Microsoft Azure Guide

Rev.: 1.0

Date: 2021-12-12

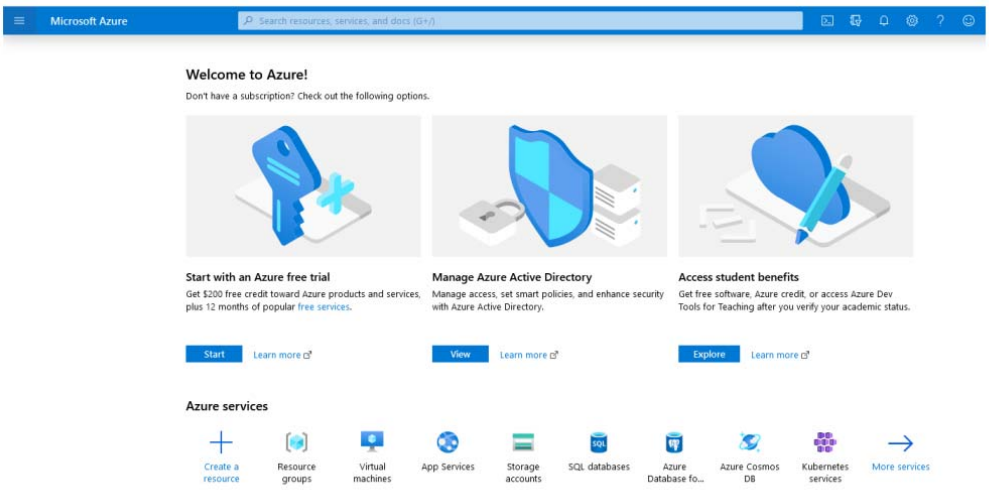
1. Microsoft Azure:

1.1. Create a Free Account

Create a free Azure account in https://azure.microsoft.com/en-gb/free/.

1.2. Enter Azure Portal

Enter Azure Portal via link https://portal.azure.com/#home.



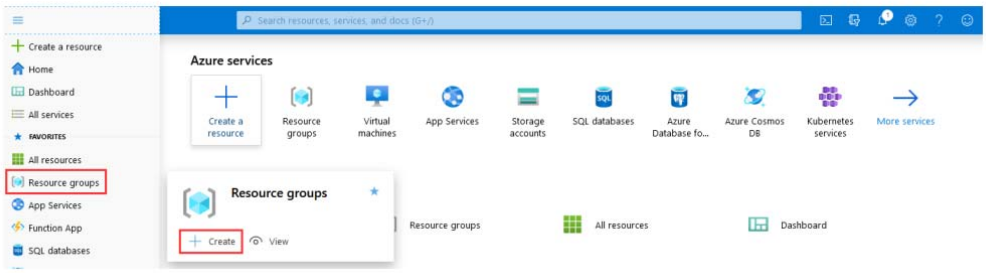
1.3. Create a Resource Group

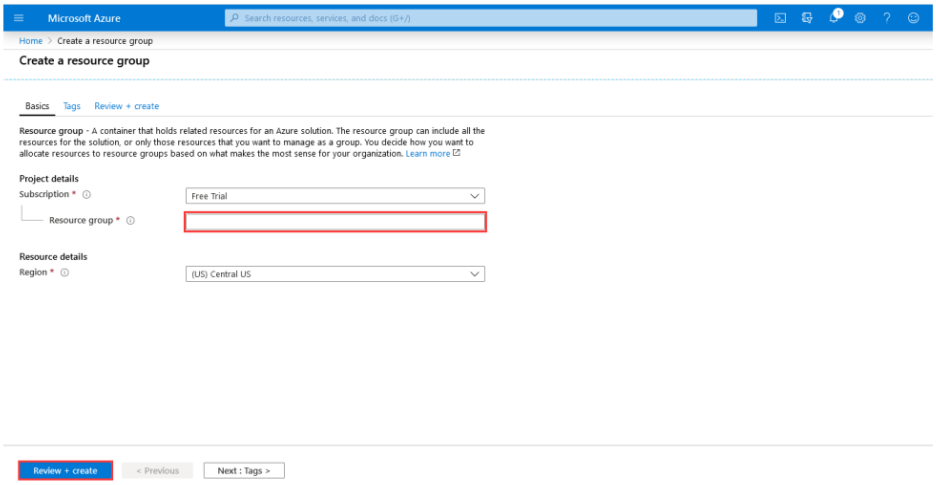
a) In the left navigation bar, move your mouse cursor to "**Resource groups**", and then click "**Create**" to
create a resource group.

b) Fill a resource group name in the box after "**Resource group**".

c) Click "**Review + create**".

d) Click "**Create**", and then check whether the resource group has been created successfully through
clicking "**Resource groups**" button in the homepage again.





1.4. Create an IoT Hub Resource

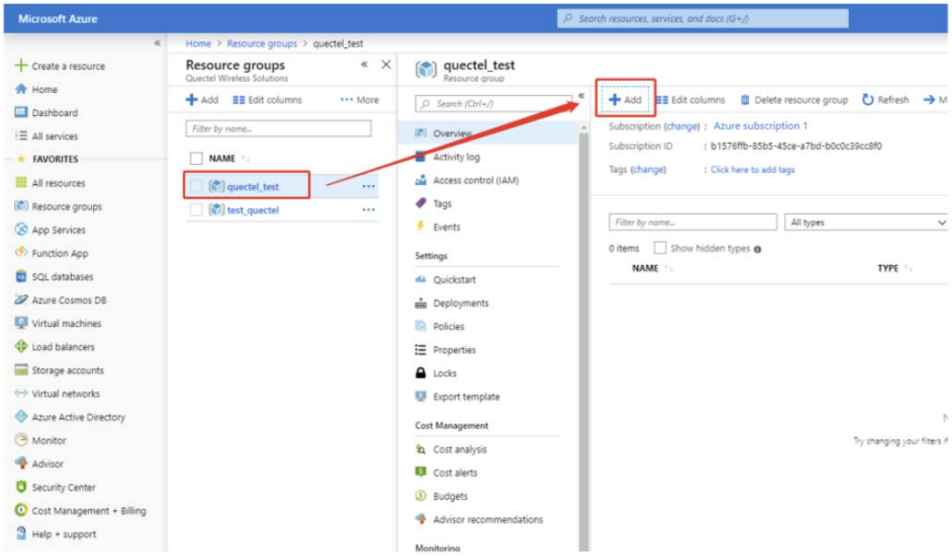
a) Click the name of the newly created resource group, and then click "**Add**".

