Connect to AWS Cloud Through MQTT with the MGate 5105 Industrial Protocol Gateway

Moxa Technical Support Team
support@moxa.com

Contents

 Released on July 15, 2021

About Moxa

Moxa is a leading provider of edge connectivity, industrial networking, and network infrastructure solutions for enabling connectivity for the Industrial Internet of Things. With over 30 years of industry experience, Moxa has connected more than 50 million devices worldwide and has a distribution and service network that reaches customers in more than 70 countries. Moxa delivers lasting business value by empowering industry with reliable networks and sincere service for industrial communications infrastructures. Information about Moxa's solutions is available at www.moxa.com.

How to Contact Moxa

Tel: +886-2-8919-1230
Fax: +886-2-8919-1231

MOXA®
Reliable Networks ▲ Sincere Service

1. Introduction

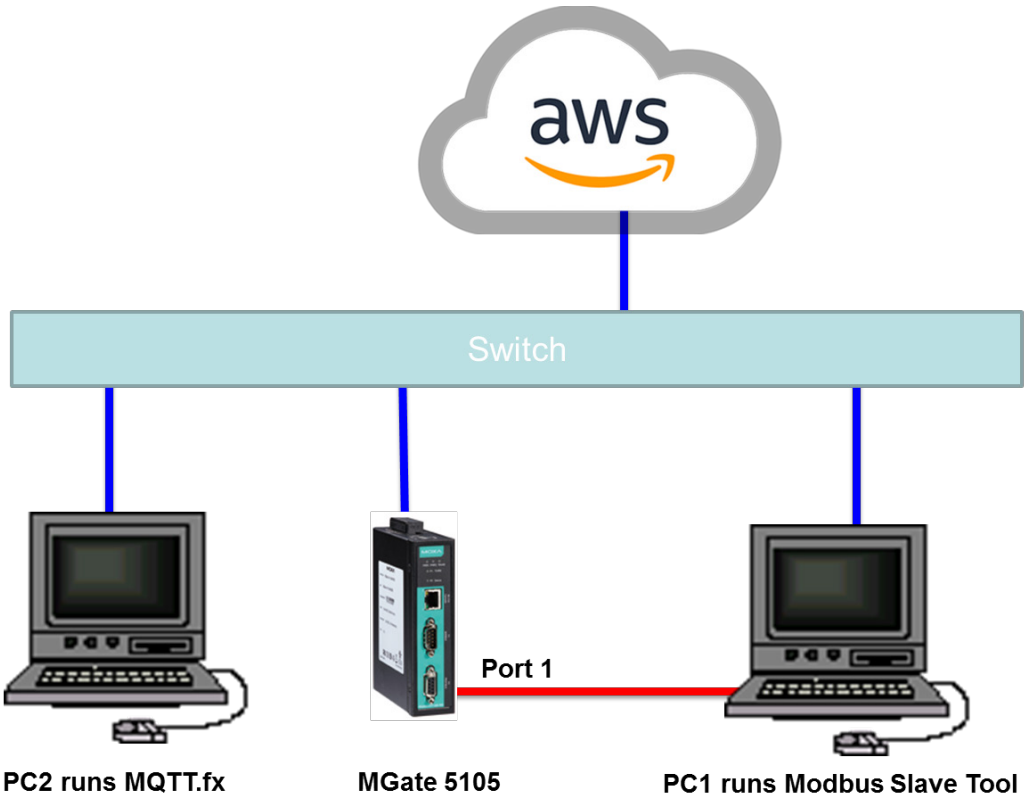
The MGate 5105 performs easy protocol conversions between Modbus RTU/ASCII, Modbus TCP, and EtherNet/IP protocols. From firmware version 4.0 and upwards, the publishing of the time stamps of fieldbus devices to cloud servers is supported. The cloud servers include Microsoft Azure, Alibaba Cloud, or MQTT Broker.

This document demonstrates how to connect the MGate 5105 to AWS IoT via Generic MQTT mode. We also demonstrate how to publish fieldbus data messages and subscribe to message from AWS IoT.

2. System Topology

Figure 1 illustrates the system topology. PC1 runs Modbus Slave tool to act as a Modbus RTU device. It connects to MGate 5105's Port 1. The MGate 5105 acts as a MQTT Client device and connects to AWS IoT. PC2 runs MQTT.fx, which is an MQTT Client. We use MQTT.fx to publish messages in AWS IoT and subscribe to MGate 5105's topic in it.

< Figure 1. System Topology >



3. Prerequisites

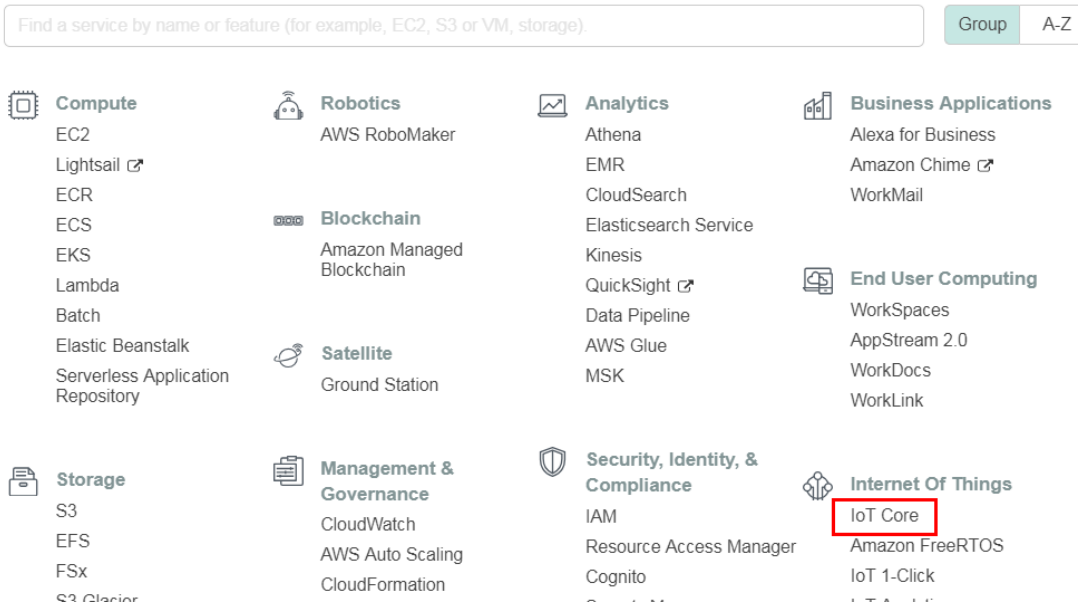
3.1 Modbus Slave Tool

Modbus Slave is a very popular Modbus slave simulator for testing and debugging of your modbus devices, which support Modbus RTU/ASCII and Modbus TCP/IP.

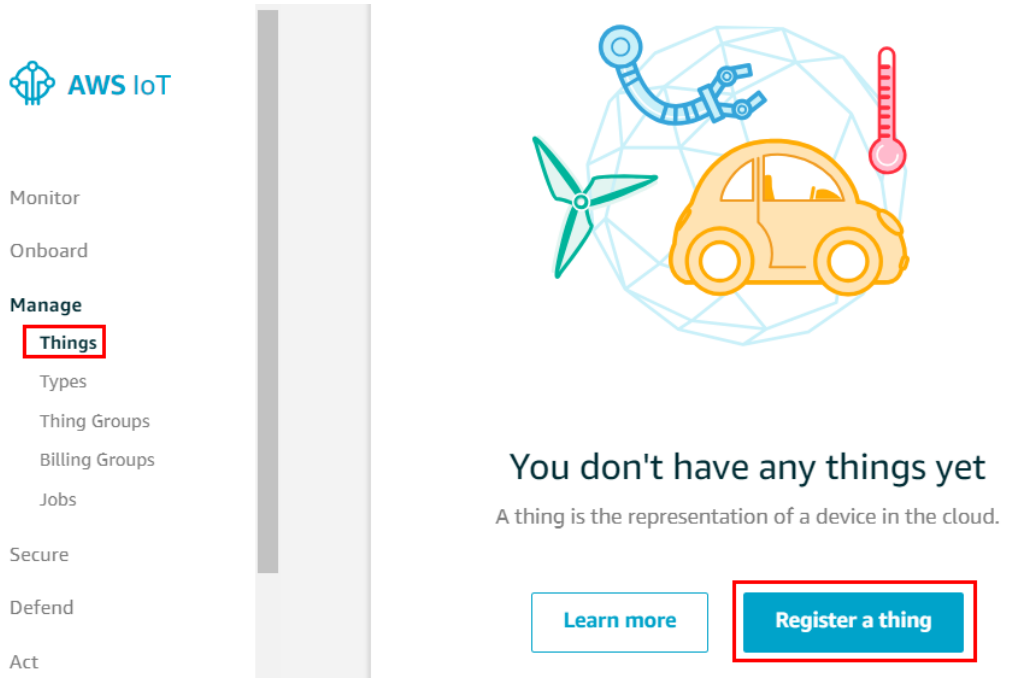
Download from website: http://www.modbustools.com/download.html

