HITACHI
Inspire the Next

Security Document

Lumada Manufacturing Insights V 2.5.0

November 2020

Contents

Chapter 1: Setting up basic auth users & permissions in Druid

Configuring Druid for basic-auth:

This document covers enabling basic authentication for Druid. This does not cover enabling TLS for Druid. The purpose of this guide is to set up basic user authentication, users, roles and permissions. Initial Admin credential can be used to access API and Druid console.

Following are the steps:

1. Edit **opt/druid-0.12.1/conf-quickstart/ druid/_common/ common.runtime.properties** and add druid-basic-security extension to **druid.extensions.loadList** property:

   druid.extensions.loadList=["druid-basic-security","druid-kafka-indexing-service"]

2. Add following properties in **common.runtime.properties** file to set up the Basic Authenticator, Authorizer, and Escalator by replacing **[ADMIN_USERNAME]** and **[ADMIN_USER_PASSWORD]** with the desired one:

   # Druid basic security

   druid.auth.authenticatorChain=["mfiMetadataAuthenticator"]

   druid.auth.authenticator.anonymous.type=anonymous

   druid.auth.authenticator.anonymous.identity=defaultUser

   druid.auth.authenticator.anonymous.authorizerName=myBasicAuthorizer

   druid.auth.authenticator.mfiMetadataAuthenticator.type=basic

   druid.auth.authenticator.mfiMetadataAuthenticator.initialAdminPassword=**[ADMIN_USER_PASSWORD]**

   druid.auth.authenticator.mfiMetadataAuthenticator.initialInternalClientPassword=**[ADMIN_USER_PASSWORD]**

   druid.auth.authenticator.mfiMetadataAuthenticator.credentialsValidator.type=metadata

   druid.auth.authenticator.mfiMetadataAuthenticator.skipOnFailure=false

   druid.auth.authenticator.mfiMetadataAuthenticator.authorizerName=mfiMetadataAuthorizer

   # Escalator

   druid.escalator.type=basic

   druid.escalator.internalClientUsername=druid_system

   druid.escalator.internalClientPassword=**[ADMIN_USER_PASSWORD]**

   druid.escalator.authorizerName=mfiMetadataAuthorizer

   # Authorizers Configuration

   druid.auth.authorizers=["mfiMetadataAuthorizer"]

   druid.auth.authorizer.mfiMetadataAuthorizer.type=basic

   druid.auth.authorizer.mfiMetadataAuthorizer.initialAdminUser=**[ADMIN_USERNAME]**

   druid.auth.authorizer.mfiMetadataAuthorizer.initialAdminRole=admin

   druid.auth.authorizer.mfiMetadataAuthorizer.roleProvider.type=metadata

3. Edit following properties of /opt/mi-data-app/config/application.properties file:
   a. druid.enable.auth=true
   b. druid.auth.basic.token=Basic **[AUTHORIZATION_TOKEN]**
      i. **[AUTHORIZATION_TOKEN]** is Base64 encoded value of **[ADMIN_USERNAME]:[ADMIN_USER_PASSWORD]** used in Step #2
4. Restart **mi-data-app**.

5. Update **/opt/startProcess.sh** by adding **Authorization** header in CURL command used for 2 Druid Indexer jobs.

Chapter 2: Generate SSL Certificates

This document describes how to generate SSL certificates for mosquitto broker and clients. MI edge is having two clients(Kepware and nifi) as of now.

Before beginning, we must install “openssl” in “mosquitto broker” and “nifi” instance.

Step 1: Install Development Tools

Install few dependencies such as “Development Tools” under RHEL/CentOS/Fedora or “build-essential” in Debian/Ubuntu as shown.

For Centos:

- sudo yum group install 'Development Tools' && yum install perl-core libtemplate-perl zlib-devel

For Ubuntu:

- sudo apt update && apt install build-essential checkinstall zlib1g-dev libtemplate-perl

Step 2: Compile OpenSSL from Sources

1. Download openssl source code and extract using tar command:
   - wget -c https://www.openssl.org/source/openssl-1.0.2p.tar.gz
   - tar -xzvf openssl-1.0.2p.tar.gz
2. Now, move into the extracted directory, configure, build, after a successful build, test the libraries and install OpenSSL in the default location, which is /usr/local/ssl, by running the following commands:
   - cd openssl-1.0.2p/
   - /config
   - make
   - make test
   - sudo make install
3. Once you have successfully installed OpenSSL, you can move into the installation directory and view the various sub-directories and files using ls command:
   - cd /usr/local/ssl/
   - ls -l

Step 3: To check the version of OpenSSL you have just installed, run the following command:

- /usr/local/ssl/bin/openssl version

Step 4: To use the newly installed OpenSSL version on your system, you need to add the directory /usr/local/ssl/bin/ to your PATH, in the file ~/.bashrc (or the equivalent for your shell).

- vi ~/.bashrc

Step 5: Add this line at the bottom of the file.

- export PATH="/usr/local/ssl/bin:${PATH}"

Step 6: Save and close the file using Esc and :wq@ and reload the configuration using the command below:

- source ~/.bashrc

Step 7: verify the openssl command as below

- openssl version

OpenSSL is installed.

It is important to use different certificate subject parameters for your CA, server and clients. If the certificates appear identical, even though generated separately, the broker/client will not be able to distinguish between them and you will experience difficult to diagnose errors.

Step 1: Generate a certificate authority certificate and key using the command below:

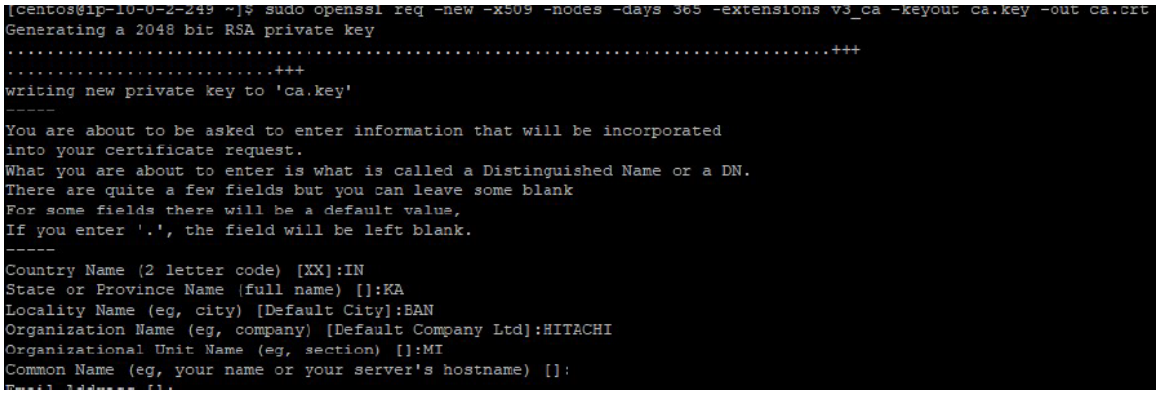
sudo openssl req -new -x509 -nodes -days 365 -extensions v3_ca -keyout ca.key -out ca.crt