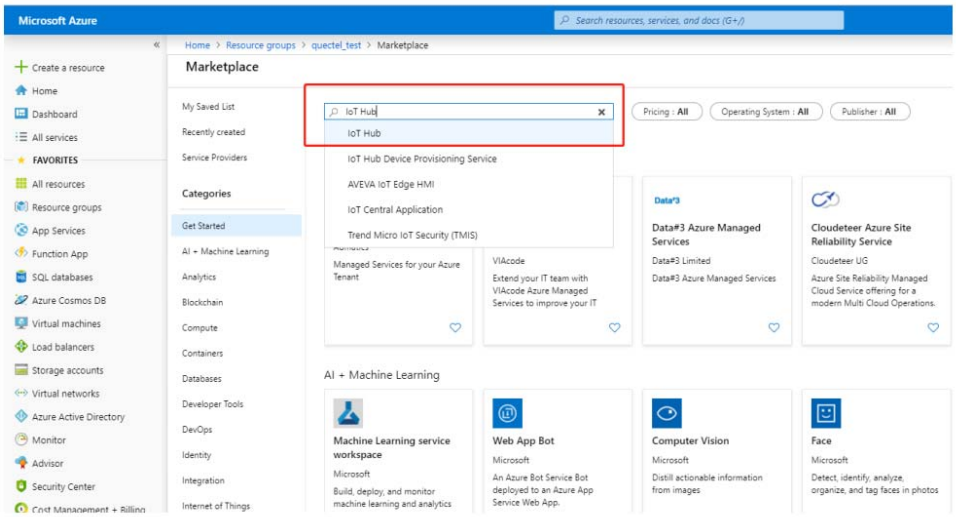
b) Afterwards, there will be a search box indicating "**Search the Marketplace**". Input "IoT Hub" in the search box to enter IoT Hub page.

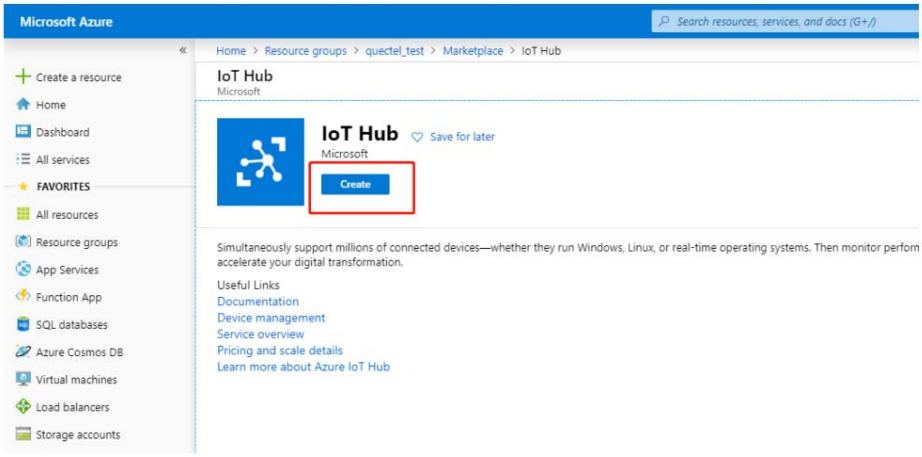
c) Click "**Create**" in IoT Hub page.

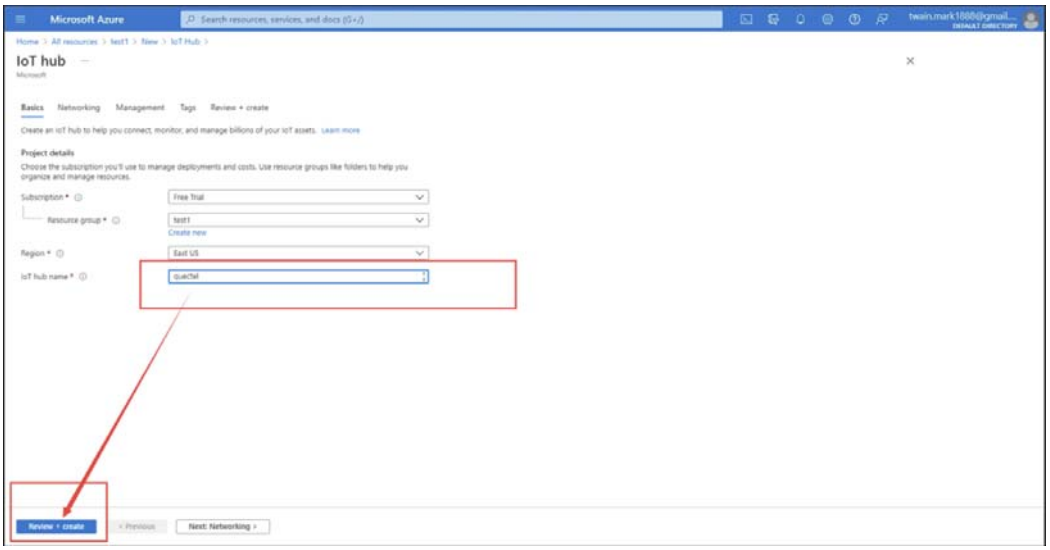
d) Name your IoT Hub in the box after "**IoT Hub Name**", and then click "**Review + create**".

e) Finally click "**Create**" to finish the operation.



