cMT/HMI MQTT User Manual

Your Industrial Control Solutions Source

www.maplesystems.com

For use with the following:

- cMT Series
- Advanced HMI Series
- EBPro

Maple Systems, Inc. | 808 134th St. SW, Suite 120, Everett, WA 98204 | 425.745.3229

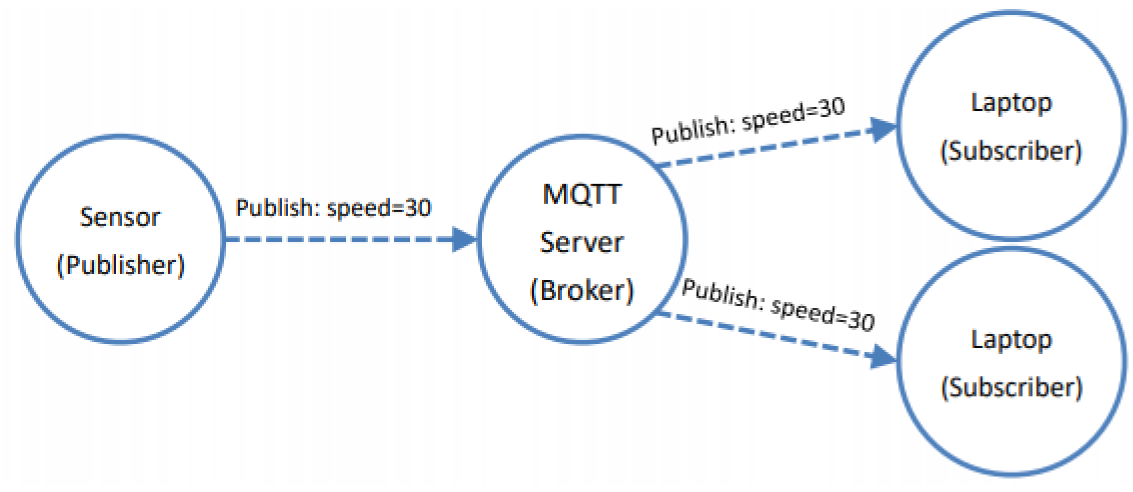
Table of Contents

Basic MQTT Features

1. Overview

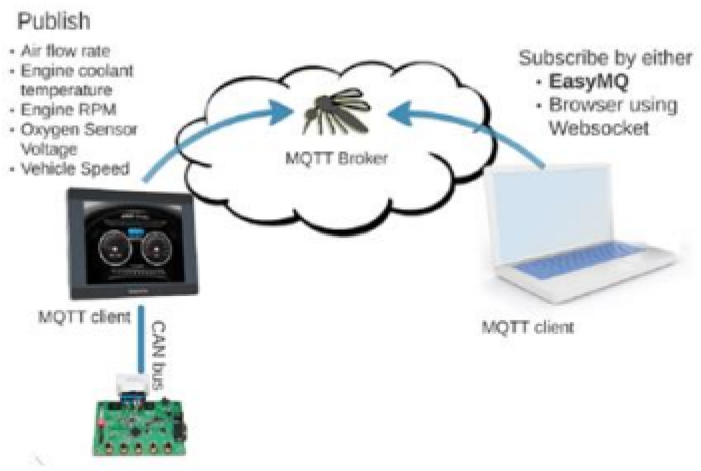
Designed to be light weight, open, and simple, MQTT is a subscriber/publisher messaging transport protocol that is considered a great solution for applications where a small code footprint is required and/or network bandwidth is scarce. It is particularly suitable for continuous monitoring of sensory data such as temperature, pressure, water level, energy monitoring…etc.

Publisher, Subscriber, and Broker are three important roles in MQTT protocol. As shown in the following figure, when the Publisher publishes a message to the Broker, the Broker will deliver the message to the subscriber.

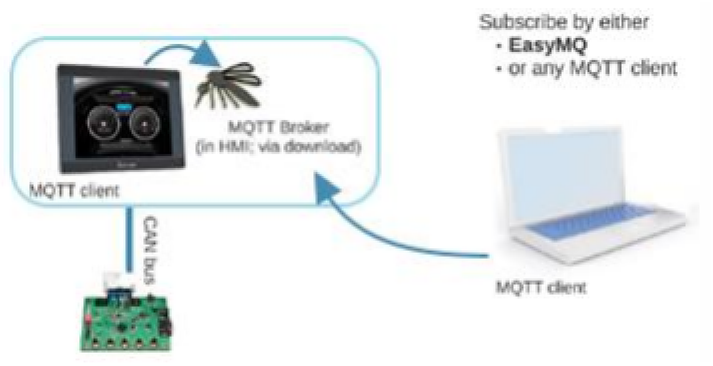


HMIs and MQTT

The Advanced HMI and cMT products can function as publishers, subscribers, and/or brokers. The HMI processes data from PLCs and can publish messages to an external MQTT broker, which will handle message delivery to the subscribers. In this manner, message publishing is possible even when the HMI is positioned behind firewalls and access to the HMI is difficult.