3.2 Create AWS IoT and Thing

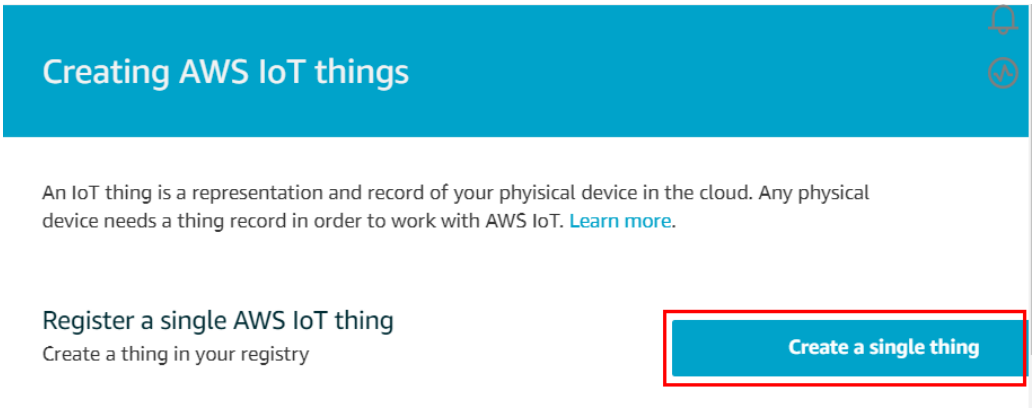
1. Use AWS user account to log in to AWS Console.
   Website: https://aws.amazon.com/console/
2. Find the Internet Of Things → IoT Core service:



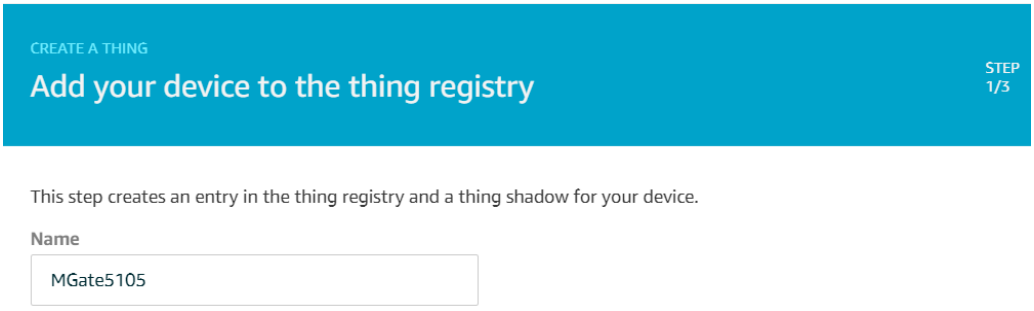
3. Create a Thing:
   a. Register a thing: **Manage** → **Things**.



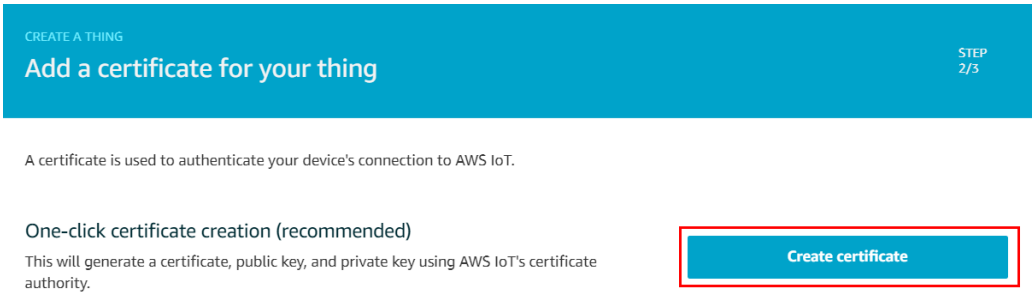
   b. Create a single thing:



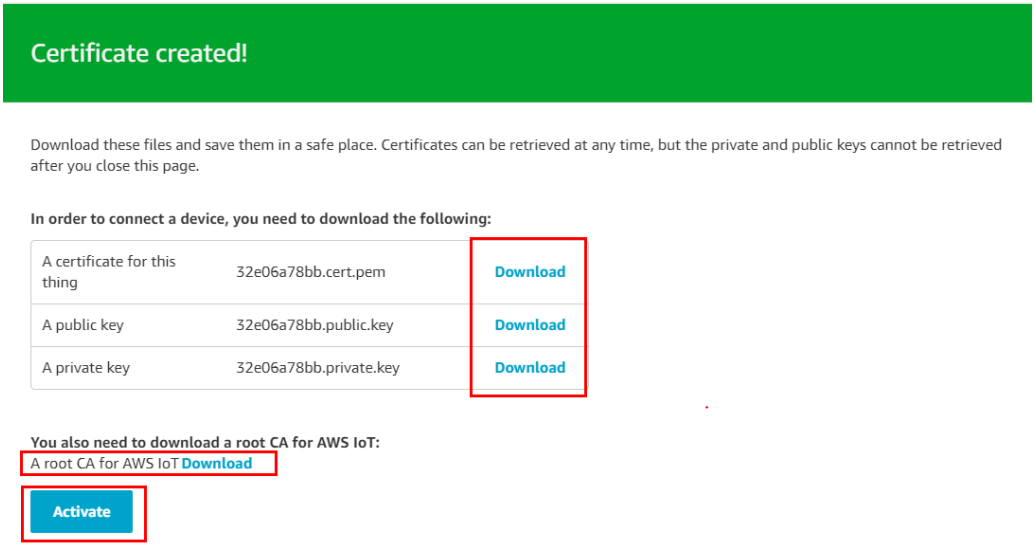
   c. Give a thing's name:



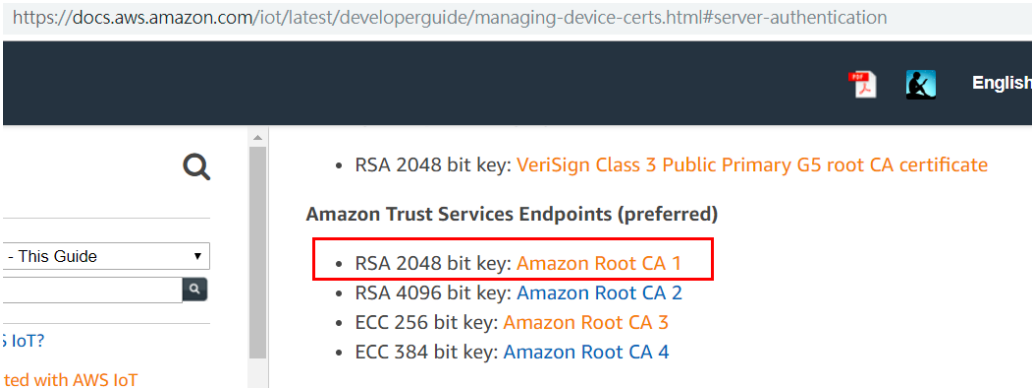
d. Execute **Create certificate**:



e. After creating a certificate, download the thing's certificate, private key, and root CA certificate. Then click **Activate**.