Upon executing above command, it will prompt you to enter the values for below:

1. Country Name (2 letter code):
2. State or Province name (full name):
3. Locality Name (eg, city) [Default City]:
4. Organization Name (eg, company) [Default Company Ltd]:
5. Organizational Unit Name (eg, section) []:
6. Common Name (eg, your name or your server's hostname) []: IP Adress of instance
7. Email Address []: it is Optional

Note: When prompted for the CN (Common Name), please enter either your server (or broker) hostname or domain name.(Enter IP address or DNS of broker). IP address is preferable.

Refer below screenshot for reference:



Step 2: Generate server certificate

Once ca.crt is generated, we can generate server certificate. Below are the steps to generate server certificate.

- **Generate a server key without encryption.**

  sudo openssl genrsa -out server.key 2048

- **Generate a certificate signing request to send to the CA.**

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

Upon executing above command, it will prompt you to enter the values for below:

1. Country Name (2 letter code):
2. State or Province name (full name):
3. Locality Name (eg, city) [Default City]:
4. Organization Name (eg, company) [Default Company Ltd]:
5. Organizational Unit Name (eg, section) []:
6. Common Name (eg, your name or your server's hostname) []: IP Adress of instance
7. Email Address []: it is Optional

Note: When prompted for the CN (Common Name), please enter either your server (or broker) hostname or domain name.(Enter IP address or DNS of broker). IP address is preferable.

Below is the sample screen shot for reference:

```
[root@ip-10-0-2-249 centos]# sudo openssl req -out server.csr -key server.key -new
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:In
State or Province Name (full name) []:ka
Locality Name (eg, city) [Default City]:bn
Organization Name (eg, company) [Default Company Ltd]:hi
Organizational Unit Name (eg, section) []:mi
Common Name (eg, your name or your server's hostname) []:[redacted]
Email Address []:
```

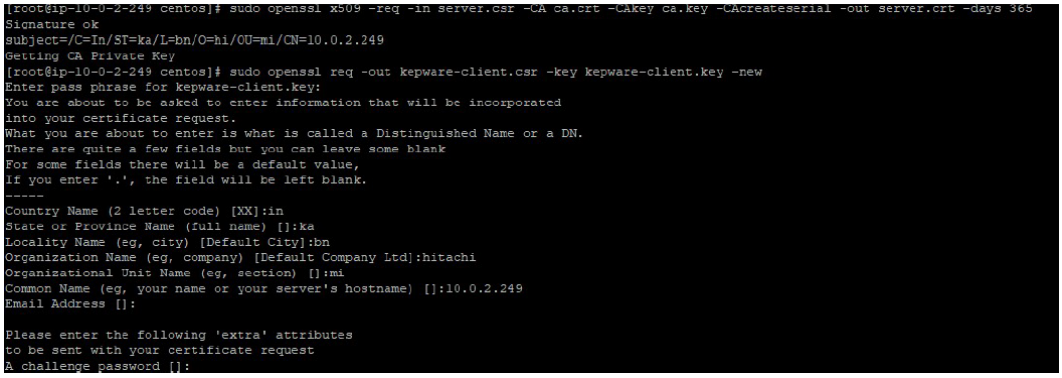
- **Send the CSR to the CA, or sign it with your CA key using below command and replace <duration> with preferable numbers in days e.g. 365**

  sudo openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days <duration>

Upon executing above command, it will prompt you to enter the values for below:

1. Country Name (2 letter code):
2. State or Province name (full name):
3. Locality Name (eg, city) [Default City]:
4. Organization Name (eg, company) [Default Company Ltd]:
5. Organizational Unit Name (eg, section) []:
6. Common Name (eg, your name or your server's hostname) []: IP Adress of instance
7. Email Address []: it is Optional

Below is the sample screen shot for reference:



Step 3: Generate client certificates for Kepware and nifi.

Note: It is important to use different certificate **subject** parameters for your CA, **server and clients**. If the certificates appear identical, even though generated separately, the broker/client will not be able to distinguish between them and you will experience difficult to diagnose errors. So for Kepware and Nifi we have to generate two different client certificates.

Certificate for kepware:

- **Generate a client key.**

  sudo openssl genrsa -des3 -out kepware-client.key 2048

  *When you encounter "Enter pass phrase for kepware-client.key:" in the cli please enter a pass phrase as "**kepwareclient1**"*

> **Note:** This pass phrase will be used in below steps.

Below is the sample screen shot for reference.

```
[root@ip-10-0-2-249 centos]# sudo openssl genrsa -des3 -out kepware-client.key 2048
Generating RSA private key, 2048 bit long modulus
...+++
..............+++
e is 65537 (0x10001)
Enter pass phrase for kepware-client.key:
Verifying - Enter pass phrase for kepware-client.key:
```

- **Generate a certificate signing request to send to the CA.**

  sudo openssl req -out kepware-client.csr -key kepware-client.key -new

Below is the sample screen shot for reference:

```
[root@ip-10-0-2-249 centos]# sudo openssl req -out kepware-client.csr -key kepware-client.key -new
Enter pass phrase for kepware-client.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:in
State or Province Name (full name) []:ka
Locality Name (eg, city) [Default City]:bn
Organization Name (eg, company) [Default Company Ltd]:hitachi
Organizational Unit Name (eg, section) []:mi
Common Name (eg, your name or your server's hostname) []:10.0.2.249
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

- **Send the CSR to the CA, or sign it with your CA key:**

  sudo openssl x509 -req -in kepware-client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out kepware-client.crt -days 365

Below is the sample screen shot for reference:

```
[root@ip-10-0-2-249 centos]# sudo openssl x509 -req -in kepware-client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out kepware-client.crt -days 365
Signature ok
subject=/C=in/ST=ka/L=bn/O=hitachi/OU=mi/CN=10.0.2.249
Getting CA Private Key
```

Certificate for Nifi:

Note: nifi client key and certificates will be same as the steps followed to generate kepware certificates.

- **Generate a client key.**

  sudo openssl genrsa -des3 -out nifi-client.key 2048

  *When you encounter "Enter pass phrase for nifi-client.key:" in the cli please enter a pass phrase as "**nificlient1**"*

- **Generate a certificate signing request to send to the CA.**

  sudo openssl req -out nifi-client.csr -key nifi-client.key -new

- **Send the CSR to the CA, or sign it with your CA key:**

  sudo openssl x509 -req -in nifi-client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out nifi-client.crt -days 365

Now we will get all the required certificates for MI edge SSL configuration.

Certificate Authority	**ca.crt** **ca.key** **ca.srl**
Server certificates	**server.crt** **server.key** **server.srl**
Kepware client certificate	**Kepware-client.crt** **Kepware-client.key** **Kepware-client.csr**
Nifi client certificate	**nifi-client.crt** **nifi-client.key** **nifi-client.csr**

This concludes SSL certificate generation steps.

Chapter 3: How to make the Data Ingestion API more secure

When you call the Data Ingestion API through a network, you might want to do it in a more secure manner. The easiest way to make the API more secure is to configure NGINX in the MI Frontend instance as a reverse proxy server for the Data Ingestion API.

An example of a configuration that enables basic authentication for the Data Ingestion API is provided below(“dataingestion.conf”).

```
server {
listen 15454;
server_name IP-address-of-MI-Frontend-instance;
sendfile off;
etag off;
if_modified_since off;
auth_basic "Data Ingestion API";
auth_basic_user_file /etc/nginx/.htpasswd;
location = /swagger-ui.html {
deny all;
}
location / {
proxy_pass http://IP-address-of-MI-Backend-instance:5454/;
}
}
```

Note: You will need to modify the yellow lines to match your environment.

You can enable the setting by performing the following steps:

Step 1: Install NGINX if not installed. For setup procedure of NGINX, please refer ENG_4MDataRecorder_Guide.pdf document Step 2.2.1-2.2.3.

Step 2: Store the "dataingestion.conf" file in the "/etc/nginx/conf.d" directory.

Step 3: Install the htpasswd command, and then use it to create the "/etc/nginx/.htpasswd" file.

Step 4: Restart NGINX:

nginx -t

systemctl stop nginx

systemctl start nginx

Step 5: Open port 15454 by changing the firewall setting of the MI Frontend instance.

firewall-cmd --list-all

firewall-cmd --add-port=15454/tcp --permanent

firewall-cmd --reload

firewall-cmd --list-all

Step 6: Limit 5454 port access of the MI Backend instance by changing the firewall setting of the instance

firewall-cmd --remove-port=5454/tcp --permanent

firewall-cmd --permanent --add-rich-rule="rule family="ipv4" source address="IPaddress-of-MI-Frontend-instance" port protocol="tcp" port="5454" accept"

firewall-cmd --reload

firewall-cmd --list-all

After you perform these steps, the Data Ingestion API endpoint URL becomes the following.

http://IP-address-of-MI-Production-Analytics-instance:15454/endpoint-path-ofeach-API

You can implement any other customer-specific requirements in the file, such as TLS settings, access control, and other authentication methods.

For details, see the official documentation of NGINX.

Chapter 4: How to make the QR code APIs more secure

When you call the QR code APIs through a network, you might want to do it in a more secure manner. The easiest way to make the API more secure is to configure NGINX in the MI Frontend instance as a reverse proxy server for the QR code APIs.

An example of a configuration that enables basic authentication for the QR API is provided below(“qrcodeapis.conf”).