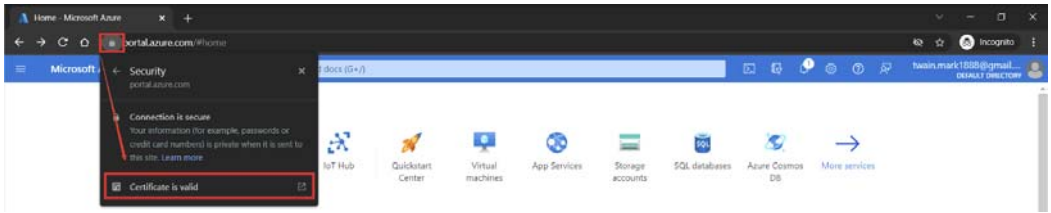


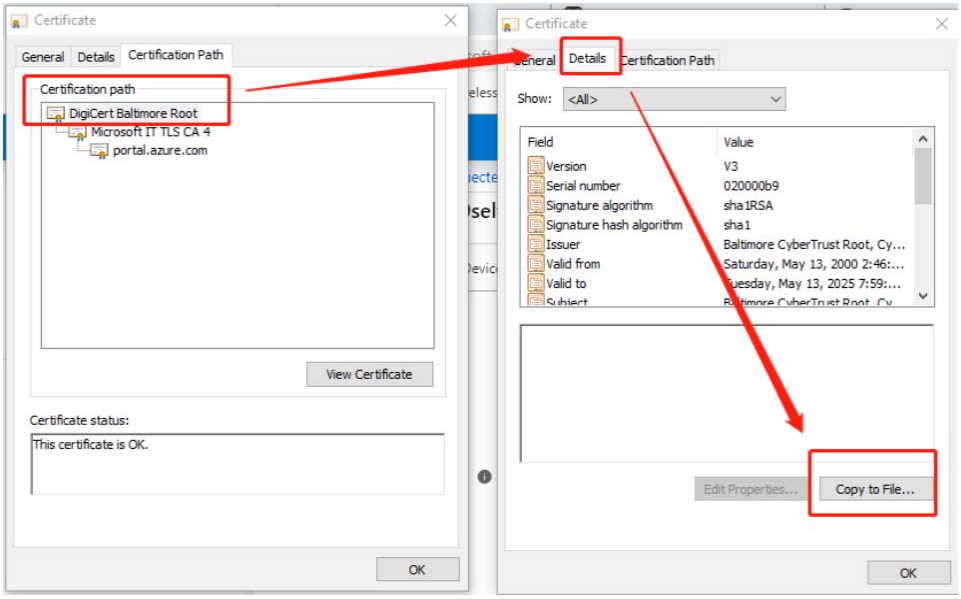


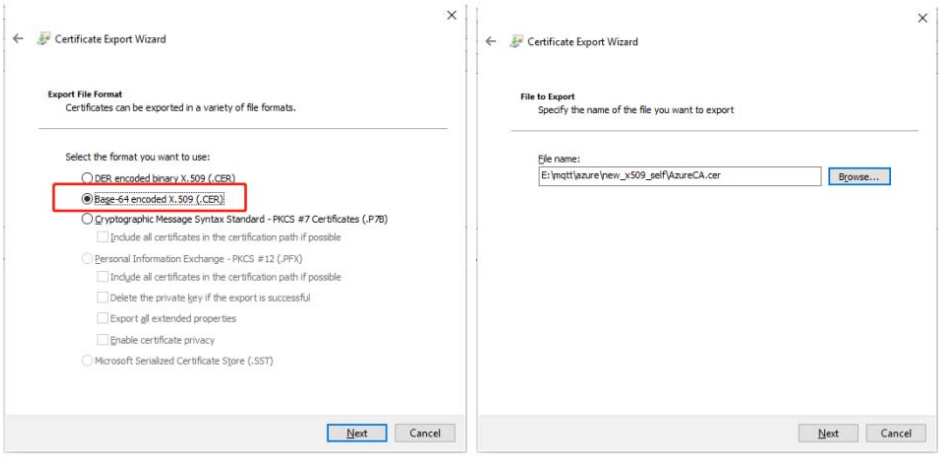


1.5. Get Azure Root CA certificate

The Azure root CA certificates can be got with Google Chrome as follows:







Or please refer to https://github.com/Azure/azure-iot-sdk-c/blob/master/certs/certs.c

2. Communicate with Azure via X.509 Self-Signed Certificate

This document takes X.509 certificate as an example. Create customized X.509 certificates using a
third-party tool such as OpenSSL. This technique is ideal for test and development purposes.

2.1. Generate X.509 Self-Signed CA Certificate

The X.509 self-signed CA certificate can be generated with OpenSSL

#Generate ca certificate private key (pem file)

openssl genrsa -out mycakey.pem 2048

#Generate the ca certificate sign application file (csr file)

openssl req -new -key mycakey.pem -out myca.csr -subj "/C=CN/ST=myprovince/L=mycity/O=myorganization/OU=mygroup/CN=myCA"

#Self-signed ca certificate

openssl x509 -req -days 365 -sha1 -extensions v3_ca -signkey mycakey.pem -in myca.csr -out mycacert.pem

```
[app@S886048 /app/quectel/azure/x509self]$ openssl genrsa -out mycakey.pem 2048
Generating RSA private key, 2048 bit long modulus
.....................+++
.....+++
e is 65537 (0x10001)
[app@S886048 /app/quectel/azure/x509self]$ openssl req -new -key mycakey.pem -out myca.csr -subj "/C=CN/ST=myprovince/L=mycity/O=myorganization/OU=mygroup/CN=myCA"
[app@S886048 /app/quectel/azure/x509self]$ ls -ltr
total 8
-rw-rw-r-- 1 app app 1675 Sep 20 11:26 mycakey.pem
-rw-rw-r-- 1 app app 1013 Sep 20 11:26 myca.csr
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$ openssl x509 -req -days 365 -sha1 -extensions v3_ca -signkey mycakey.pem -in myca.csr -out mycacert.pem
Signature ok
subject=/C=CN/ST=myprovince/L=mycity/O=myorganization/OU=mygroup/CN=myCA
Getting Private key
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$ ls -ltr
total 12
-rw-rw-r-- 1 app app 1675 Sep 20 11:26 mycakey.pem
-rw-rw-r-- 1 app app 1013 Sep 20 11:26 myca.csr
-rw-rw-r-- 1 app app 1216 Sep 20 11:26 mycacert.pem
[app@S886048 /app/quectel/azure/x509self]$
```

2.2. Add X.509 Self-Signed CA Certificate

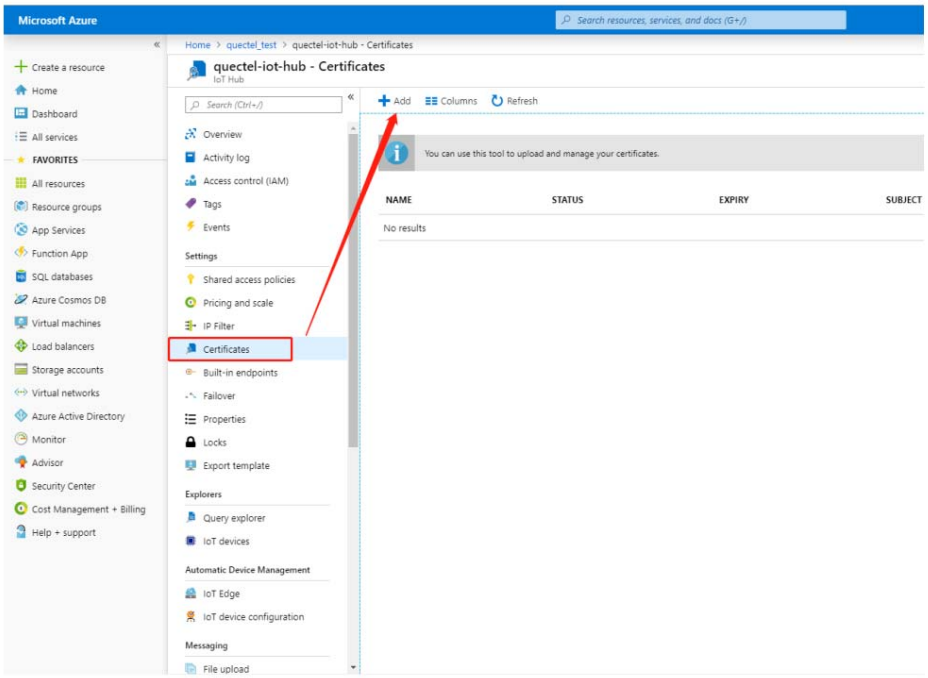
Follow the steps below to add the generated X.509 self-signed CA certificates into Azure IoT Hub.