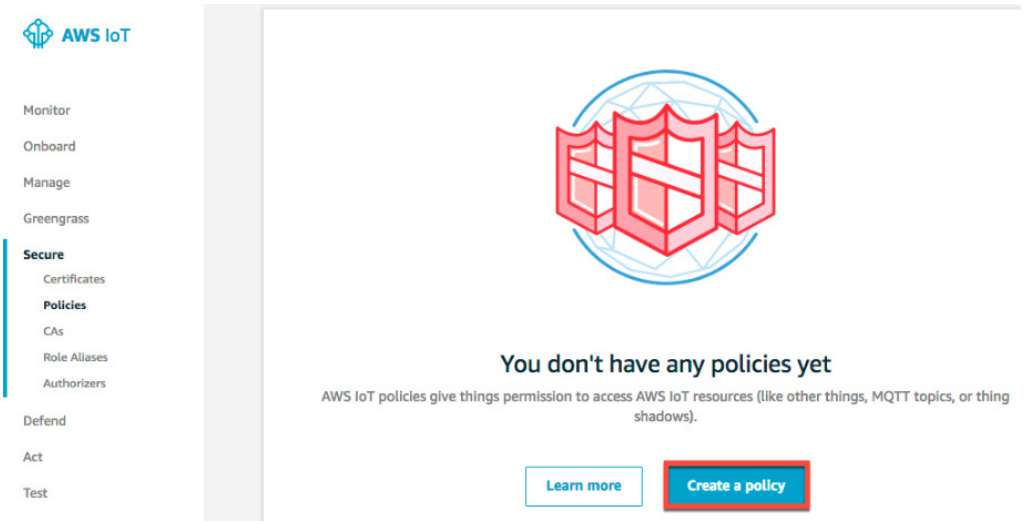


When you click **Download** of root CA, a new web page pops up as below:

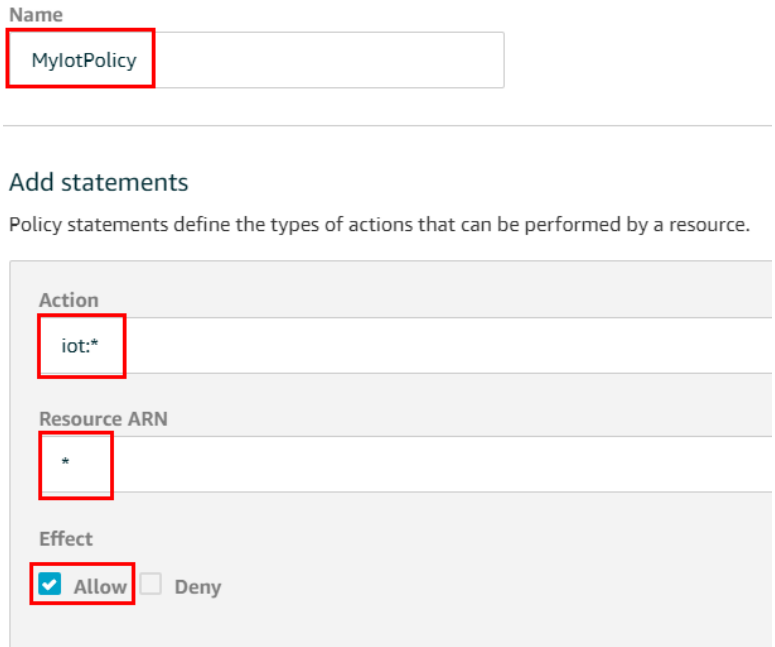


Select **Amazon Root CA 1**. A new web page that shows an Amazon Root CA certificate in PEM format will pop up. Save the content as a *.pem file.

4. Create a Policy
   a. In Secure → Policies, click **Create a policy**:

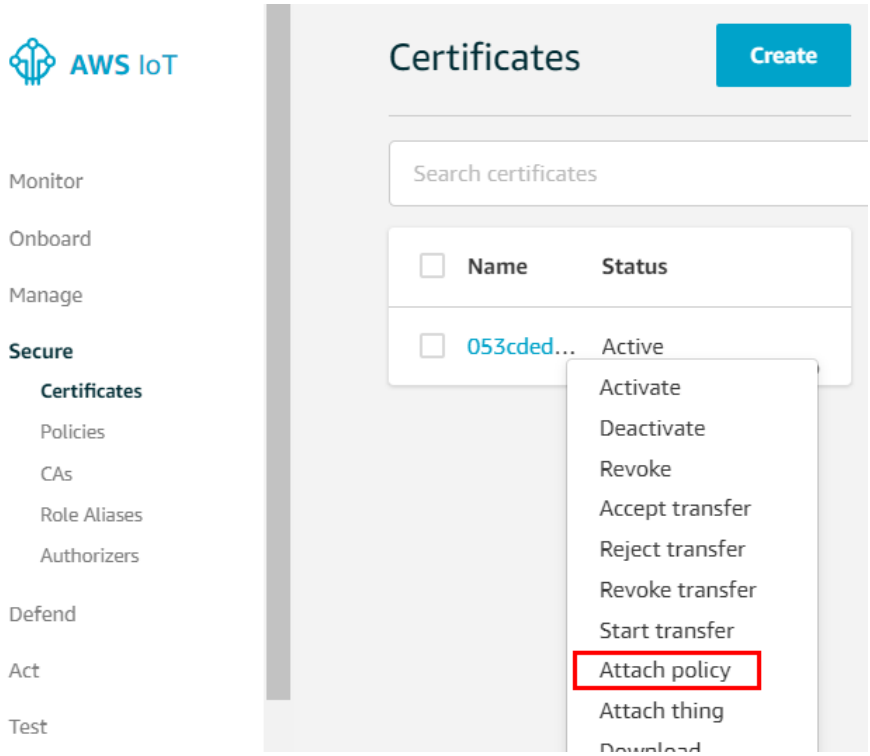


   b. In the **Action** field, you can type the actions you want to perdorm with the AWS IoT. For all actions, please type **iot:***. In the **Resource ARN** field, type *. Select the **Allow** check box. This allows all clients to connect to AWS IoT.

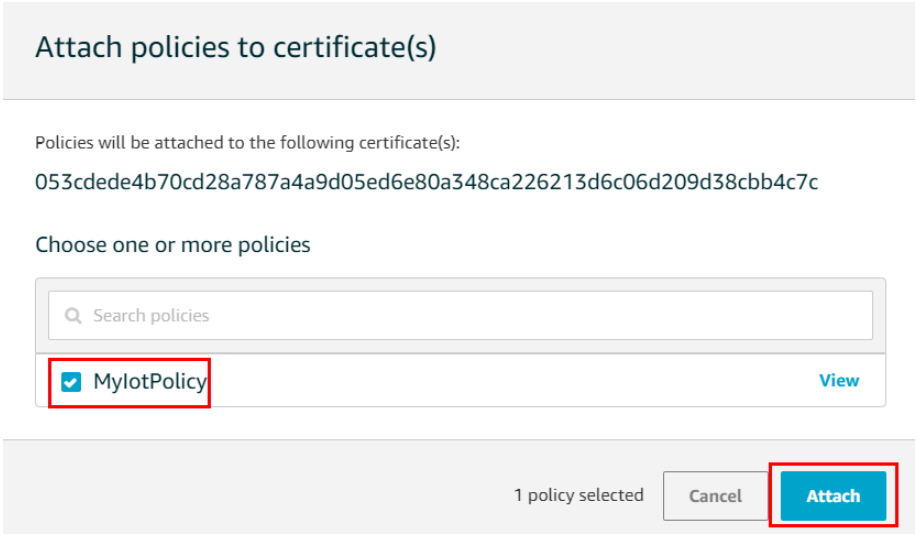


After you have entered the information for your policy, choose **Create**.

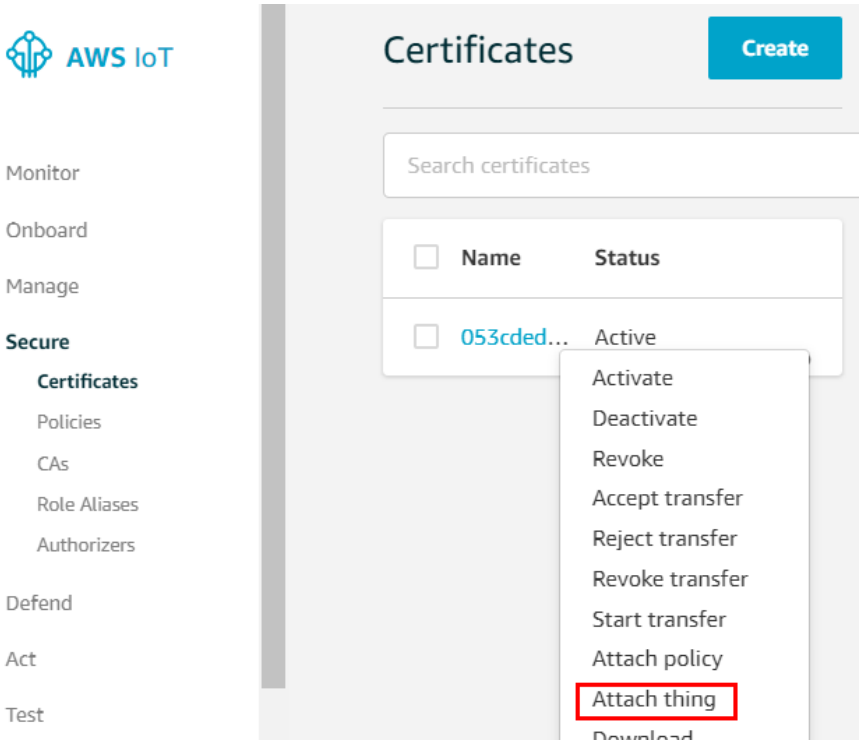
5. Attach an AWS IoT Policy to a Device Certificate
   a. Choose **Secure** → **Certificates**. In the box you created for the certificate, choose ... to open a drop-down menu, and then choose **Attach policy**.



