Connect to Mosquitto MQTT with the MGate 5105 Industrial Protocol Gateway

Moxa Technical Support Team
support@moxa.com

Contents

 Released on March 30, 2019

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

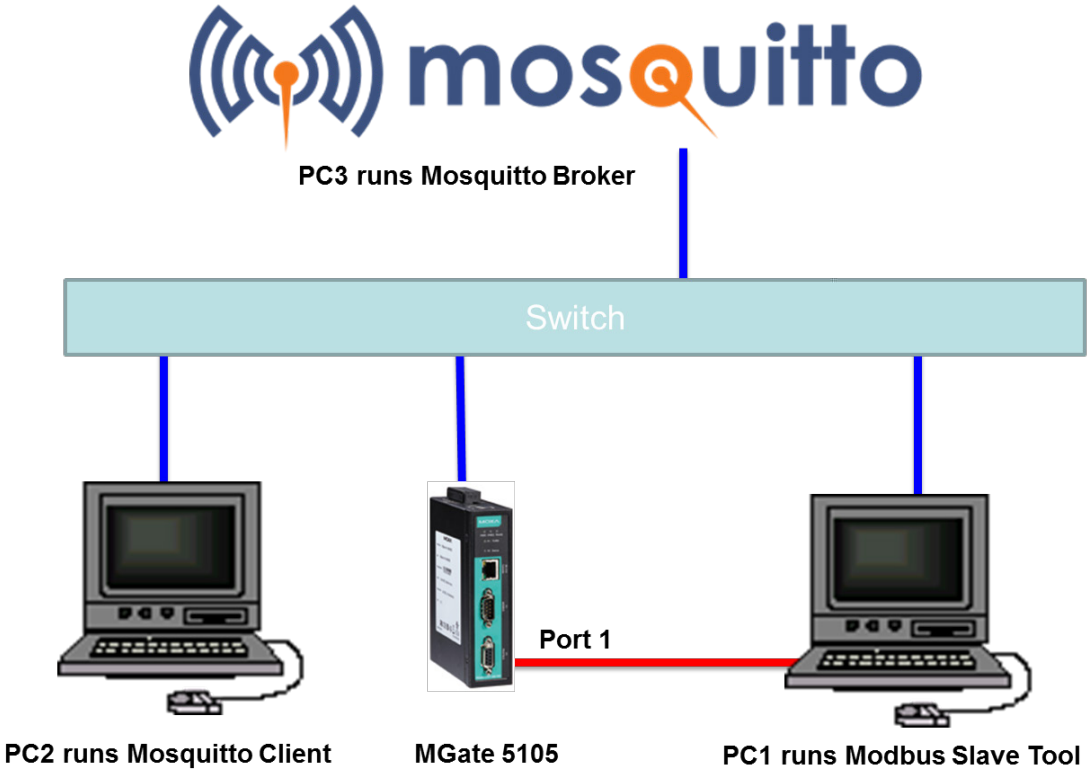
The MGate 5105 performs easy protocol conversions between Modbus RTU/ASCII, Modbus TCP, and EtherNet/IP protocols. From Firmware Versions 4.0 upwards support publishing the time stamps of the fieldbus devices to cloud servers. The cloud server include Microsoft Azure, Alibaba Cloud, or MQTT Broker.

This document demonstrates how to connect the MGate 5105 Eclipse Mosquitto MQTT Broker. We also demonstrate how to publish fieldbus data messages to and subscribe message from Mosquitto MQTT Broker.

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 Mosquitto MQTT Broker. PC3 runs Mosquitto MQTT Broker. PC2 runs Mosquitto Client to publish message to Mosquitto MQTT Broker and subscribe topics from Mosquitto MQTT Broker.

< 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 Eclipse Mosquitto

Eclipse Mosquitto is an open source message broker that implements the MQTT protocol versions 3.1 and 3.1.1. The binary installation packages are listed on the website: https://mosquitto.org/download/. In this demonstration, we use the Windows x64 version.

After installation, the Mosquitto tools are listed in C:\Program Files\mosquitto. There are several useful command line tools:

1. mosquitto.exe is a MQTT Broker.
2. mosquitto_pub.exe is a MQTT Client used to publish message.
3. mosquitto_sub.exe is a MQTT Client used to subscribe topic.