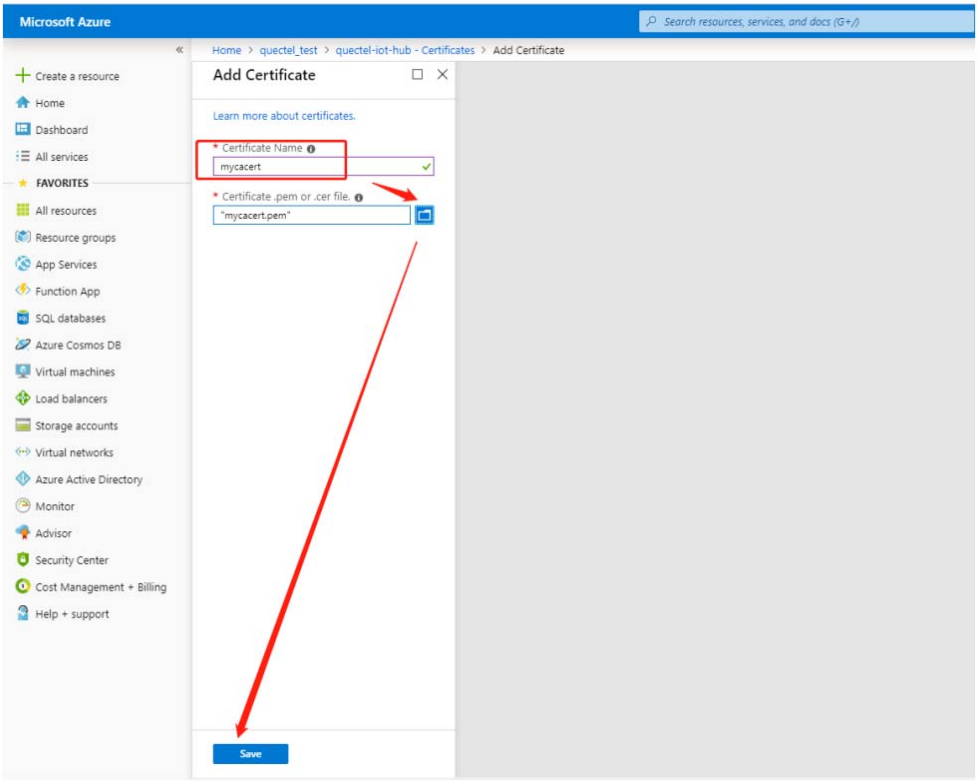
a) Click the name of "**quectel-iot-hub**" resource.

b) Click "**Certificates**", and then click "**Add**" to add a certificate.

c) Fill the certificate name (*mycacert* for instance) and then upload the certificate *mycacert.pem*.

d) Click "**Save**" to save the operations.



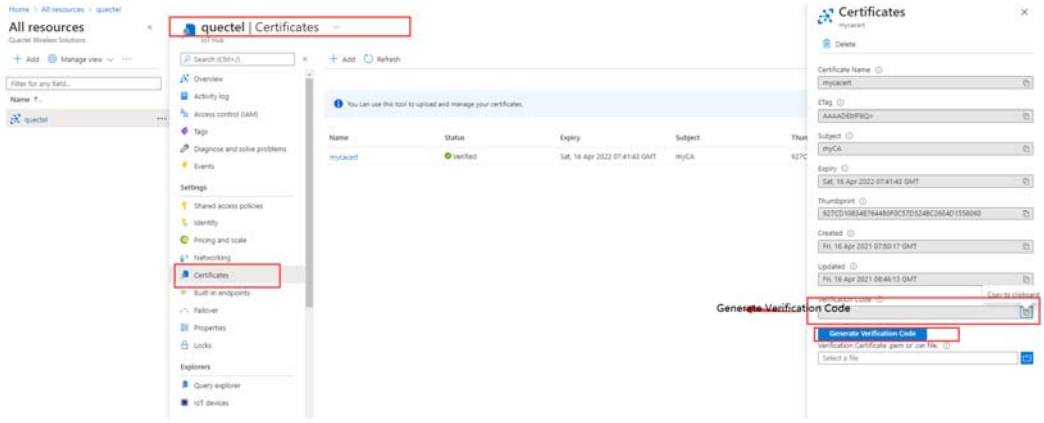


2.3. Verify X.509 Self-Signed CA Certificate

Follow the steps illustrated in the chapters below to verify the X.509 Self-signed CA certificate.

2.3.1. Generate Verification Code

a) Select "**mycacert**" certificate.
b) Click "**Generate Verification Code**".
c) After the "**Verification Code**" is generated successfully, click "**Copy**" button to copy the code to
clipboard.



2.3.2. Generate Verification Certificate

Here are the details about how to generate verification certificate.