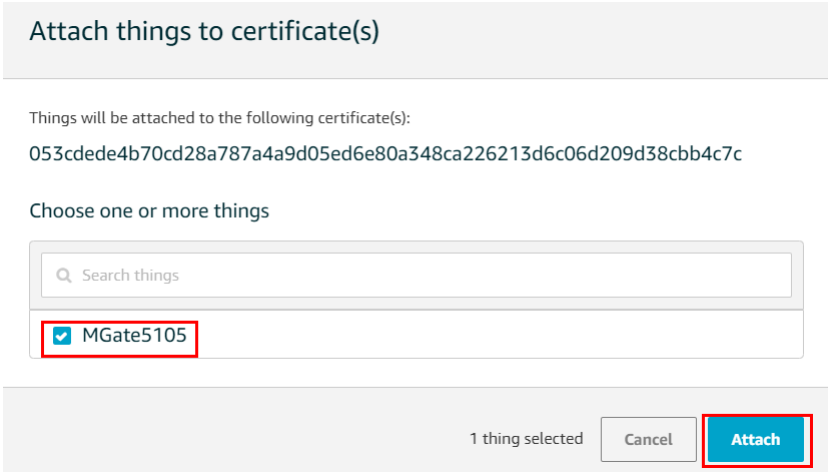
b. In the **Attach policies to certificate(s)** dialog box, select the check box next to the policy you created in the previous step, and then choose **Attach**.



6. Attach a Certificate to a Thing:
   A device must have a certificate, private key, and root CA certificate to authenticate with AWS IoT.
   a. In the box created for the certificate, choose ... to open a drop-down menu, and then choose **Attach thing**.



b. In the **Attach things to certificate(s)** dialog box, select the check box next to the thing you registered, and then choose **Attach**.



3.3 MQTT.fx Tool

MQTT.fx is a MQTT Client written in Java, based on Eclipse Paho. You can download thr latest version at the following website:https://mqttfx.jensd.de/index.php/download.

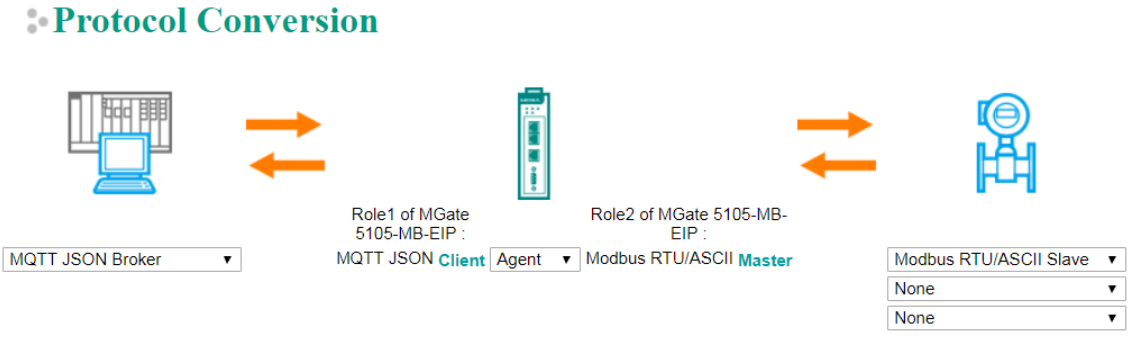
4. MGate 5105 Settings

Log in to MGate 5105's web console, then do the following settings:

4.1 Protocol Conversion

The MGate 5105 supports two kinds of MQTT data message formats: JSON and RAW. In this demonstration, we use the JSON format. In Protocol Conversion Settings, choose **MQTT JSON Client** as Role1. In the fieldbus site, choose the following protocols: Modbus RTU/ASCII Slave, Modbus TCP Server, or EtherNet/IP Adapter. Note that multiple combinations are allowed for settings in Role2. In this demonstration, we choose Modbus RTU/ASCII Slave.

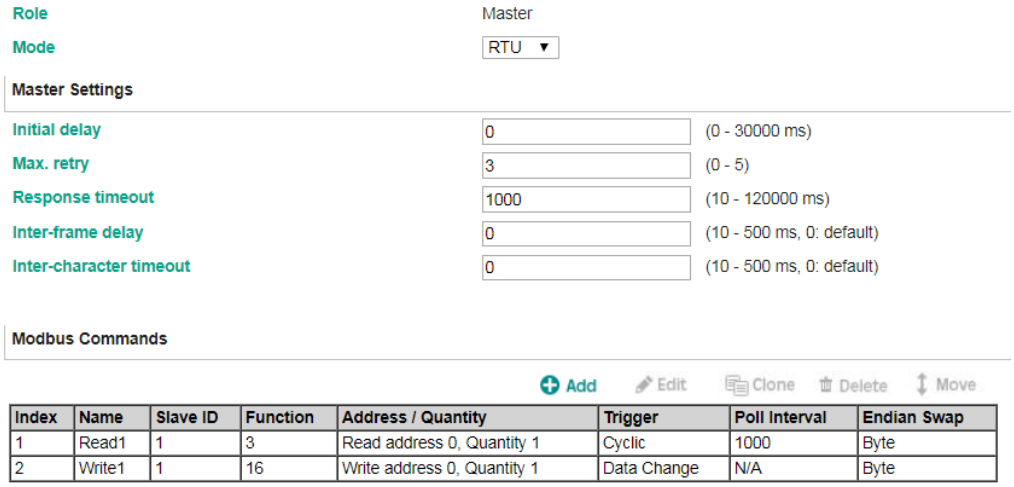
Set as below:



4.2 Modbus RTU Master Settings

1. In the **Modbus RTU/ASCII Master** Settings web page, we choose **RTU** for Mode and keep **Master Settings** as the default setting.
2. Add a **Read1** Modbus command to send a function code 03 and a command for quantity as 1, and Endian Swap as Byte. Poll interval is 1000 ms.
3. Add a **Write1** Modbus command to send a function code 06 command, and Endian Swap as Byte. Its **Trigger** command is **Data Change**.

Set as below:



Index	Name	Slave ID	Function	Address / Quantity	Trigger	Poll Interval	Endian Swap
1	Read1	1	3	Read address 0, Quantity 1	Cyclic	1000	Byte
2	Write1	1	16	Write address 0, Quantity 1	Data Change	N/A	Byte

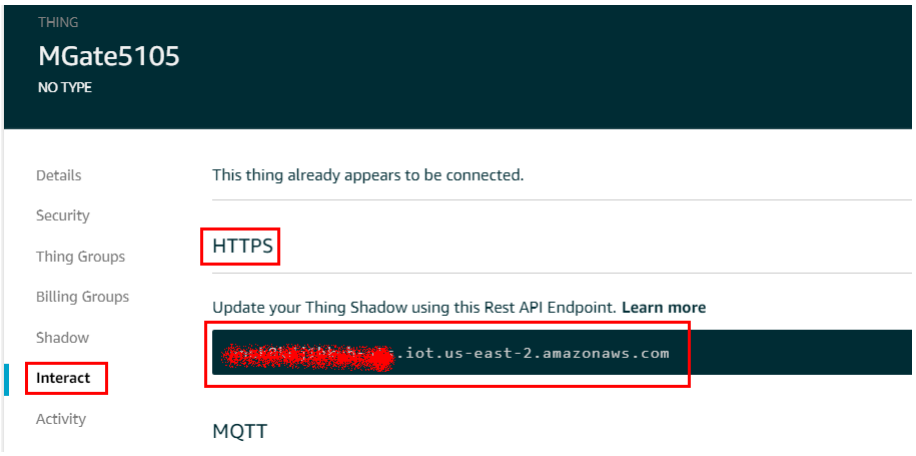
4.3 MQTT JSON Client Settings

1. Basic Settings:

   In **Basic Settings** → **Remote MQTT broker** string, fill in your MQTT Broker IP address or hostname and broker's listen port.