4. MGate 5105 Settings

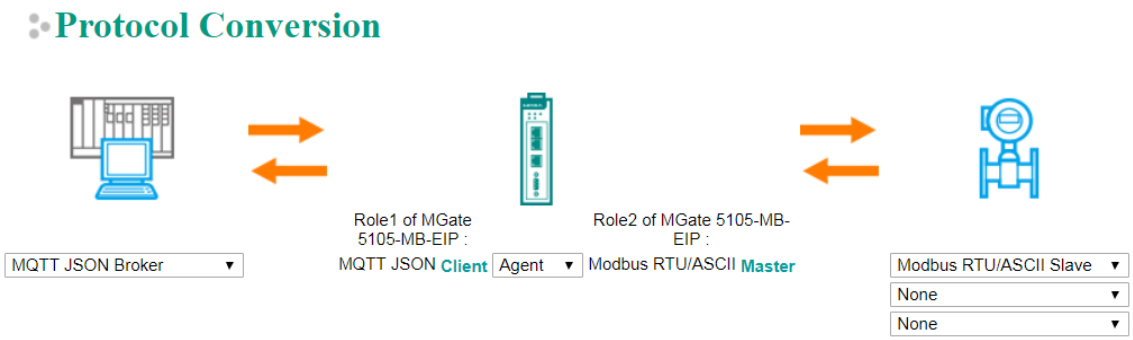
Log in to MGate 5105's web console, then do the following settings:

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:

Role	Master	
Mode	RTU	
Master Settings		
Initial delay	0	(0 - 30000 ms)
Max. retry	3	(0 - 5)
Response timeout	1000	(10 - 120000 ms)
Inter-frame delay	0	(10 - 500 ms, 0: default)
Inter-character timeout	0	(10 - 500 ms, 0: default)

Modbus Commands

Add Edit Clone Delete Move

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

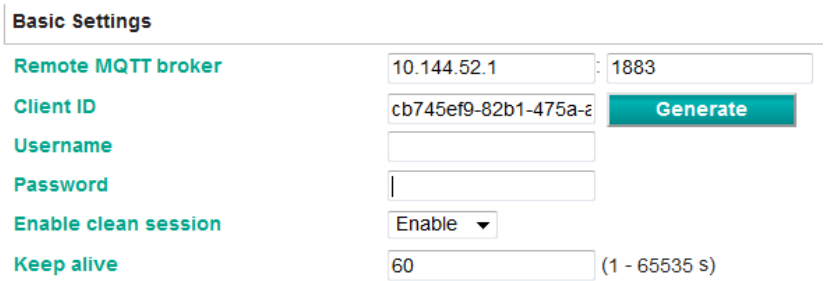
4.3 MQTT JSON Client Settings

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

   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.

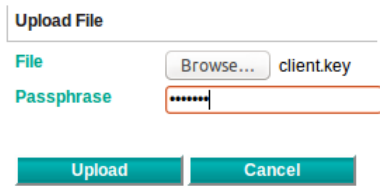
   Set as below:



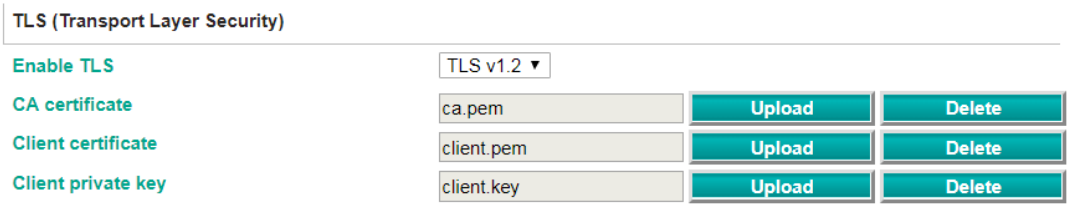
2. TLS (Transport Layer Security) setting:
   The MGate 5105 supports TLS to secure communications between MQTT Broker and Client. Here, we use version 1.2.

   To enable a TLS transmission, upload the CA certificate, client certificate, and client keyfile. The certificates and keyfile must be PEM encoded.