```
server { listen 15656;
server_name IP-address-of-MI-Frontend-instance;
sendfile off;
etag off;
if_modified_since off;
auth_basic "QR API";
auth_basic_user_file /etc/nginx/.htpasswd;
location = /swagger-ui.html {
deny all;
}
location / {
proxy_pass http://IP-address-of-MI-Backend-instance:5656/;
}
}
```

Note: You will need to modify the yellow lines to match your environment.

You can enable the setting by performing the following steps:

Step 1 : Install NGINX if not installed. For setup procedure of NGINX, please refer ENG_4MDataRecorder_Guide.pdf document Step 2.2.1-2.2.3.

Step 2: Store the " qrcodeapis.conf" file in the "/etc/nginx/conf.d" directory.

Step 3: Install the htpasswd command, and then use it to create the "/etc/nginx/.htpasswd" file.

Step 4: Restart NGINX:

nginx -t

systemctl stop nginx

systemctl start nginx

Step 5: Open port 15656 by changing the firewall setting of the MI Frontend instance

firewall-cmd --list-all

firewall-cmd --add-port=15656/tcp --permanent

firewall-cmd --reload # firewall-cmd --list-all

Step 6: Limit 5656 port access of the MI Backend instance by changing the firewall setting of the instance

firewall-cmd --remove-port=5656/tcp --permanent

firewall-cmd --permanent --add-rich-rule="rule family="ipv4" source address="IPaddress-of-MI-Frontend-instance" port protocol="tcp" port="5656" accept"

firewall-cmd --permanent --add-rich-rule="rule family="ipv4" source address="IPaddress-of-MI-Frontend-instance" port protocol="tcp" port="5656" accept"

firewall-cmd --reload # firewall-cmd --list-all

After you perform these steps, the Data Ingestion API endpoint URL becomes the following. http://IP-address-of-MI-Frontend-instance:15656/endpoint-path-ofeach-API

You can implement any other customer-specific requirements in the file, such as TLS settings, access control, and other authentication methods. For details, see the official documentation of NGINX.

Chapter 5: MI edge SSL Configuration

This document describes how to configure SSL/TLS for the edge component used for MI edge. To secure edge pipeline we must secure Kepware,

Mosquitto and Nifi. So, this document covers the steps to configure SSL/TLS for Kepware, Mosquitto and Apache Nifi.

Assuming we have below certificates which is required for Mosquitto SSL configuration:

- Certificate authority e.g ca.crt
- Certificate authority Key e.g. ca.key
- Server certificate e.g. server.crt
- Server certificate key e.g. server.key

Step 1: Copy ca.crt, ca.key, server.key and server.crt into "/etc/mosquitto/certs" directory.

Note: In Ubuntu "/etc/mosquitto/certs" directory is available. But in case of centos certs directory won't be available.

In case "/etc/mosquitto/certs" is not found please create "certs" directory using below command:

- *mkdir /etc/mosquitto/certs*

To copy the files execute below command:

- *cp ca.* /etc/mosquitto/certs*
- *cp server.* /etc/mosquitto/certs*

Step 2: Generate password file for Mosquitto by executing following commands (Replace <username> as per desired Username in 4th command and when prompted for Password enter desired Password):

- *sudo su –*
- *mkdir /etc/mosquitto/pwd*
- *cd /etc/mosquitto/pwd*
- *mosquitto_passwd -c mosquitto_pwd_file <username>*

Step 3: Edit /etc/mosquitto/mosquitto.conf

- vi /etc/mosquitto/mosquitto.conf

Append following lines at the end of the file:

port 8883

cafile /etc/mosquitto/certs/ca.crt

certfile /etc/mosquitto/certs/server.crt

keyfile /etc/mosquitto/certs/server.key

allow_anonymous false

require_certificate true

password_file /etc/mosquitto/pwd/mosquitto_pwd_file

Once you above steps are done please save and close the file using Esc and :wq@

Step 4: Restart Mosquitto broker to apply SSL configuration:

For Ubuntu: service restart mosquitto