#Generate verificationCert.csr

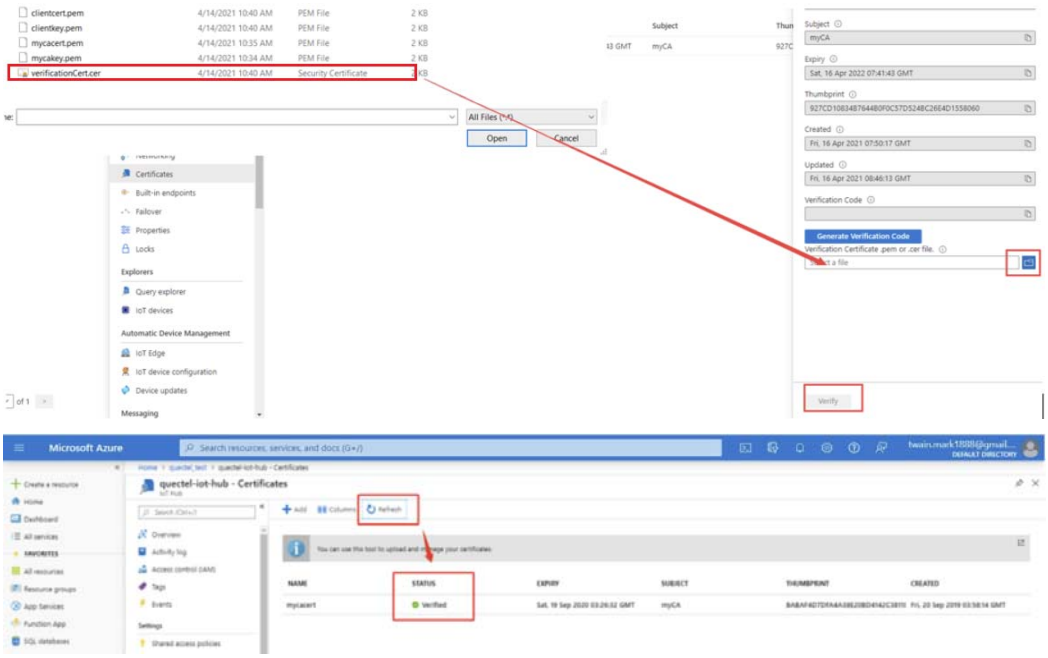
openssl req -new -newkey rsa:2048 -nodes -subj "/CN= (the Verification Code Gengrated) /" -keyout ./verificationCert.key -sha256 -days 365 -out ./verificationCert.csr

#Self-signed verificationCert

openssl x509 -req -in ./verificationCert.csr -CA ./mycacert.pem -CAkey ./mycakey.pem -CAcreateserial -out ./verificationCert.cer -days 365 -sha256

2.3.3. Verify Verification Certificate

a) Click "**Upload**" button to upload file *verificationCert.cer*.
b) Click "**Verify**" to verify the certificate.
c) Click "**Refresh**" to check whether the verification certificate has been **Verified**.



2.4. Generate X.509 Self-Signed Client Certificate

#Generate client certificate private key (pem file)

```
openssl genrsa -out clientkey.pem 2048
```

#Generate client certificate sign application file (csr file)

```
openssl req -new -key clientkey.pem -out client.csr -subj "/C=CN/ST=myprovince/L=mycity/O=myorganization/OU=mygroup/CN=myClient"
```

#Self-signed client certificate

```
openssl x509 -req -days 365 -sha1 -extensions v3_req -CA ./mycacert.pem -CAkey ./mycakey.pem -CAcreateserial -in client.csr -out clientcert.pem
```

#verify

```
openssl verify -CAfile ./mycacert.pem clientcert.pem
```

#convert PEM to CRT format

```
openssl x509 -outform der -in ./clientcert.pem -out ./clientcert.crt
```

2.5. Create X.509 Self-Signed Device

Please create X.509 self-signed device in Azure IoT Hub as follows:

a) Click "**IoT devices**" and then click "**New**".

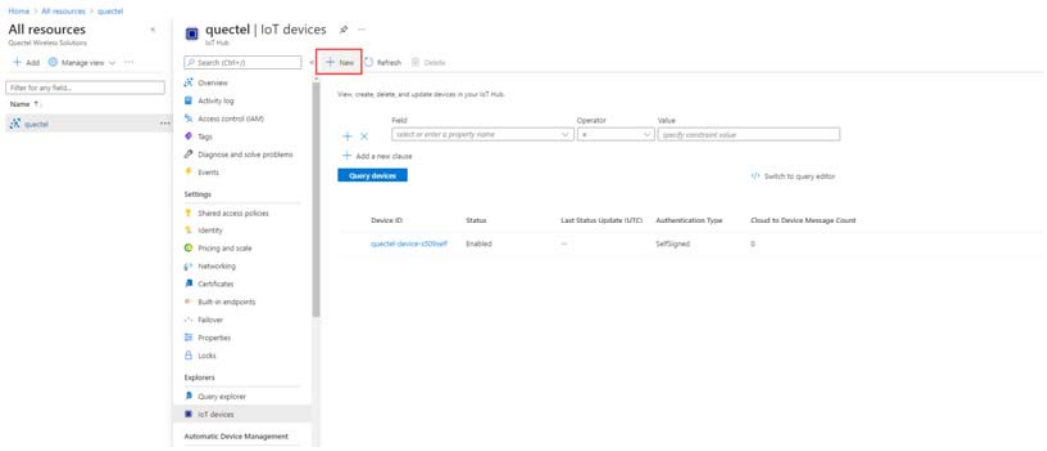
b) Select "**X.509 Self-Signed**".

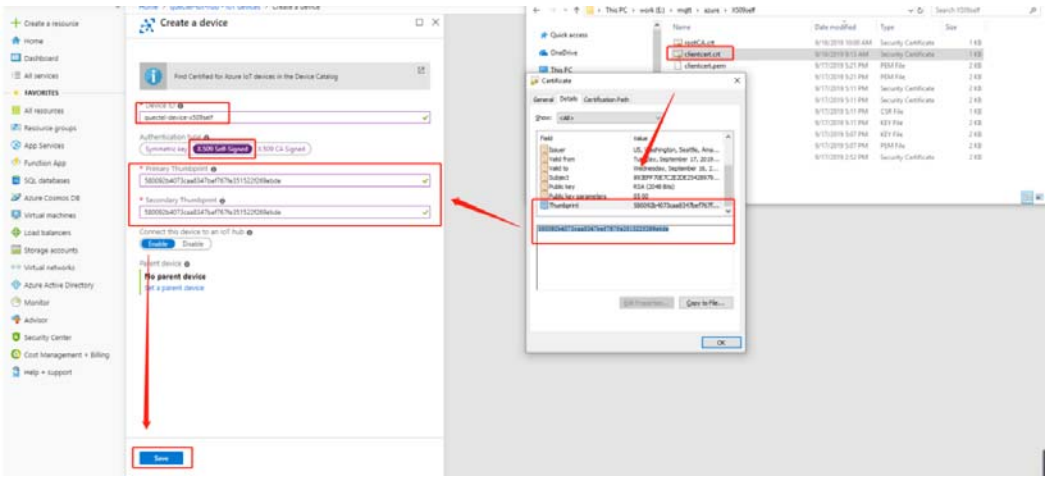
c) Open *clientcert.crt* to get "Thumbprint".