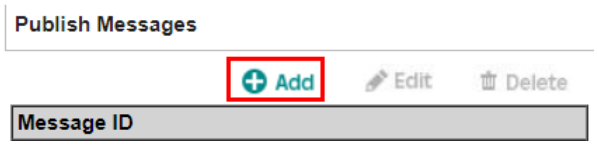
   If your keyfile has a passphrase, fill in the correct passphrase when uploading the keyfile, as below:



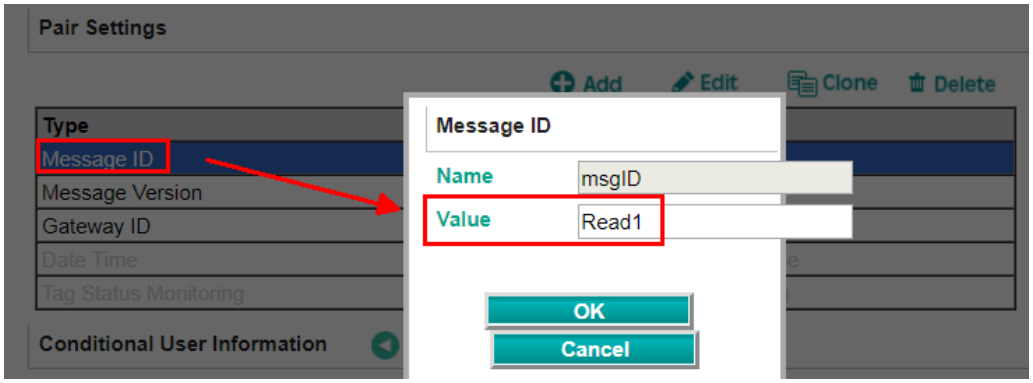
   Set as below:



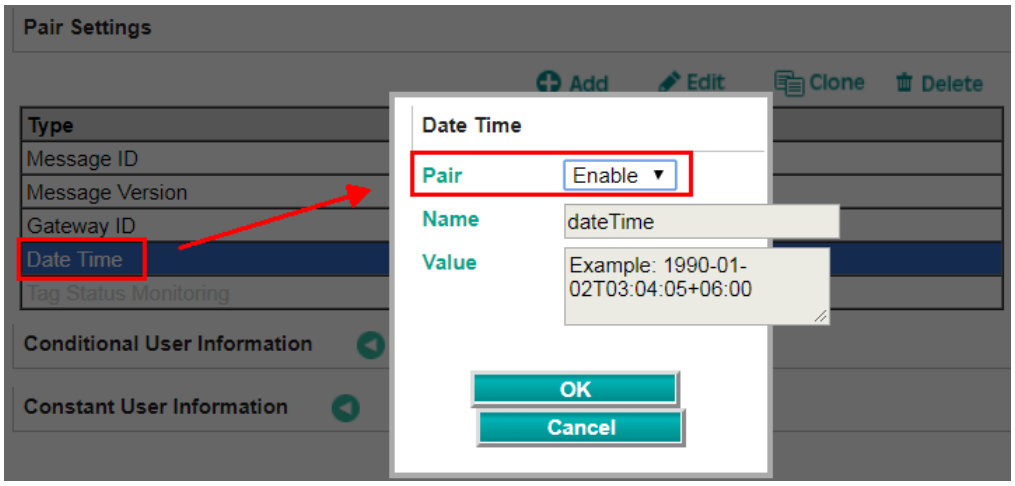
3. Publish Messages:
   Click the **Add** button to create a Publish Message and click it to edit the message settings.



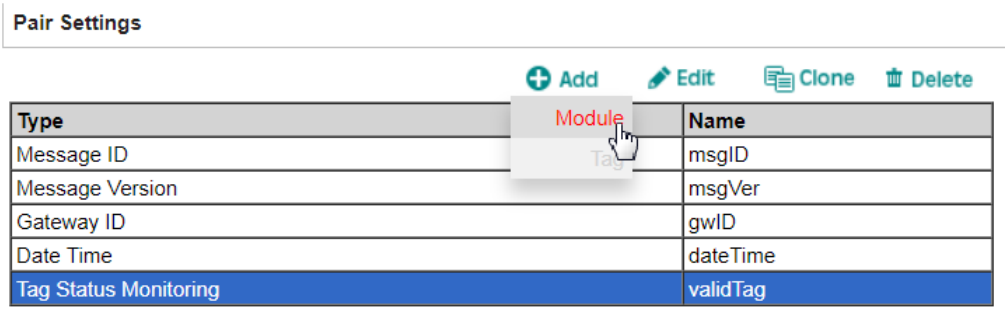
In **Pair Settings**, click **Message ID** to edit **Name**, and set **Value** as **Read1**.



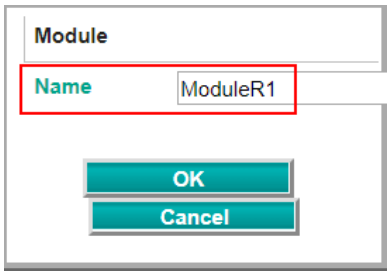
Click **Date Time** to enable **dateTime** padding in the message.



