the original connector.

```
<Connector port="80" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="443"
          compression="on"
          URIEncoding="UTF-8"
         compressionMinSize="2048" noCompressionUserAgents="gozilla, traviata"
compressableMimeType="text/html,text/xml,text/plain,text/css,text/javascript,text/json,application/javascript,application/json"/>


<Connector port="443"          protocol="HTTP/1.1" SSLEnabled="true"  maxThreads="150"
       scheme="https" secure="true" clientAuth="false" sslProtocol="TLS"
       keystorePass="<password>"
       keystoreFile="C:\SAP BusinessObjects\tomcat\conf\cert-key.p12" />
```

Remember to adjust your Windows firewall settings to allow the ports through, if in doubt turn Windows firewall off for testing purposes.

Start Tomcat and SIA.

IMPORTANT: Remember to amend the OpenDocument properties URL within BI 4.2 and change from HTTP to HTTPS, this is also within Applications.

Also IMPORTANT, if you are going to create a production environment under HTTPS make sure your IT department provide you with the certificates you need.

Running CreateCertGUI.exe

Run the file as Administrator and complete the following.

Years = Years till certificate expires
County = Enter EN
Organisation = Anything you like
Common Name = Must be full hostname especially if you are on a domain.
P12 Password = Do not forget this it will be used in the server.xml configuration file later on.

This will generate 3 files for you;
cert-xxxxxx-crt
cert-key-xxxxx.p12
key-xxxxxxx.pem

It is the p12 file we will require for the BI 4.2 server. Copy this file to the BI 4.2 Tomcat folder.

Adding the custom element to BI 4.2

Log on to BI 4.2 CMC, click on Applications, then select Web Intelligence and then Custom Element. Add a new custom element which is the URL to the NodeJS custom element JS file.

EG : http://<hostname>:port>

When entering a name for the custom element use as few characters as possible so they appear within the BI 4.2 report element selection, so do not enter “This is my custom element” as you will only see “This is my.” within the window panel and the customer would have to scroll the window to see the full name element name.

When adding the custom element to Web Intelligence, if you get the following error then follow the steps to add the custom element and get NodeJS URL working inside BI 4.2. You may not get an error if the NodeJS URL is HTTP.

Test has failed. (because NodeJS is under HTTPS and needs to be trusted within BI)
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target

---

Getting NodeJS URL working inside BI 4.2 HTTPS

This step is not required if the BI 4.2 server is running under HTTP rather than HTTPS.

You will need to install InstallCert java file to generate files and we will also be altering the BI 4.2 cacerts file to trust our custom element URL. The install can be on any machine that has Java installed, best to place InstallCert on the BI 4.2 server so we can use the SAPJVM version of Java.

The following instructions can be found on https://github.com/escline/InstallCert

On your BI 4.2 server we need to make sure JAVA_HOME is set to point to the 64bit version of SAPJVM. On my test machine this location is C:\<BOE Install>\SAP BusinessObjects Enterprise XI 4.0\win64_x64\sapjvm\

To check open Windows CMD as Administrator and type in "java -version" without quotes.

> *java version "1.8.0_131"*
> *Java(TM) SE Runtime Environment (build 8.1.030)*
> *SAP Java Server VM (build 8.1.030 25.51-b10, Apr 26 2017 16:21:57 - 81_REL - optU - windows x86 - 6 - bas2:288514 (mixed mode))*

If all ok then navigate to the folder location where InstallCert resides.

We need now to compile, enter javac InstallCert.java and press enter.

Next we create a certificate, enter java InstallCert [host]:[port], where host is the full hostname and port running the NodeJS custom element.

It will create a jssecacerts file where at the end you are asked to "Enter certificate to add to trusted keystore", press enter.

Next navigate to SAPJVM 64bit folder and paste in the jssecacerts file to this folder;
C:\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\win64_x64\sapjvm\jre\lib\security

Type in the following command;
keytool -exportcert -alias <hostname>-1 -keystore jssecacerts -storepass changeit -file <mycert>.cer

Enter the full hostname, you do not need to enter a port, and enter a certificate filename.

Next type in the following command;
keytool -importcert -alias <hostname> -keystore cacerts -storepass changeit -file <mycert>.cer

You will be asked to trust the certificate, enter "yes" without quotes and press enter. Stop all BI 4.2 services, delete the Tomcat Catalina folder cache and restart the services.

Make sure all the CMC services are running before adding the custom element.