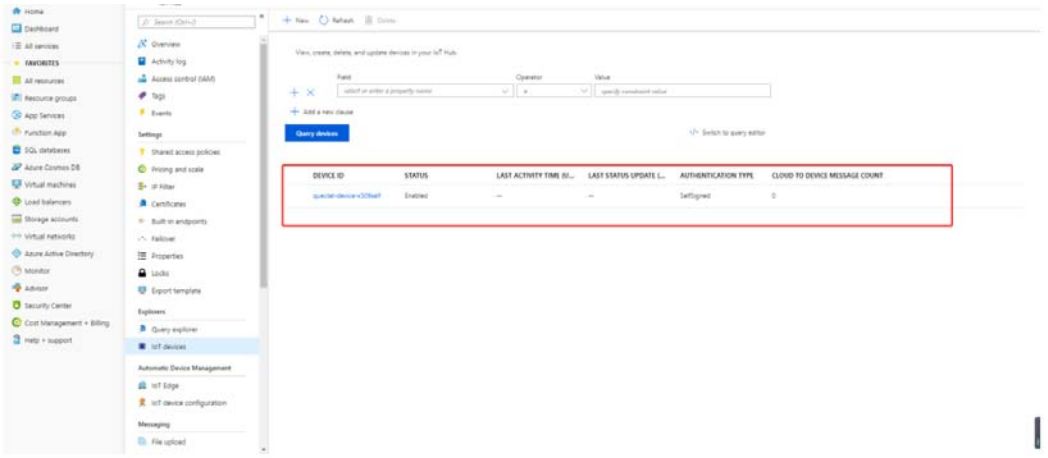
d) Input "Device ID", "Primary Thumbprint" and "Secondary Thumbprint".

e) Click "**Save**".

f) Check the creation result.







3. Usage of Device Explorer Tool

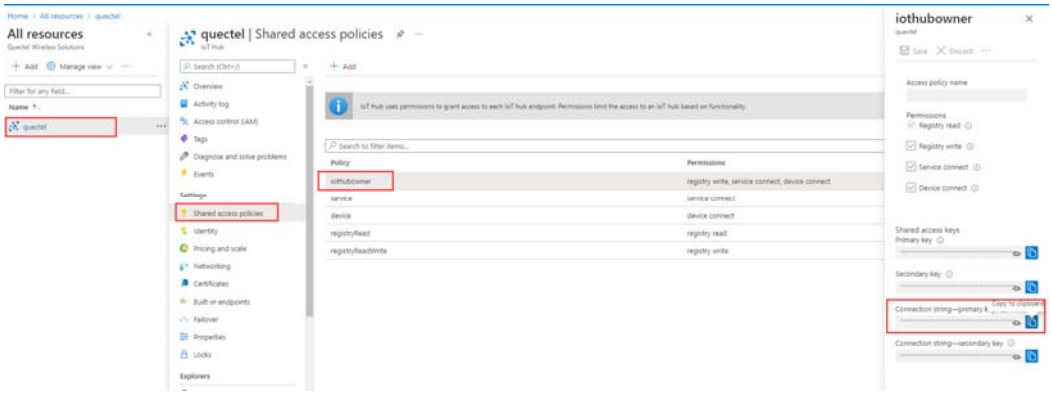
The Device Explorer tool can be used to manage devices connecting to customer's IoT hub, for example,
registering a device with customer's IoT hub, monitoring messages from the devices, and sending
messages to the devices.
This chapter describes the usage of the Device Explorer tool which will be used as Azure server tool.

3.1. Download/Install

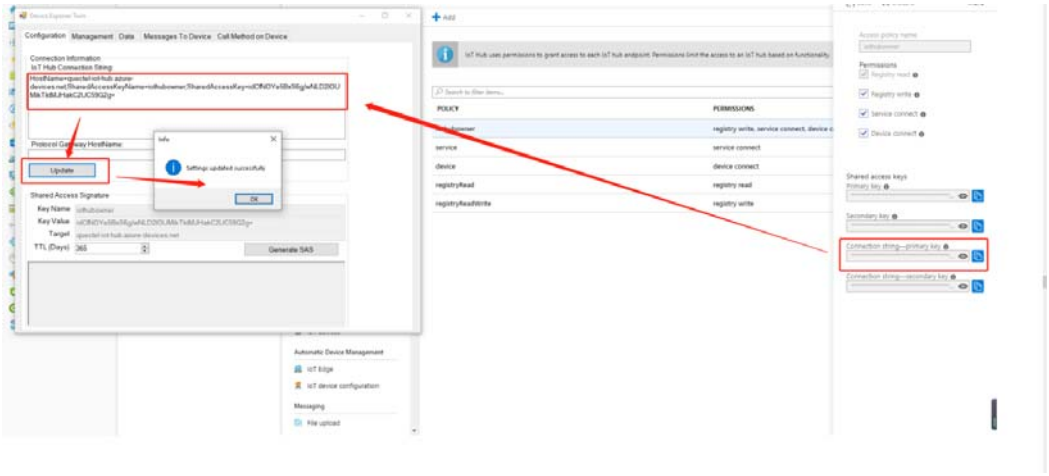
The Device Explorer tool can be downloaded from link: https://github.com/Azure/azure-iot-sdk-csharp/releases/download/2019-1-4/SetupDeviceExplorer.msi.
More details about downloading and installation can be reached from link: https://github.com/Azure/azure-iot-sdk-csharp/tree/master/tools/DeviceExplorer.

3.2. Configuration

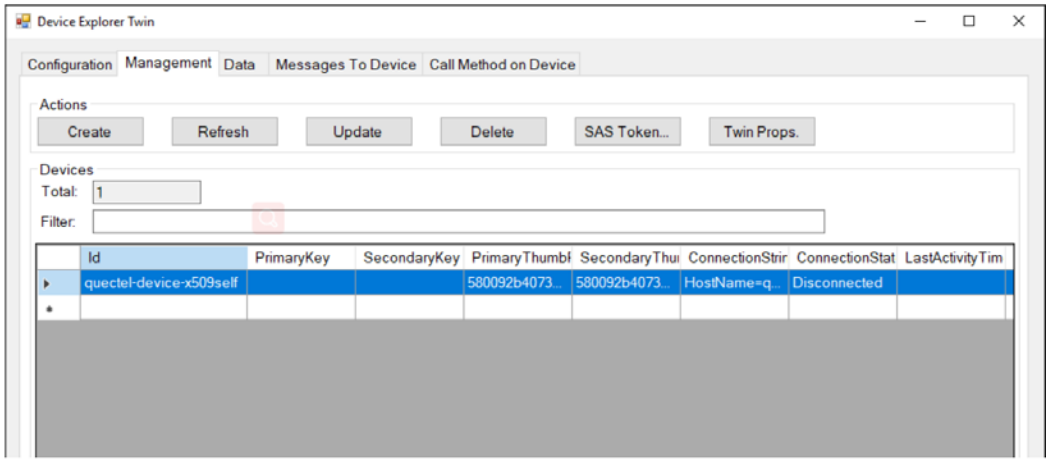
a) Get "Connection string—primary key" from Azure iothubowner, and copy the connection string to
clipboard.
b) Click the "**Configuration**" tab in the Device Explorer Twin window, and then paste the connection
string to the input box of "IoT Hub Connection String". After that, click "**Update**" → "**OK**" to finish the operation.