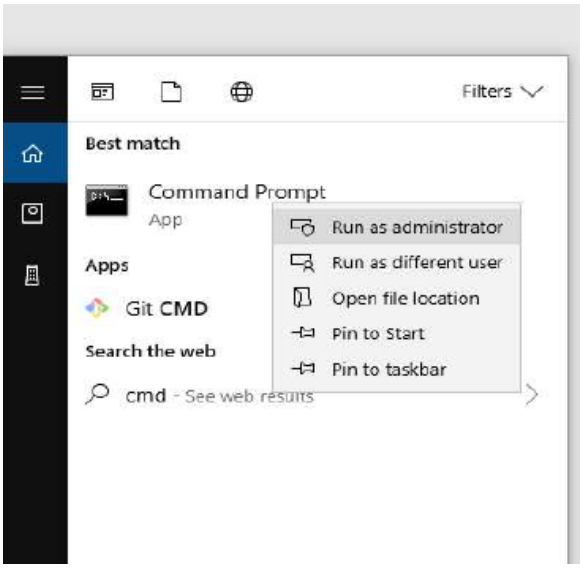
For Centos: systemctl restart mosquitto

SSL configuration for Mosquitto is done.

Step 1: Assuming client certificates (Example: **kepware-client.crt** file) are available. These client certificates are generated using same certificate authority which is used for server certificate.

Step 2: Add **ca.crt** (Certificate authority) to root trust certificate. Following steps to be followed to add certificate authority to root trust certificate:

- Launch windows command prompt by running it as administrator as shown below:



- In the command prompt window, navigate to the location of the certificate file **ca.crt.**
- Execute following command:

  certutil -addstore "Root" **ca.crt**
- The import is successful if following output appears on screen:

  CertUtil: -addstore command completed successfully.

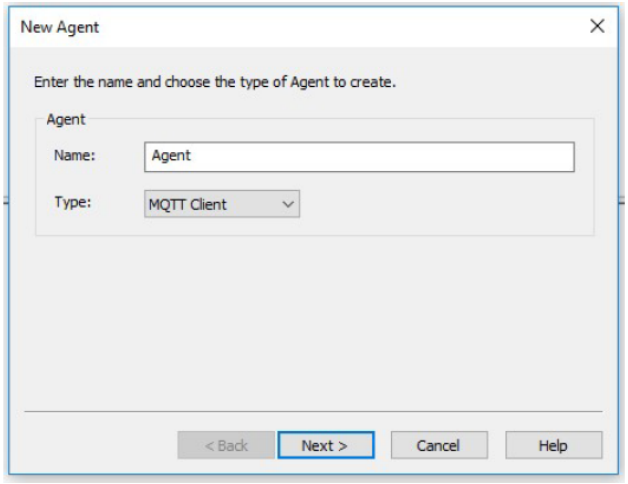
Step 3: Kepware Agent configuration

1. Please skip this step if you have already MQTT agent available.

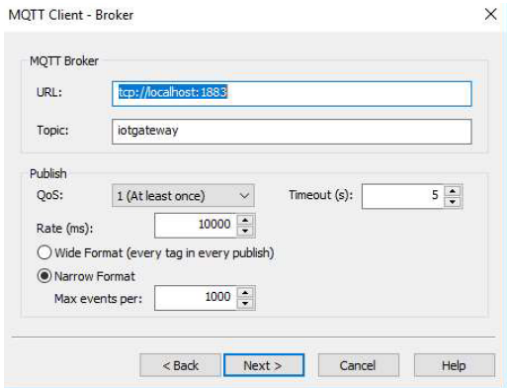
Below are the steps to new MQTT Agent.

- Open KEPServerEx configuration and expand IoT gateway.
- Right click on IOT Gateway and click on New Agent.

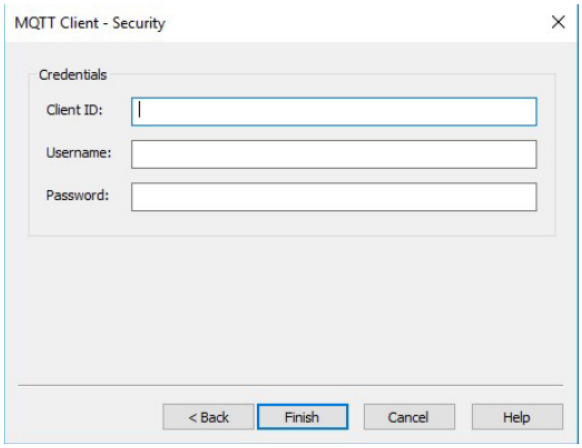
- Enter Agent name and select Type as MQTT Client and click on next button. Refer below screenshot.



- Enter broker URL and topic name to which sensor data will be published and click on next button. Below is the screenshot for reference.

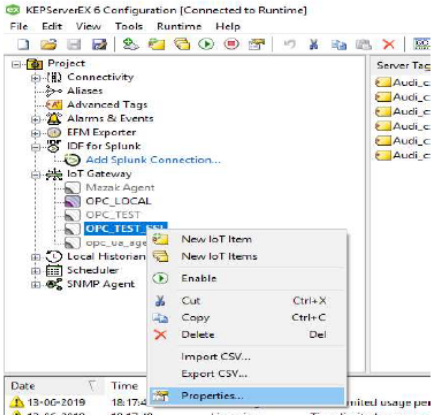


- Enter client ID. Also enter Username and password which you have created during “Step2 of Mosquitto SSL configuration”.



Upon clicking Finish button a MQTT Agent will be created and displayed under IoT Gateway.

2. Right click on the MQTT agent which is configured earlier or you have created using above step. Select “Properties” option:

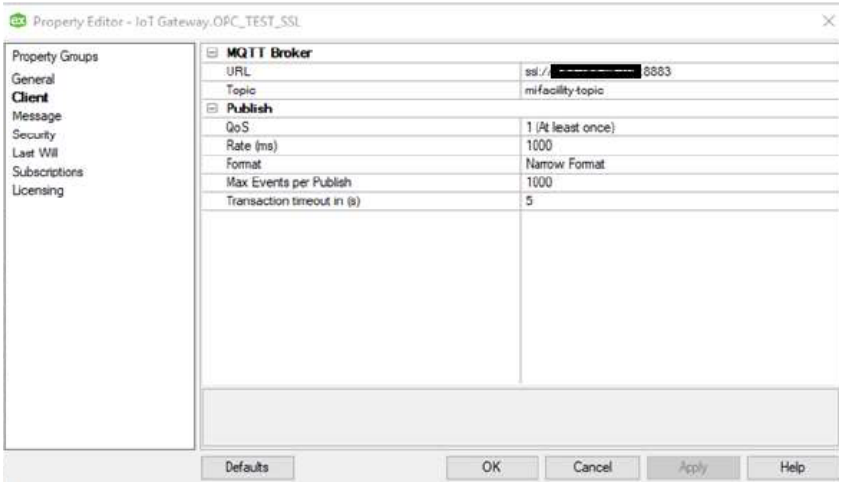


3. Once clicked on Properties menu item, Property editor will be launched. Then select “**Client**” under Property Groups on left hand side and update URL property for

MQTT Broker by replacing **<MQTT_BROKER_IP>** with the IP of MQTT Broker as ssl://**<MQTT_BROKER_IP>**:8883

Other properties under **Publish** section will remain as it is.

Below is the Screen shot for reference.



4. Now enable client certificate by selecting "**Security**" section under "Property Groups" on left hand side and update following properties:

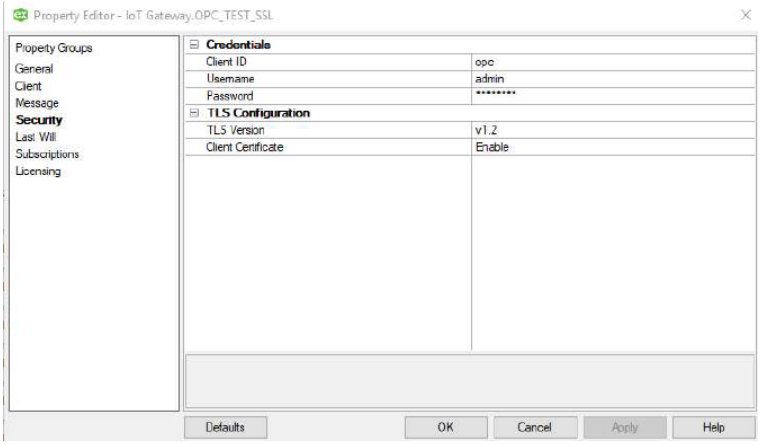
   a. **Username**: Update username same as the username used to generate Mosquitto password file in Step #2 of "Mosquitto SSL/TLS configuration"

   b. **Password:** Update password same as the password used to generate Mosquitto password file in Step #2 of "Mosquitto SSL/TLS configuration"

   c. **TLS Version:** Specify the TLS version as per SSL certificate

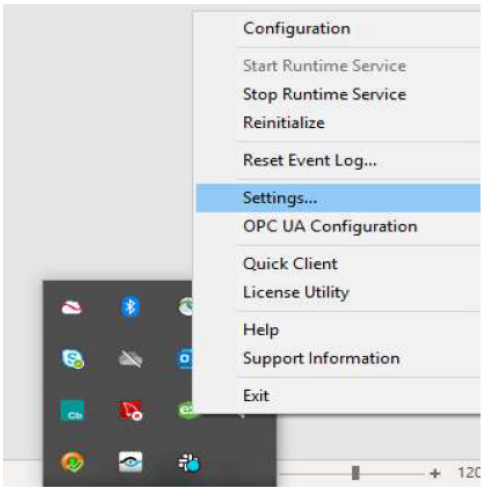
   d. **Client Certificate:** Enable

Below is the screen shot for reference:



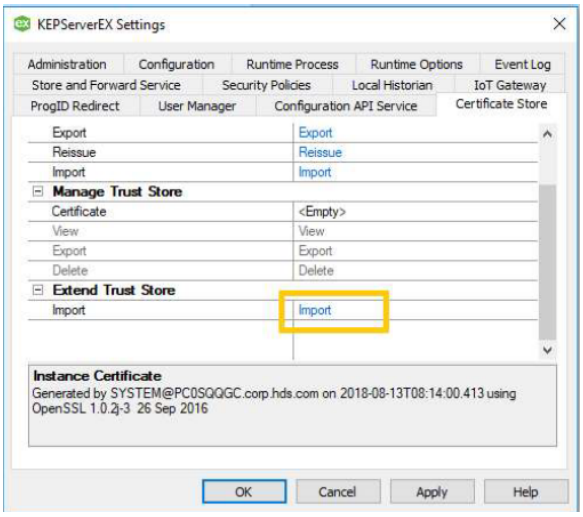
5. **Import MQTT client certificates:**

   a. Open KepServerEx Administration from the tool bar task list as provided in below screen shot. Click on Settings menu item:



KEPServerEX Settings pop up will open with multiple tabs. Click on “Certificate Store” tab then click on Import link under “Extend Trust Store”.

Below is the screen shot for reference.



Select client certificate file (Example: kepware-client.crt file) and click on apply.

Above steps concludes Kepware MQTT SSL configuration.

Assuming client certificates (e.g. **nifi-client.crt , ca.crt and nifi-client.key**) are available. These client certificates are generated using same certificate authority which is used for server certificate.