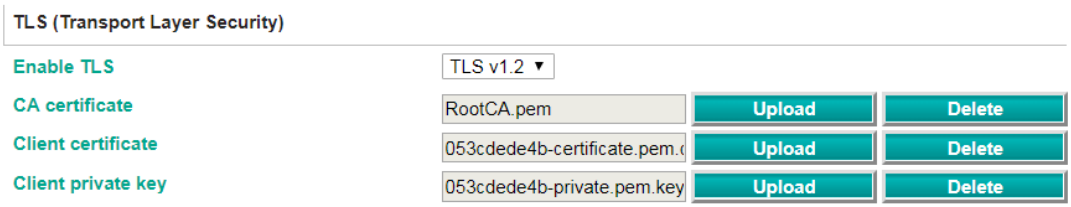
   Find the broker address for your device (thing) by selecting your device/thing in the AWS IoT console, and then click on **Interact** menu.

   (Things > THING_NAME > Interact)

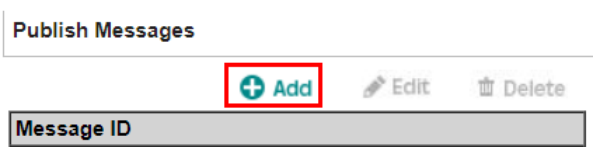


The Rest API endpoint name under HTTPS section is your broker address.
The port number for the secured MQTT connection is "8883".
Client ID setting is an identity of MQTT session. It must be unique. The broker does not accept the same Client ID connection for a second time. You can fill in an identifiable ID or click the **Generate** button to generate a random ID.
The broker may need the client to provide an username and password to authenticate the client connection. If you need to, fill in the correct username and password.
Set as below:

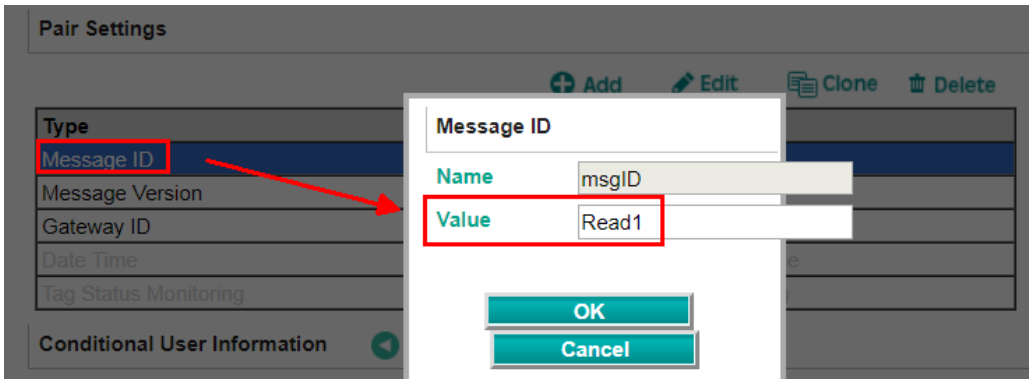
2. TLS (Transport Layer Security) Setting:
The MGate 5105 supports TLS to secure communications between MQTT Broker and Client. Here, we use version 1.2.
To enable a TLS transmission, upload the CA certificate, client certificate, and client keyfile. The certificates and keyfile must be PEM encoded.
Set as below:



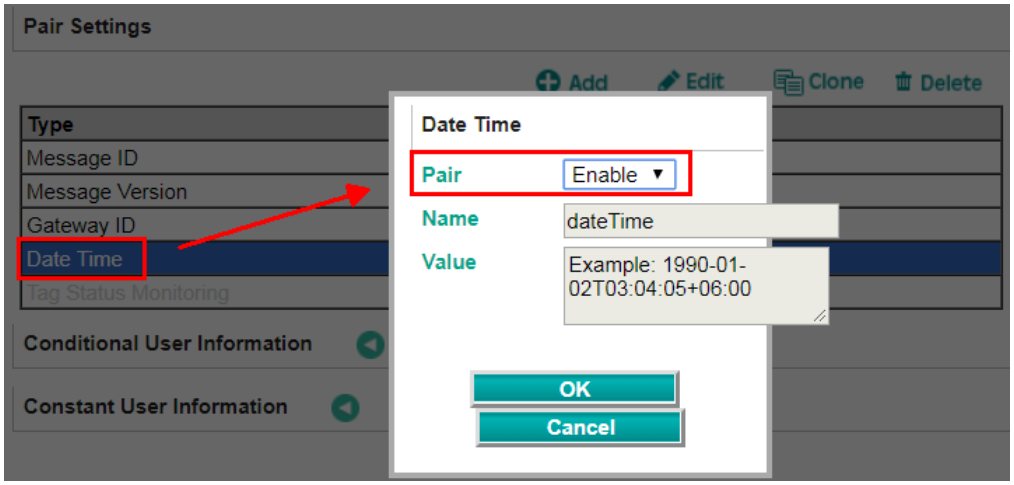
3. Publish Messages:
Click the **Add** button to create a **Publish Message** and click it to edit the message settings.



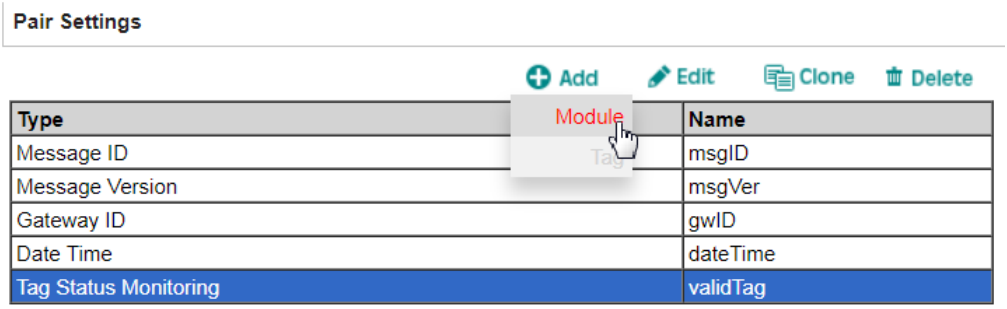
In **Pair Settings**, click **Message ID** to edit **Name**, and set **Value** as **Read1**.



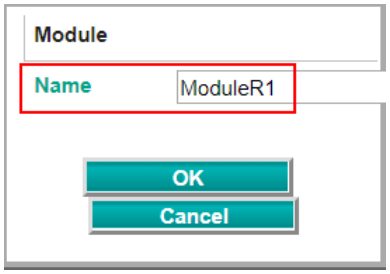
Click **Date Time** to enable **dateTime** padding in the message.



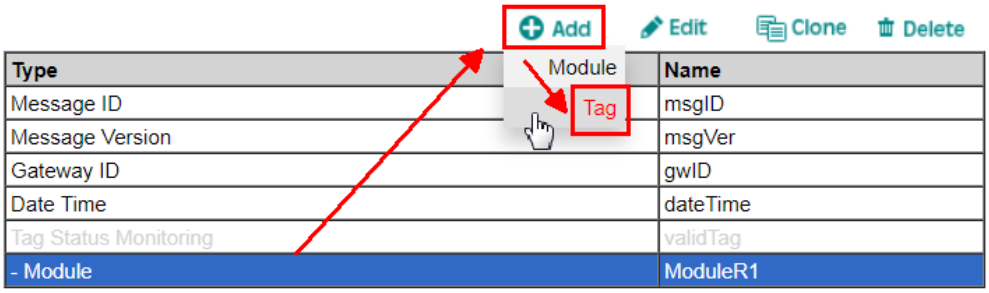
Click **Add** → **Module** to create a new module.