Device Explorer Configuration (Step a)



Device Explorer Configuration (Step b)



Use Device Explorer

4. Usage of MQTT.fx Tool

MQTT.fx is a MQTT Client written in Java based on Eclipse Paho. This chapter describes the usage of the
MQTT.fx tool which will be used as a MQTT client.

4.1. Download/Install

MQTT.fx tool can be downloaded from http://mqttfx.jensd.de/index.php/download.

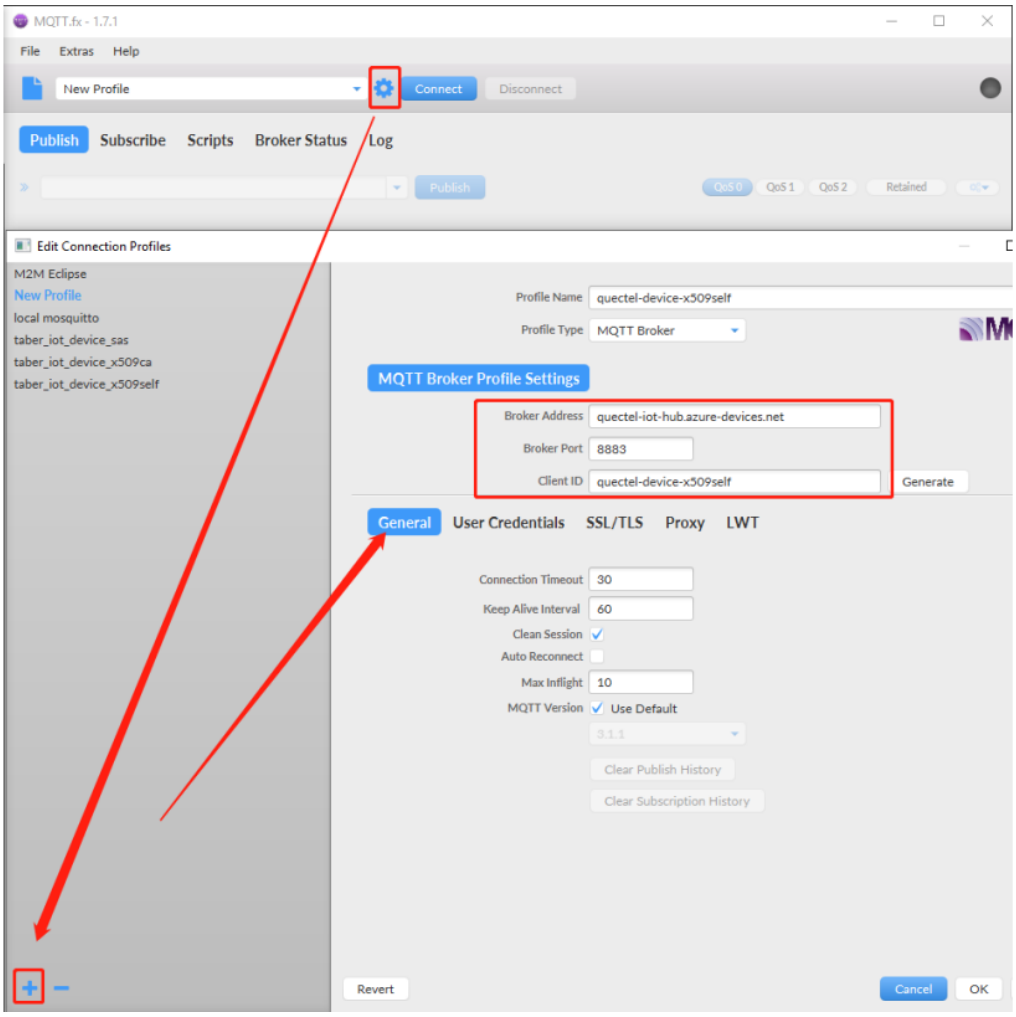
4.2. Edit Connection Profiles

- **MQTT Broker Profile Settings**

Broker Address: {customized hub name}.azure-devices.net
Broker Port: 8883
Client ID: {device_id}