Step 1: Generate client key store and trust store from **nifi-client.crt** and **nifi-client.key** file.

- Execute following command to create PKCS12 keystore from client key and client certificate by replacing **<PATH_TO_ nifi-client.crt>** and **<PATH_TO_ nifi-client.key>** with **absolute path of nifi-client.crt** and **nifi-client.key**:

  e.g.: If **nifi-client.crt** and **nifi-client.key** are located at /usr/local/ssl then **<PATH_TO_ nifi-client.crt>** will be at **/usr/local/ssl/nifi-client.crt** and **<PATH_TO_ nifi-client.key>** will be at **/usr/local/ssl/nifi-client.key**

  *openssl pkcs12 -nodes -export -name client-cert -in **<PATH_TO_ nifi-client.crt>** -inkey **<PATH_TO_ nifi-client.key>** -out clientkeystore.p12*

  Upon running above command, it will prompted you to enter the following:

  - Enter pass phrase for ./nifi-client.key: Enter pass phrase which you have enter while creating nifi-client.key. In this case it will be **nificlient1**.
  - Enter Export Password: Any password you can enter
  - Verifying - Enter Export Password:

- Execute below command to convert a PKCS12 keystore into a JKS keystore

  keytool -importkeystore -destkeystore nifi-client.keystore -srckeystore clientkeystore.p12 -srcstoretype pkcs12 -alias client-cert

  Upon running above command, it will prompted you to enter the following:

  - Enter destination keystore password: Enter nificlient1
  - Re-enter new password: Enter nificlient1
  - Enter source keystore password: Enter nificlient1

- Execute following command to import a server's certificate of authority to the client's trust store by replacing **<path_to_ca.crt>** with **absolute path of ca.crt file path**. E.g. **/usr/local/ssl/ca.crt**

  *keytool -import -alias server-cert -file **<path_to_ca.crt>** -keystore nifi-client.truststore*

  Upon running above command, it will prompted you to enter the following:

  - Enter keystore password: Enter nificlient1
  - Re-enter new password:
  - Trust this certificate? [no]: Enter yes

- Execute below command to import a client's certificate to the client's trust store by replacing **<PATH_TO_ nifi-client.crt>** with **absolute path of nificlient.crt** file path. E.g. **/usr/local/ssl/nifi-client.crt**

  *keytool -import -alias client-cert -file **<PATH_TO_ nifi-client.crt>** -keystore nifi-client.truststore*

  Upon running above command, it will prompted you to enter the following:

  - Enter keystore password: Enter nificlient1.

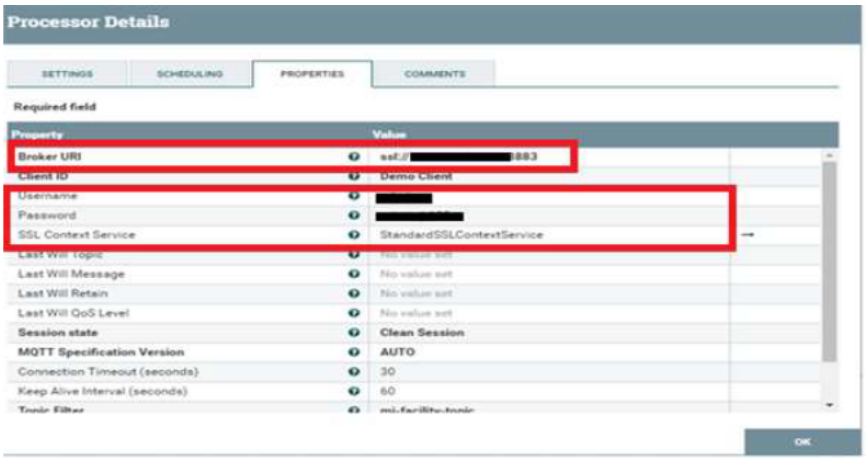
Above steps will generate “nifi-client.keystore” and “nifi-client.truststore” file.

Step 2: Configure MQTT Consumer Processor

Once the files are generated, we can use these files to configure MQTT consumer processors in the NIFI template used for mi data pipeline.

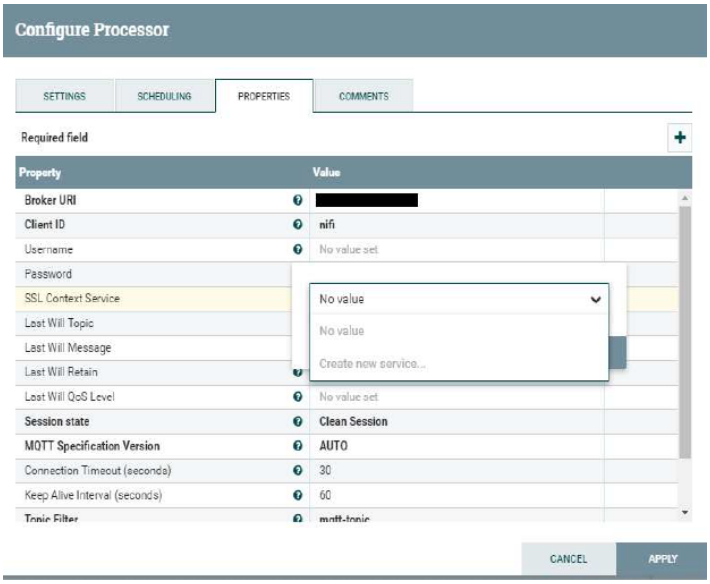
- Access Nifi console ex: http://<IP_ADDRESS_OF_NIFI>:8080/nifi/
- Double click on processor group which we are going to modify and double click on "Consume MQTT" to access Configure Processor.
- Select Properties tab and update the values specified in the instructors below.
  - **Broker URI**: Update Broker URI by replacing **<MQTT_BROKER_IP>** with the IP of MQTT Broker as ssl://**<MQTT_BROKER_IP>**:8883.
  - **Username:** Update username same as the username used to generate Mosquitto password file in Step #2 of “Mosquitto SSL/TLS configuration”.
  - **Password:** Update password same as the password used to generate Mosquitto password file in Step #2 of “Mosquitto SSL/TLS configuration”

Below is the screen shot for reference.

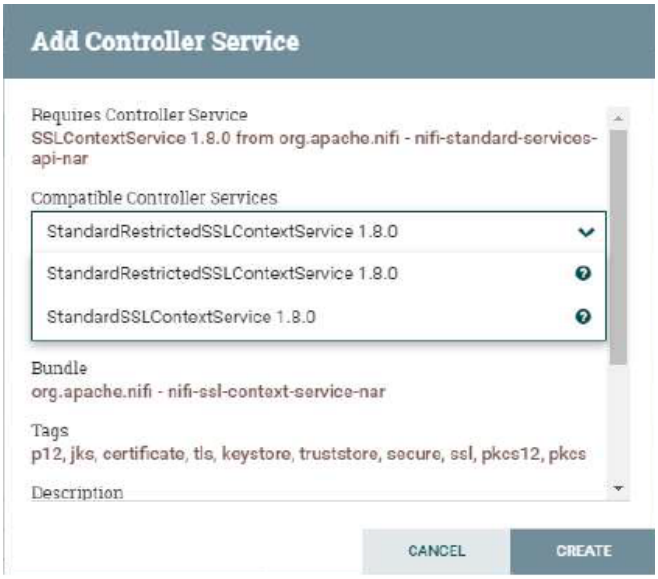

Processor Details
SETTINGS
SCHEDULING
PROPERTIES
COMMENTS
Required field
Property
Value
Broker URI
ssl://[redacted]:883
Client ID
Demo Client
Username
[redacted]
Password
[redacted]