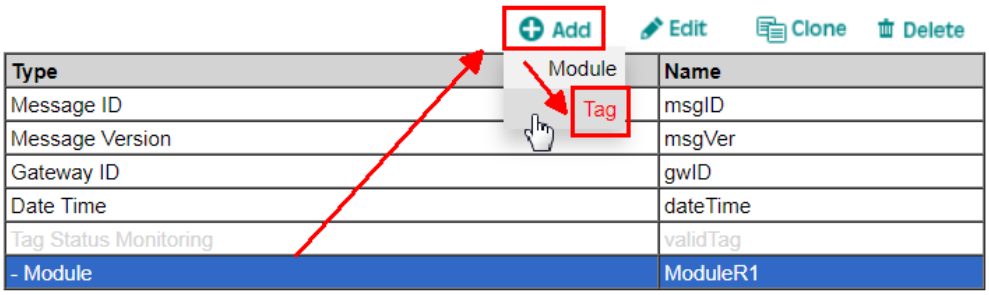
Click **Add** → **Module** to create a new module.



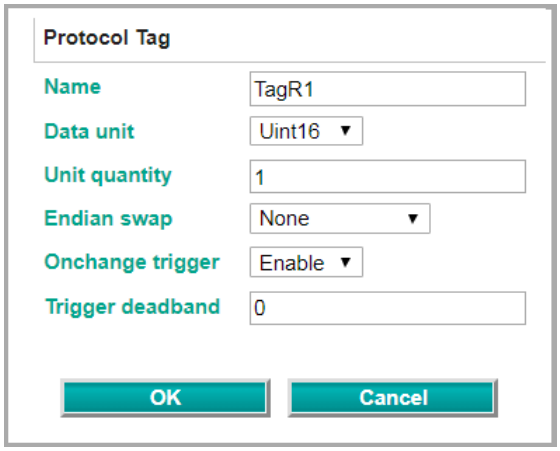
Set **Name** as **ModuleR1**.



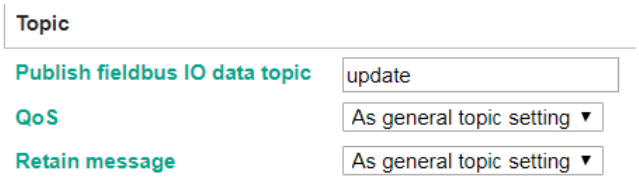
Choose **ModuleR1** and then click **Add** → **Tag**.



Create a Protocol Tag as below:

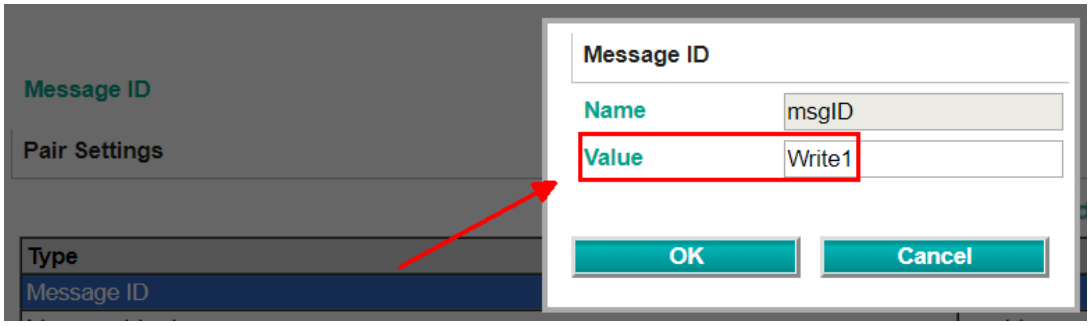


Set the topic name of this message as **update**:

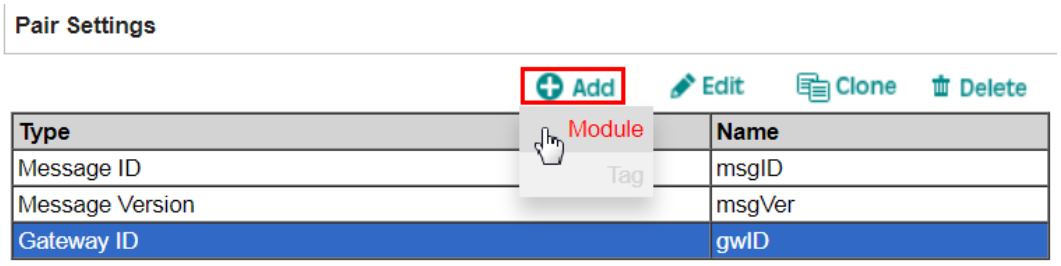


4. Subscribe Messages:
   Click the **Add** button to create a subscribe message and click it to edit message settings.

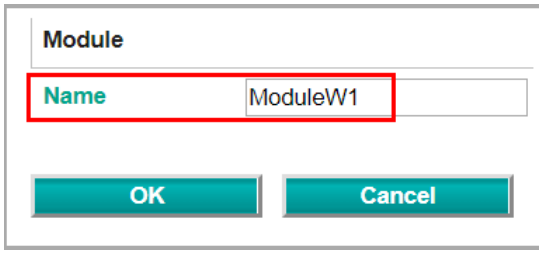
In **Pair Settings**, click **Message ID** to edit **Name** and set **Value** as **Write1**.



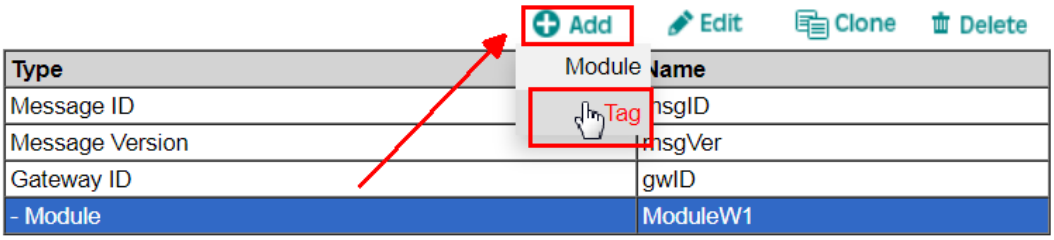
Click **Add** → **Module** to create a new module.



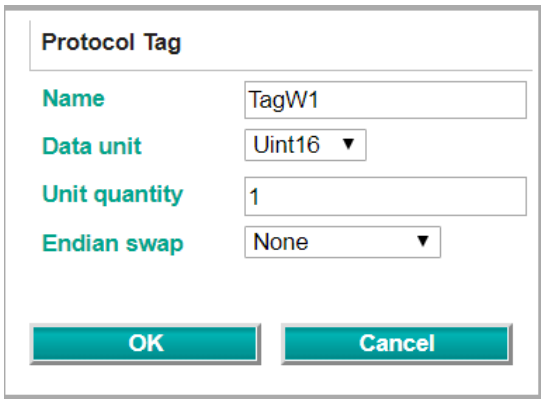
Set **Name** as **ModuleW1**.



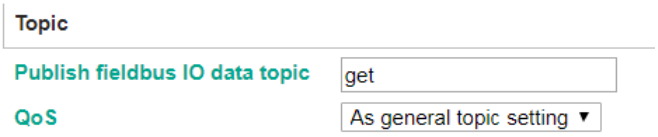
Choose **ModuleW1** and click **Add** → **Tag**.



Create a Protocol Tag as below:

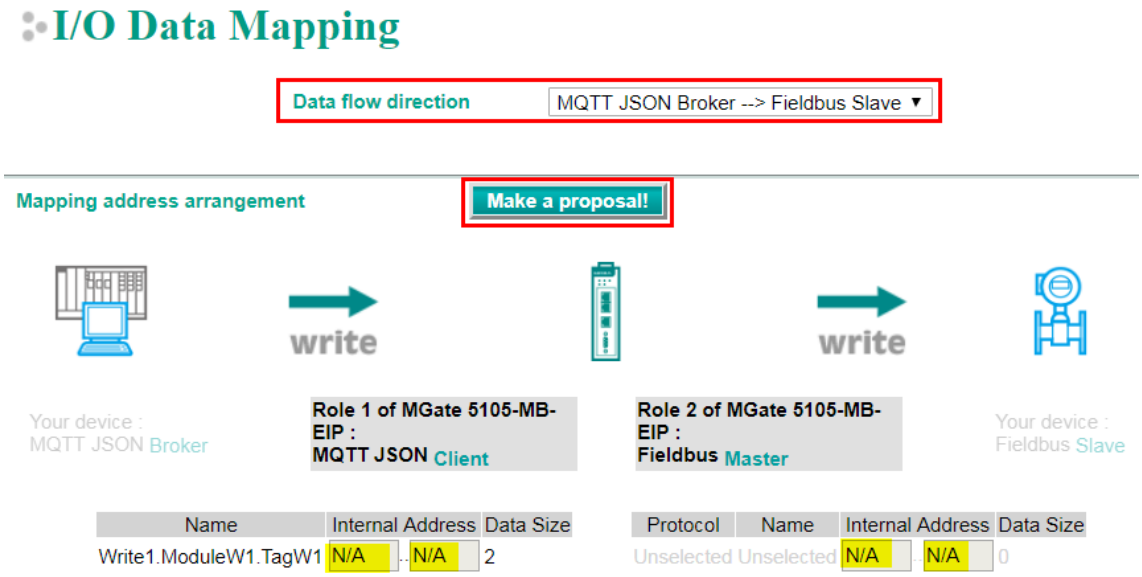


Set the topic name of this message as **get**.

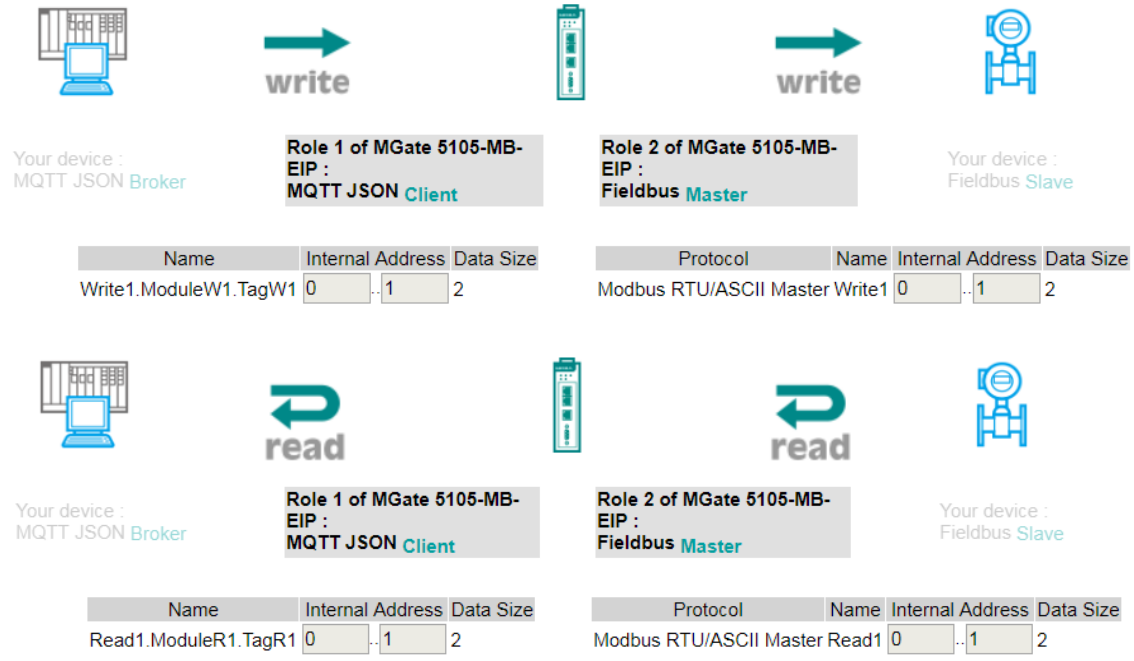


4.4 I/O Data Mapping

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

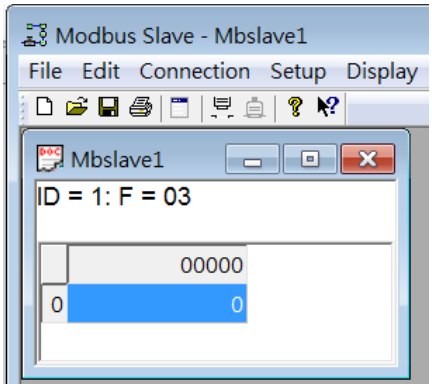
Set as below:

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. Mosquitto MQTT Broker Settings

If you need to customize the MQTT Broker settings, you can modify the mosquitto.conf file. The file is also under path **C:\Program Files\mosquitto**. Here are some customized settings:

1. TLS version

```
239 # This option defines the version of the TLS protocol to use for this listener.
240 # The default value allows v1.2, v1.1 and v1.0. The valid values are tlsv1.2
241 # tlsv1.1 and tlsv1.
242 #tls_version
243 tls_version tlsv1.2
```