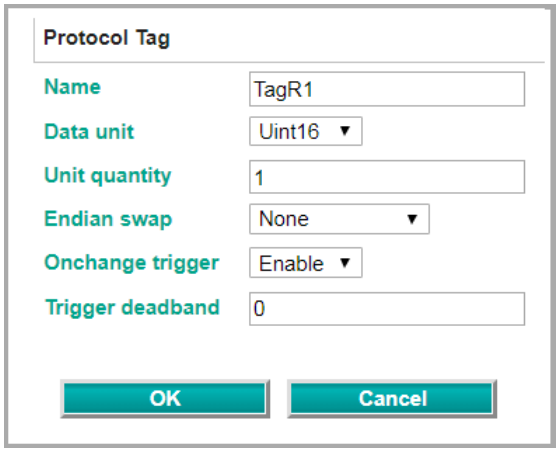
Set **Name** as **ModuleR1**.



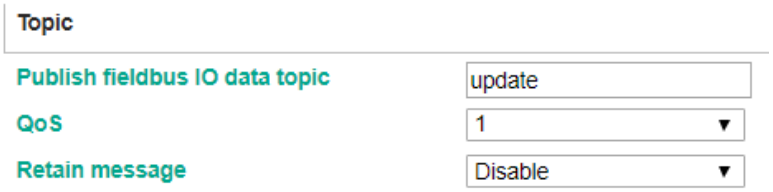
Choose ModuleR1 and then click **Add** → **Tag**.



Create a Protocol Tag as below:

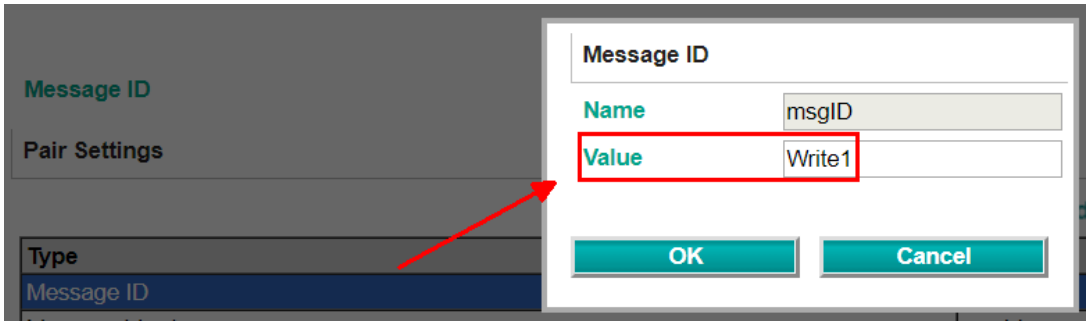


We set the topic name of this message as **update**. Since AWS doesn't support **QoS 2** and **Retain message** functions, we select the parameters of **QoS 1** and **Disable** mode under **Retain message**.

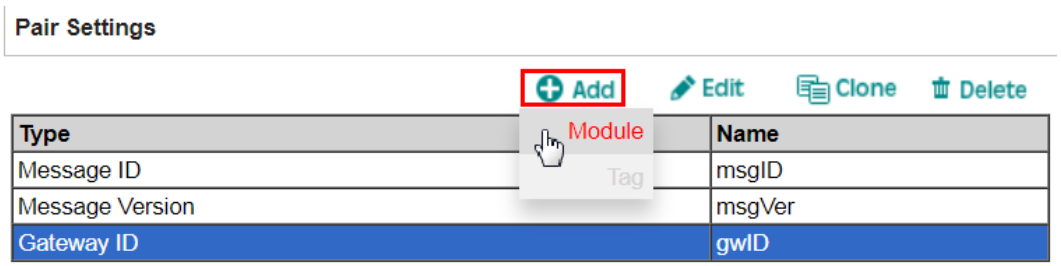


4. Subscribe Messages:
   Click the **Add** button to create a subscribe message and click it to edit message settings.

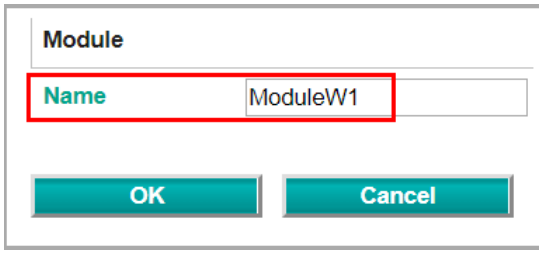
In **Pair Settings**, click **Message ID** to edit **Name** and set **Value** as **Write1**.



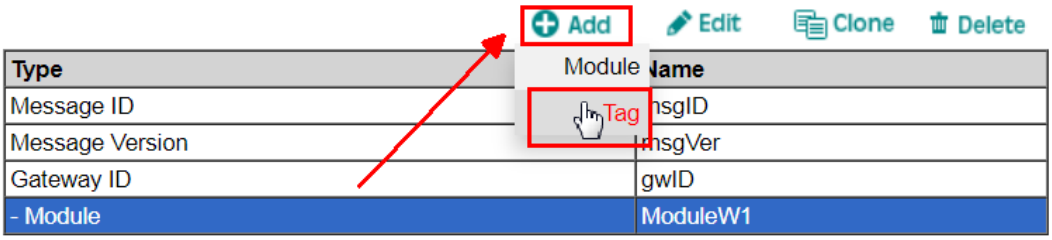
Click **Add** → **Module** to create a new module.



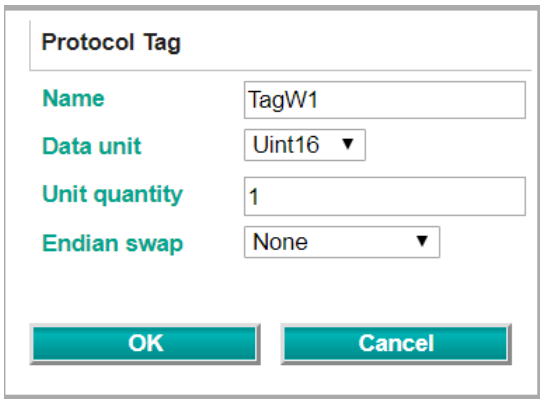
Set **Name** as **ModuleW1**.



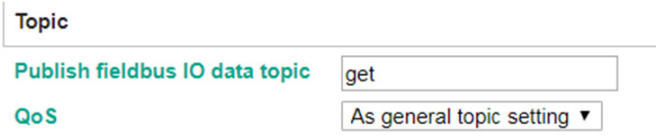
Choose **ModuleW1**, then click **Add** → **Tag**.



Create a Protocol Tag as below:

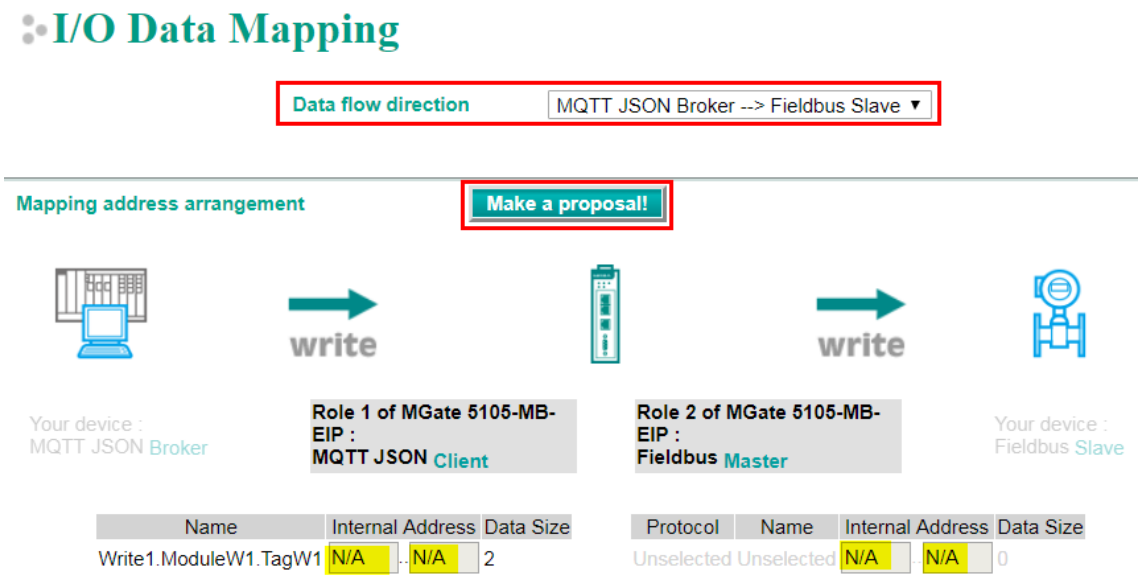


We set the topic name of this message as **get**.