SSL Context Service
StandardSSLContextService
Last Will Topic
No value set
Last Will Message
No value set
Last Will Retain
No value set
Last Will QoS Level
No value set
Session state
Clean Session
MQTT Specification Version
AUTO
Connection Timeout (seconds)
30
Keep Alive Interval (seconds)
60
OK

- **“SSL Context Service”:**

  Click on the value section of SSL Context Service and select “Create new Service”.



  Select “StandardSSLContextService 1.8.0” from the “Compatible Controller Services” drop down and click on create.



Once the value is populated for “**SSL Context Service**” property, click on the arrow shown right most column against the property “**SSL Context Service**”.

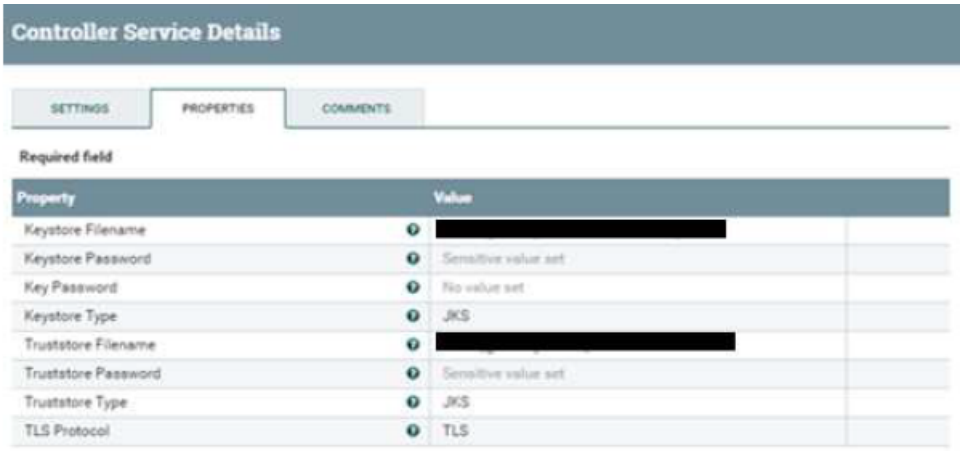
Below pop up will open to modify Controller Service which we have created.

Click on the configure icon which is highlighted on the above screenshot

A pop up will lunch to update SSL Context Service. Update following properties by selecting "**PROPERTIES**" tab in the pop up.

- **Keystore filename :** <Path_to_nifi-client.keystore>
- **Keystore Type:** JKS
- **Truststore FileName :** < Path_to_nifi-client.truststore>
- **Truststore Type:** JKS

Note: Replace <Path_to_nifi-client.keystore> and < Path_to_nifi-client.truststore> with actual file path.



Once you updated all required fields , you can enable the service.

Step 3: Right click on MQTT consumer process and select option start to Start MQTT consumer process.

This ends the SSL configuration of MI data pipeline i.e. from Kepware through Mosquitto to Nifi.

Chapter 6: NiFi_Security

Secure NiFi with SSL

This guide describes how to enable SSL for NiFi and configure client to communicate with NiFi over SSL. As part of enabling SSL, NiFi will also automatically enable authentication requiring all users to provide a client certificate to access the NiFi UI unless an additional authentication method is configured. Information on additional authentication methods can be found in the NiFi System Administrator's Guide under User Authentication. Please see Enabling SSL for NiFi for additional instructions on enabling SSL for NiFi.

Client certificates are used to authenticate the default admin user (i.e. Initial Admin Identity) and other NiFi nodes. It can either be a self-signed certificate or one signed by a CA (certificate authority). A self-signed certificate is the most common option and is the easiest to install, but a certificate signed by a CA provides an additional chain of trust for those that need it.

It is recommended to start with a self-signed certificate and verify that everything works. It can then be replaced with a certificate signed by a CA.

NiFi provides a toolkit for generating a self-signed CA and the certificates necessary for SSL communication and authentication. However, web browsers will display an "untrusted certificate" error when accessing the NiFi UI and will only allow access after acknowledging the risk. Adding the CA to the web browser's trust store will prevent the error from being displayed. We are providing shell script to generate NiFi self-signed certificates.

1. Execute following commands to generate Keystore and Truststore files

   Note: Keep note of values of [KEYSTORE_PASSWORD] and [TRUSTSTORE_PASSWORD] as it is used

   in further steps also):

   - cd /opt/nifi-1.8.0/cert
   - ./createKey.sh keystore [KEYSTORE_PASSWORD] [TRUSTSTORE_PASSWORD]

2. Execute following commands to generate certificates for Admin user:

   - cd /opt/nifi-1.8.0/cert
   - ./createKey.sh [ADMIN_USERNAME] [ADMIN_PASSWORD] [TRUSTSTORE_PASSWORD]

   Note: [TRUSTSTORE_PASSWORD] □ Same value as used in Step #1

Above step will generate following files:

- [ADMIN_USERNAME].cer
- [ADMIN_USERNAME].p12