- **General**

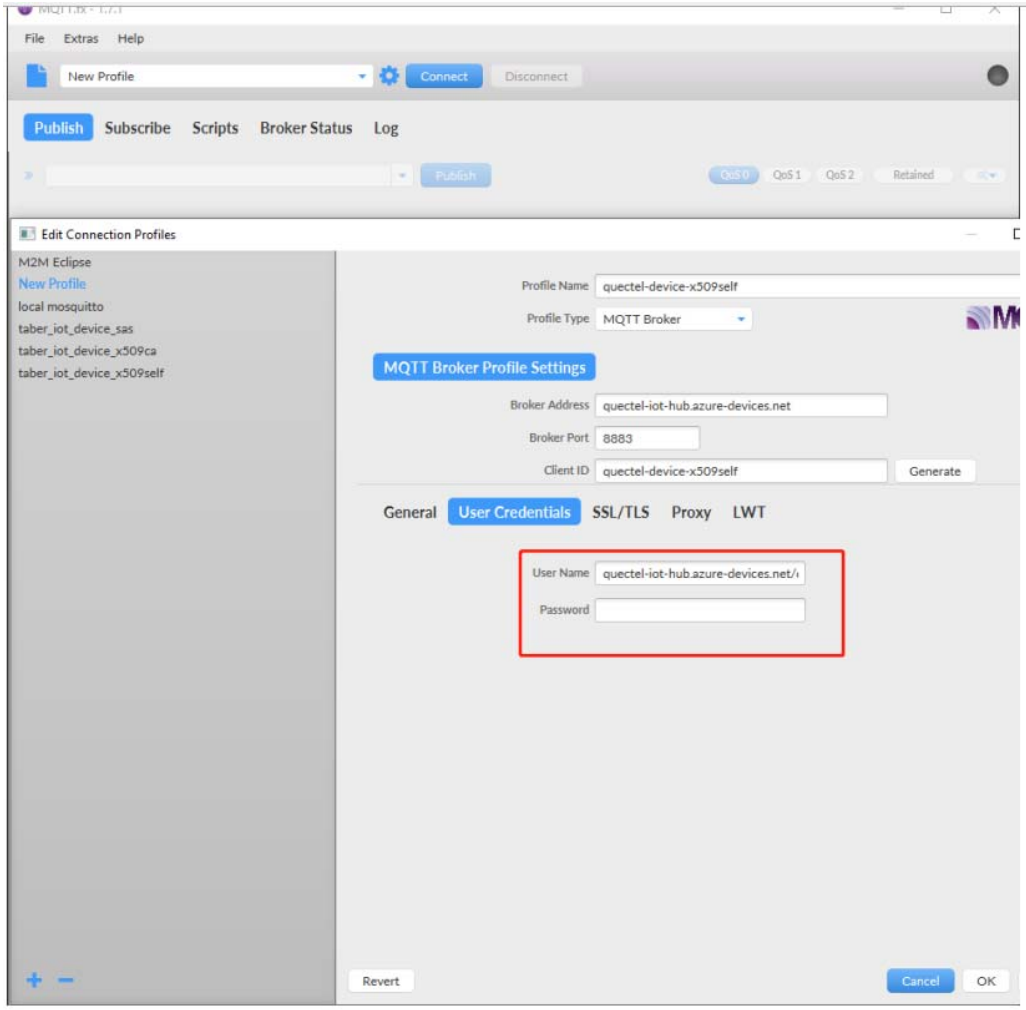
Use the default setting



User Credentials

User Name: *{customized hub name}.azure-devices.net/{device_id}/?api-version=2018-06-30*

Password: no password required.



User Credentials Configuration

SSL/TLS

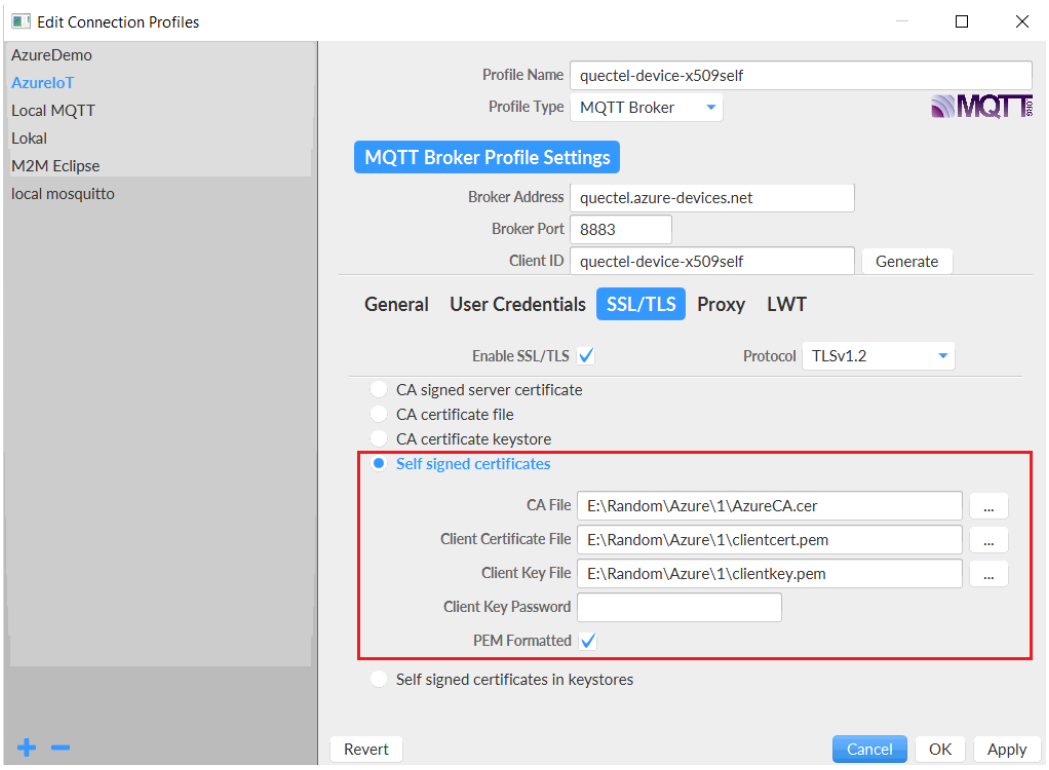
Select "**Enable SSL/TLS**" option

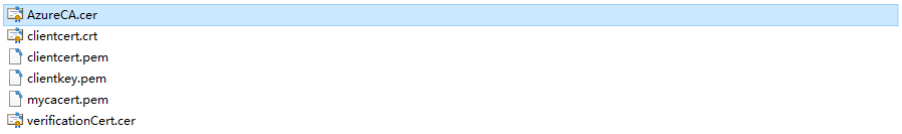
Check "**Self signed certificates**"

CA File: *AzureCA.cer*

Client Certificate File: *clientcert.pem*

Client Key File: *clientkey.pem*





4.3. Send Device-to-Cloud Messages

a) Click "**Data**" of Device Explorer tool, and then click "**Monitor**".

b) Click "**Connect**" of MQTT.fx tool and then input the topic *devices/{device_id}/messages/events/.*

Finally click "**Publish**".

5. Using Azure service on Module

5.1. AT Command Example

AT+QFLST command will list all the files in cert storage
AT+QFDEL="*" command will delete all files in cert storage
AT+QFUPL="UFS:AzureCA.cer",2818
AT+QFUPL="UFS:clientcert.cer",1258
AT+QFUPL="UFS:client.key",1702
AT+QSSLCFG="seclevel",1,2
AT+QSSLCFG="cacert",1,"UFS:AzureCA.cer"
AT+QSSLCFG="clientcert",1,"UFS:clientcert.cer"
AT+QSSLCFG="clientkey",1,"UFS:client.key"
AT+QSSLCFG="session_cache",1,0
AT+QMTCFG="ssl",0,1,1
AT+QSSLCFG="ignorelocaltime",1,1
AT+QICFG="tcp/keepalive",1,1,25,3
AT+QMTCFG="version",0,4
AT+QMTCFG="will",0,0,0,0
AT+QMTCFG="timeout",0,5,3,0
AT+QMTCFG="session",0,1
AT+QMTCFG="keepalive",0,60
AT+QMTCFG="recv/mode",0,1,1

AT+QMTOPEN=0,"mylotquectel.azure-devices.net",8883
AT+QMTCONN=0,"quectel-device-x509self","mylotquectel..azure-devices.net/quectel-device-x509self/?api-version=2018-06-30"