2. CA certificate path

```
218 # At least one of cafile or capath must be defined. They both
219 # define methods of accessing the PEM encoded Certificate
220 # Authority certificates that have signed your server certificate
221 # and that you wish to trust.
222 # cafile defines the path to a file containing the CA certificates.
223 # capath defines a directory that will be searched for files
224 # containing the CA certificates. For capath to work correctly, the
225 # certificate files must have ".crt" as the file ending and you must run
226 # "openssl rehash <path to capath>" each time you add/remove a certificate.
227 #cafile
228 #capath
229 cafile C:\Cert\ca.pem
```

3. Server certificate path

```
231 # Path to the PEM encoded server certificate.
232 #certfile
233 certfile C:\Cert\server.pem
```

4. Server keyfile path

```
235 # Path to the PEM encoded keyfile.
236 #keyfile
237 keyfile C:\Cert\server.key
```

In command line, execute the mosquito.exe with –c and –v parameters as below:

```
C:\Program Files\mosquitto>mosquitto.exe -c mosquitto.conf -v
```

Parameter -c is using specific config file; -v is verbose mode.

The MQTT Broker is running with TLS enabled. If your server keyfile has a passphrase, a hint wondow will pop up to ask you to input your passphrase:

```
C:\Program Files\mosquitto>mosquitto.exe -c mosquitto.conf -v
1551937805: mosquitto version 1.5.8 starting
1551937805: Config loaded from mosquitto.conf.
1551937805: Opening ipv6 listen socket on port 1883.
1551937805: Opening ipv4 listen socket on port 1883.
Enter PEM pass phrase:
```

7. Mosquitto MQTT Clients Settings

On PC2, we will run mosquitto_sub.exe to subscribe MGate 5105 publish topic and use mosquitto_pub.exe to publish message to MQTT Broker anf then redirect to MGate 5105.

Two tool settings are as follows:

7.1 mosquitto_sub Setting

When you run mosquitto_sub, you can input parameters in command line. Another easy way is using a config file. You can add a config file name as mosquitto_sub.conf and put it in the path C:\Users\${Your username}\. Below is a config example:

```
#Broker Address
-h 10.144.52.1
#Broker Listen Port
-p 1883
#Client ID
--id SubClient
#TLS enable
--tls-version tlsv1.2
--insecure
--cafile C:\Cert\ca.pem
--cert C:\Cert\client.pem
--key C:\Cert\client.key
#Subscribe Topic
-t update
#Debug Mode
-d
```

When you run mosquitto_sub and your client keyfile has a passphrase, a hint window will pop up to ask you to input a passphrase as below:

```
C:\Program Files\mosquitto>mosquitto_sub
Enter PEM pass phrase:
```

If your passphrase is correct, it will connect to MQTT Broker and Subscribe “update” Topic.

```
Client SubClient sending CONNECT
Client SubClient sending CONNECT
Client SubClient received CONNACK (0)
Client SubClient sending SUBSCRIBE (Mid: 1, Topic: update, QoS: 0)
Client SubClient received SUBACK
```

7.2 mosquitto_pub Setting

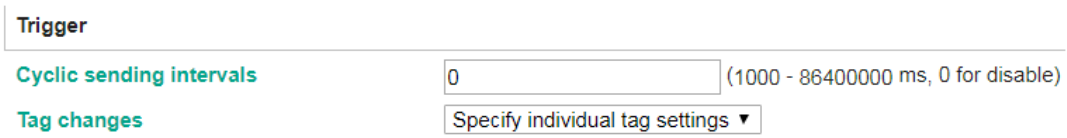
mosquitto_pub also can read its config file. Set config file name as mosquitto_pub.conf and put it in the same path C:\Users\${Your username}\. Below is a config example:

```
#Broker Address
-h 10.144.52.1
#Broker Listen Port
-p 1883
#Client ID
--id pubClient
#TLS enable
--tls-version tlsv1.2
--insecure
--cafile C:\Cert\ca.pem
--cert C:\Cert\client.pem
--key C:\Cert\client.key
#Debug Mode
-d
```

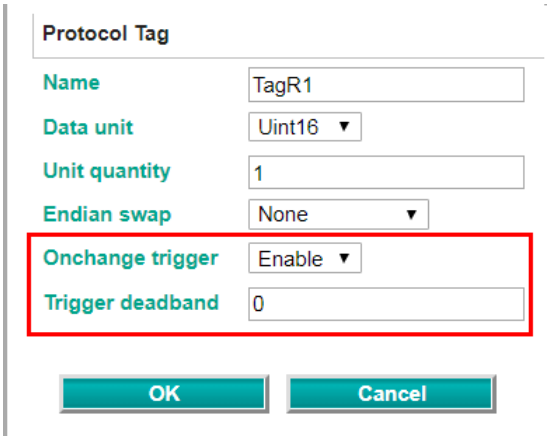
8. Communication Test

8.1 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 MQTT Broker.