- [ADMIN_USERNAME].jks
- keystore.p12
- keystore.cer
- keystore.jks
- truststore.jks

3. Update following properties in **/opt/nifi-1.8.0/conf/nifi.properties** file:

   # Site to Site properties

   nifi.remote.input.secure=true

   # web properties

   nifi.web.http.host=

   nifi.web.http.port=

   nifi.web.http.network.interface.default=

   nifi.web.https.host=

   nifi.web.https.port=8443

   # security properties #

   nifi.security.keystore=/opt/nifi-1.8.0/cert/keystore.jks

   nifi.security.keystoreType=JKS

   nifi.security.keystorePasswd=**[KEYSTORE_PASSWORD value used in Step #1]**

   nifi.security.keyPasswd=**[KEYSTORE_PASSWORD value used in Step #1]**

   nifi.security.truststore=/opt/nifi-1.8.0/cert/truststore.jks

   nifi.security.truststoreType=JKS

   nifi.security.truststorePasswd=**[TRUSTSTORE_PASSWORD value used in Step #1]**

   nifi.security.user.authorizer=managed-authorizer

   nifi.security.user.login.identity.provider=

   nifi.security.ocsp.responder.url=

   nifi.security.ocsp.responder.certificate=

   nifi.security.needClientAuth=true

   nifi.security.support.new.account.requests=true

4. Configure NiFi to give admin access to the client certificate by editing **/opt/nifi-1.8.0/conf/authorizers.xml** and adding the Initial Admin Identity property as shown below by replacing **[ADMIN_USERNAME used in step #2]** value:

Note: The *Initial Admin Identity* must exactly match the subject of the client certificate as displayed by keytool. NiFi will automatically add a matching user to /opt/nifi-1.8.0/conf/users.xml. If the client certificate is updated with a different CN, then users.xml will also need to be updated.

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<authorizers>

<userGroupProvider>

<identifier>file-user-group-provider</identifier>

<class>org.apache.nifi.authorization.FileUserGroupProvider</class>

<property name="Users File">./conf/users.xml</property>

<property name="Legacy Authorized Users File"></property>

<property name="Initial User Identity 1">CN=**[ADMIN_USERNAME used in step #2]**, OU=NIFI</property>

</userGroupProvider>

<accessPolicyProvider>

<identifier>file-access-policy-provider</identifier>

<class>org.apache.nifi.authorization.FileAccessPolicyProvider</class>

<property name="User Group Provider">file-user-group-provider</property>

<property name="Authorizations File">./conf/authorizations.xml</property>

<property name="Initial Admin Identity">CN=**[ADMIN_USERNAME used in step #2]**, OU=NIFI</property>

<property name="Legacy Authorized Users File"></property>

<property name="Node Identity 1"></property>

</accessPolicyProvider>

<authorizer>

<identifier>managed-authorizer</identifier>

<class>org.apache.nifi.authorization.StandardManagedAuthorizer</class>

<property name="Access Policy Provider">file-access-policy-provider</property>

</authorizer>

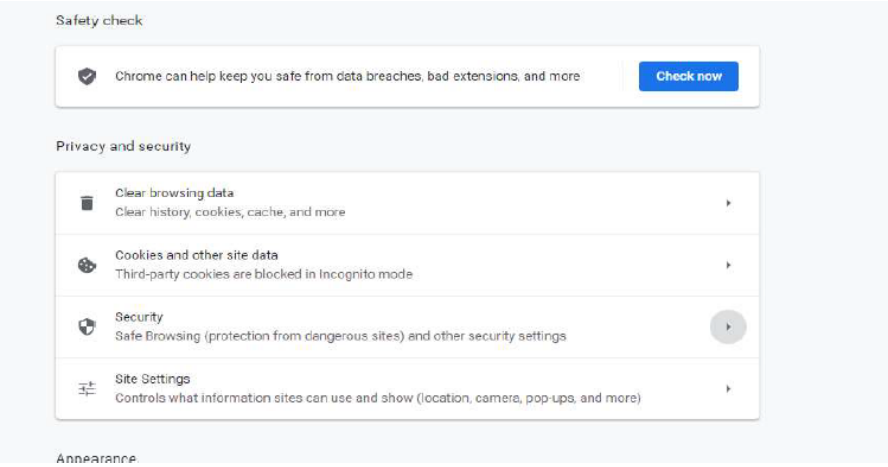
</authorizers>

5. Restart NiFi by executing following commands:
   - **/opt/nifi-1.8.0/bin/nifi.sh stop**
   - **/opt/nifi-1.8.0/bin/nifi.sh start**

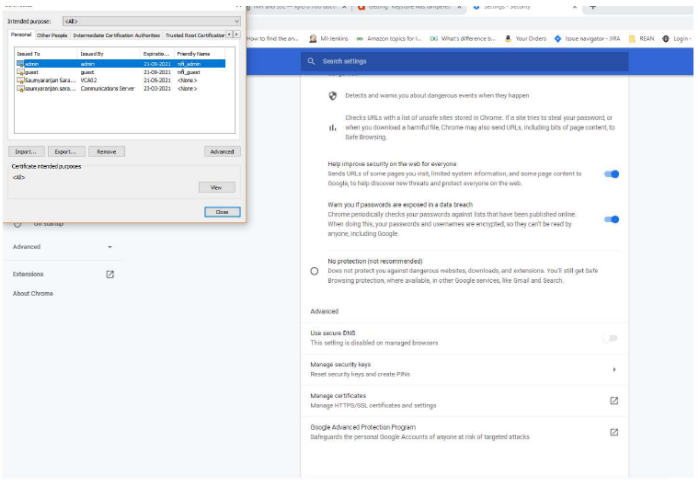
Web browsers can also be configured to use the client certificate to access NiFi.

Importing the Client Cert on windows machine:

1. Copy the .p12 file that you created at step #2 to your Desktop.
2. Open chrome browser setting.
3. Go to Privacy and Security from the left navigation menu.
4. Go to security option.



5. Select security and click on manage certificates.



Import all the user certificates. While importing you have to provide .p12 file and password which you have provided in the step#2.

6. Once completed, you should able to access NiFi web URL as: https://{HOST_IP}:8443/nifi
7. Update **/opt/startProcess.sh** by replacing 8080 with 8443.