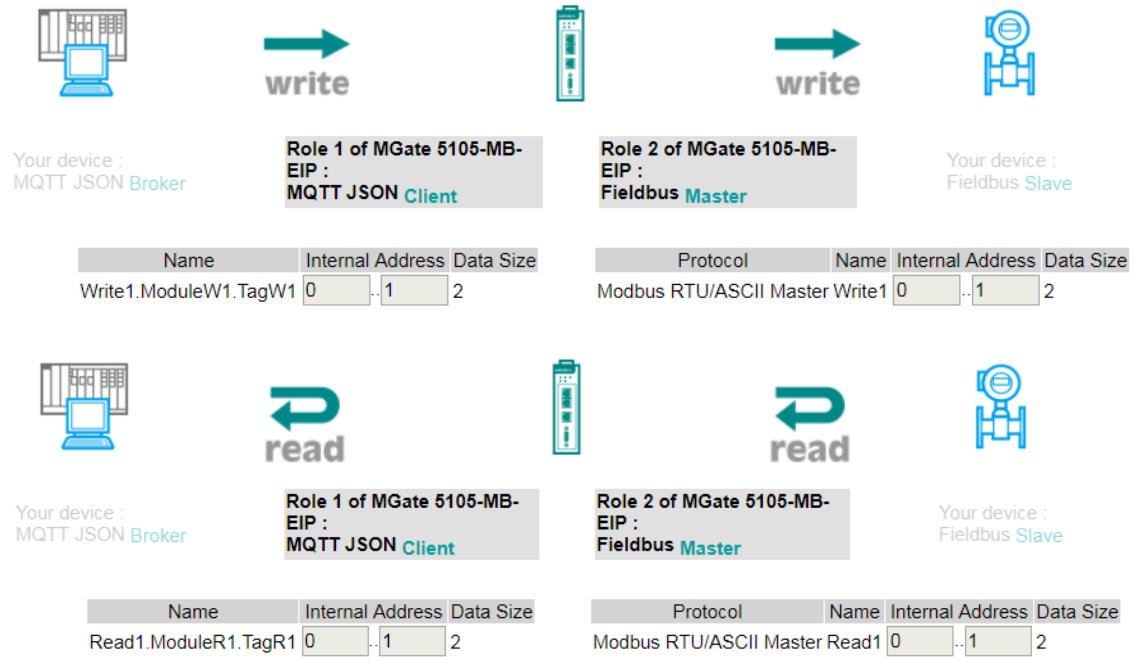


4.4 I/O Data Mapping

When the protocol settings are done, only one more step of I/O Data mapping for protocol configuration is required. Click the **Make a proposal** button for auto mapping in both **MQTT JSON Broker → Fieldbus Slave** direction and **Fieldbus Slave → MQTT JSON Broke**r direction.



The mapping result is as below:



4.5 Serial Settings

Serial Port1 connects to Modbus RTU device, so you must set the serial parameters of Port1.

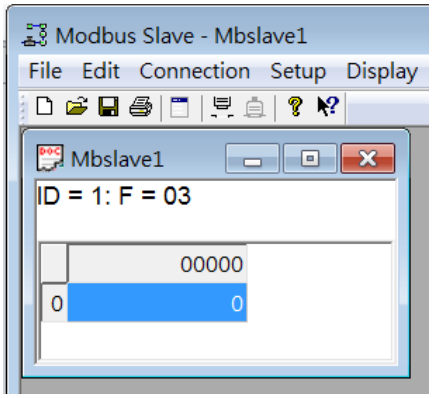
Set as below:

Serial Settings

Port	Baud rate	Parity	Data bit	Stop bit	Flow control	FIFO	Interface	RTS on delay	RTS off delay
1	115200	Even	8	1	None	Enable	RS-232	0	0

5. Modbus Slave Tool Settings

PC1 runs **Modbus Slave tool** and connects to MGate 5105's Serial Port. Add the Modbus definition below:



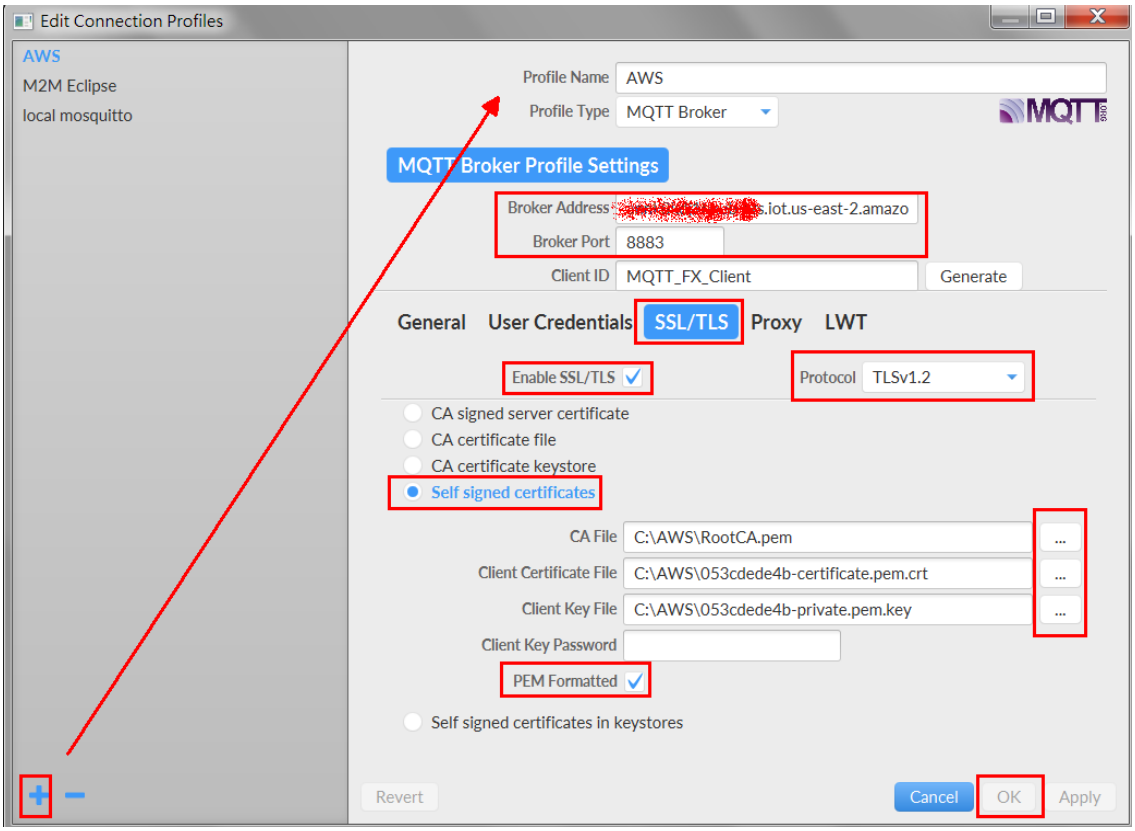
6. MQTT.fx Settings

6.1 Connection Settings

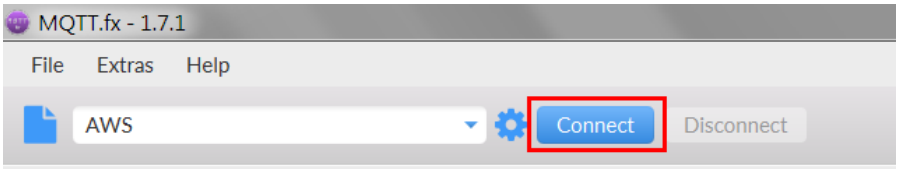
PC2 runs the MQTT.fx client. Create a new connection profile by going to the following menu option : **Extras** > **Edit Connection Profiles** and then click the **+** button (bottom left).

Input the correct Broker Address and Broker Port as 8883. In the **SSL/TLS** tab, enable **SSL/TLS** and choose **TLSv1.2** for **Protocol**.

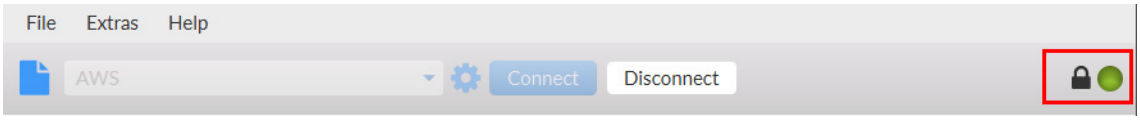
Select **Self signed certificates in keystores**, then upload **Root CA certificate**, **Client Certificate**, and **Client Key**. Enable **PEM Formatted**. Then press **OK** or **Apply** to finish the connection settings.