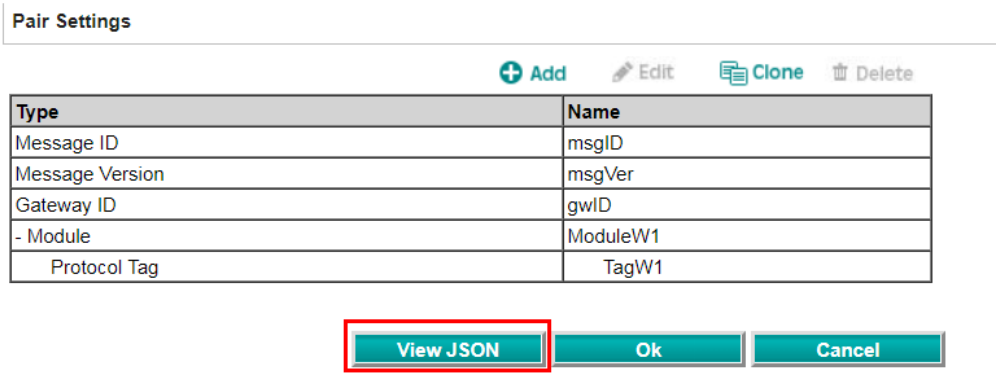
Now, update Modbus Register0's value as 1. In mosquitto_sub tool, TagR1's value is shown as 1 and with dateTime padding.

```
C:\Program Files\mosquitto>mosquitto_sub.exe -v
Client SubClient sending CONNECT
Client SubClient received CONNACK (0)
Client SubClient sending SUBSCRIBE (Mid: 1, Topic: update, QoS: 0)
Client SubClient received SUBACK
Subscribed (mid: 1): 0
Client SubClient received PUBLISH (d0, q0, r1, m0, 'update', ... (113 bytes))
update {"msgID":"Read1","msgVer":"1.0","gwID":"MGateMQTT","ModuleR1":{"TagR1":1},"dateTime":"2019-04-02T14:37:56+08:00"}
```

8.2 Subscribe message

Mosquito_pub can be used to send messages to the device. You can follow the steps below:

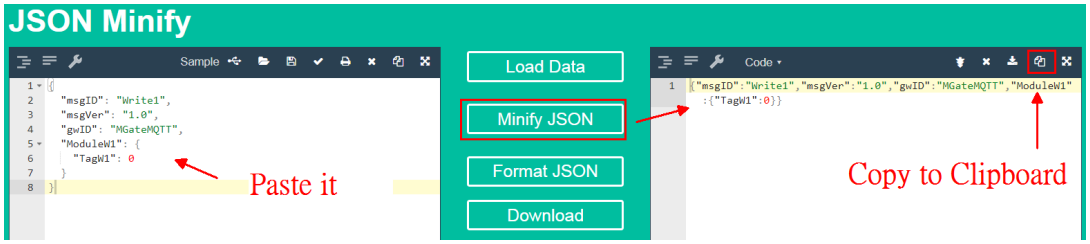
1. Click **View JSON** button.



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. Edit mosquitto_pub.conf to add –t parameter value as **get** and add –m parameter by pasting the clipboard message. Then, change **TagW1** value to 5 as below:

```
#Publish Topic
-t get
#Publish Message
-m {"msgID":"Write1","msgVer":"1.0","gwID":"MGateMQTT","ModuleW1":{"TagW1":5}}
```

4. Run mosquitto_pub.exe. You can see mosquitto_pub publish topic get the message to MQTT Broker.

```
C:\Program Files\mosquitto>mosquitto_pub.exe
Enter PEM pass phrase:
Client pubClient sending CONNECT
Client pubClient received CONNACK (0)
Client pubClient sending PUBLISH (d0, q0, r0, m1, 'get', ... (75 bytes))
Client pubClient sending DISCONNECT
```

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