Select your AWS connection profile then click **Connect**:

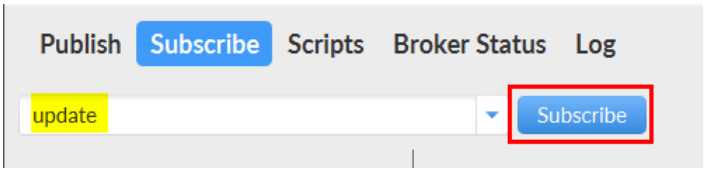


When connected, the connection status signal turns green.



6.2 Subscribe Topic

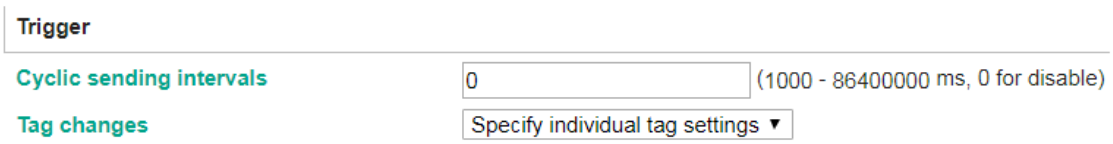
In the Subscribe tab, input **update**, which is the MGate 5105's publish topic name. Then click **Subscribe**.



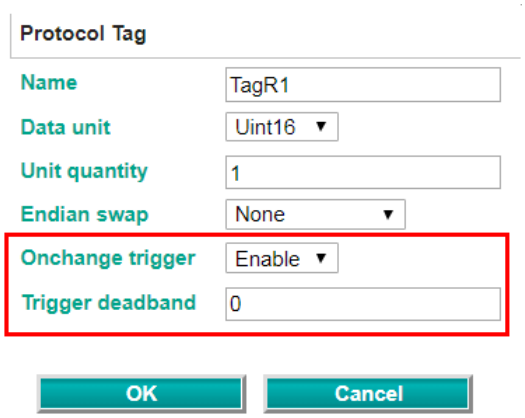
7. Communication Test

7.1 Publish message

We set **Trigger** as follows: For Cyclic sending interval, choose **0**; for tag changes, choose **Specify individual tag settings**:

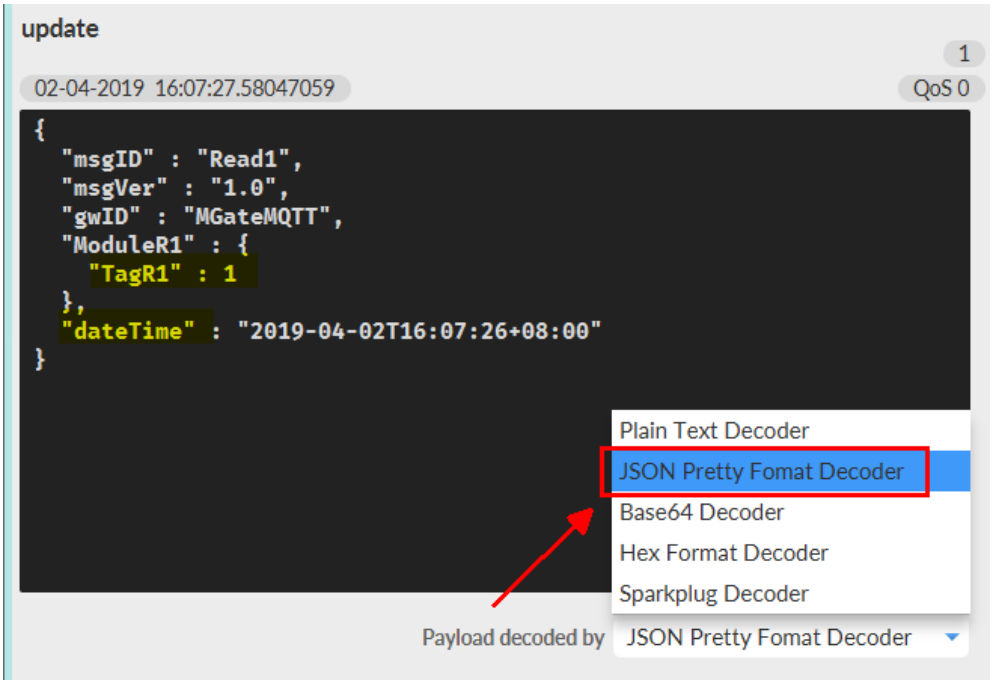


We set TagR1 **Onchange trigger** as enable with **Trigger deadband** as 0.



So when the MGate 5105 gets Modbus RTU device Register0's value changed, it triggers to publish message to AWS IoT Thing.

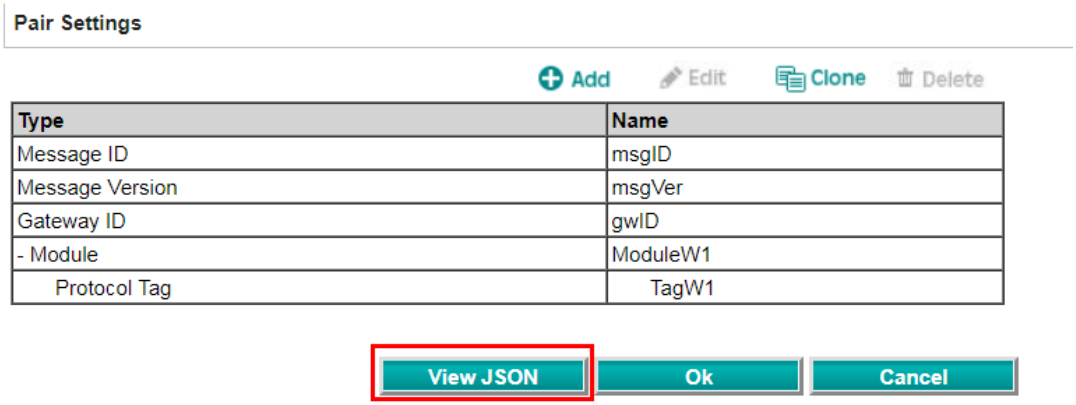
Now, update Modbus Register0's value as 1. In MQTT.fx tool, TagR1's value is shown as 1 and with dateTime padding.



7.2 Subscribe message

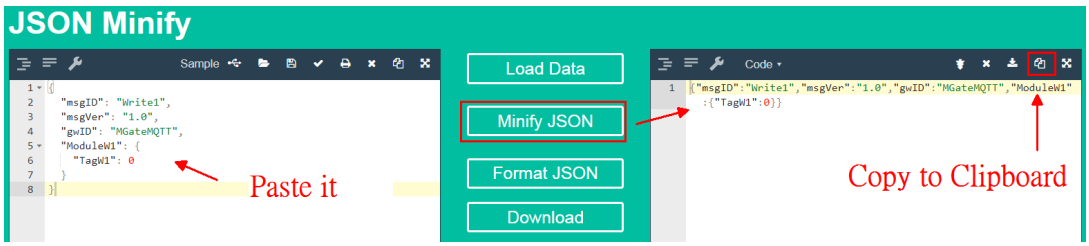
We use MQTT.fx to send messages to the device. You can follow the following steps:

1. Click **View JSON** button.

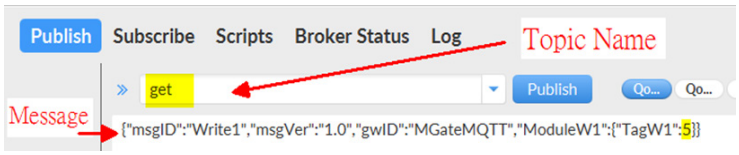


Copy Subscribe message JSON format:

2. The copied message has a lot of space and line feed. Use tool to compact it. Download a free online tool: https://jsonformatter.org/json-minify
   Paste the message on the left side, then click **Minify JSON**. It will show a compact JSON format message on the right side. Click **Copy to Clipboard**.



3. In the **MQTT.fx's Publish** tab, input **get** as Topic Name. Paste the message and change the **TagW1** value to 5 as below:



   Click **Publish** to send out message.

4. Check on Modbus Slave tool; Register0's value is updated as 5.