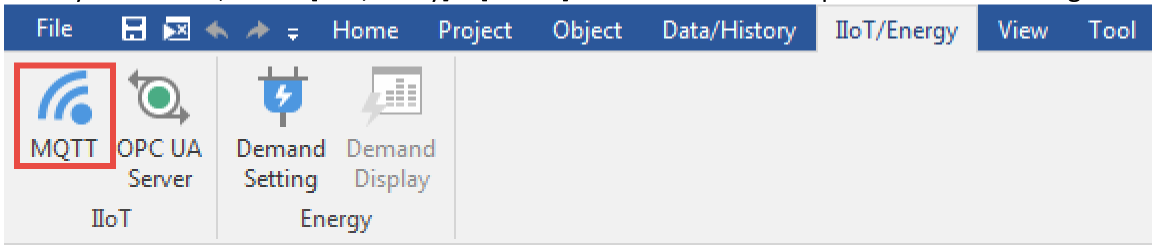
Alternatively, an external broker is not necessary. MQTT messages can be published to a built-in internal MQTT broker, and an MQTT client can subscribe directly to the MQTT broker inside the HMI to receive message updates. This scheme can be realized even remotely as long as the MQTT client can connect directly to HMI, such as with VPN or EasyAccess 2.0.



MQTT support was introduced in EZwarePlus version 5.03.02.026. The current feature set described in this manual is based on EasyBuilder Pro (the next generation of EZwarePlus) version 6.01.01.117. Not all features may be present in earlier versions. MQTT support is available in the Advanced HMI Series and the cMT Series. Some advanced features are limited to the cMT Series and are described later in this manual.

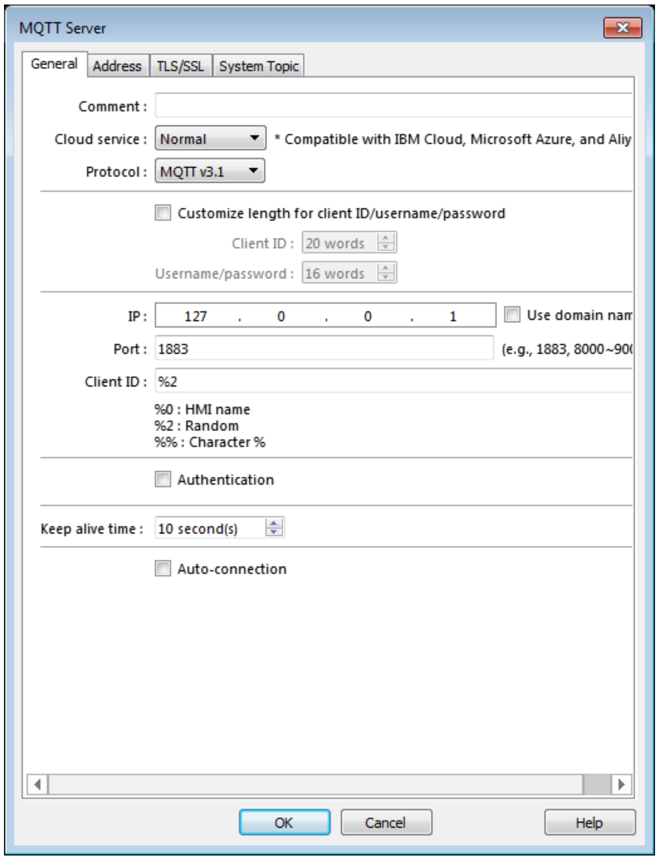
2. EasyBuilder Pro Settings

In EasyBuilder Pro, select [IIoT/Enery] » [MQTT] from the ribbon to open the MQTT settings.



Server Settings – General Tab

First, the MQTT Broker information must be entered. The IP address or domain name of the server is the IP address or domain name of the MQTT broker. When the local address (127.0.0.1) is used, messages will be published to the built-in broker in the HMI. To designate a broker by a domain name, select the [Use domain name] checkbox and enter the domain name in the IP field. To prevent unauthorized subscribers from connecting to the broker, select [Authentication] checkbox. With this checkbox selected, connecting MQTT Broker will require Username and Password. In [Auto-connection] mode, the connection will be automatically terminated if there's no data update for a specified period of time. Topic list shows the topics that can be subscribed on current HMI. To avoid publishing topic list every time when connecting the broker, select [Publish topic list only at the first time].



Comment :

Cloud service : Normal * Compatible with IBM Cloud, Microsoft Azure, and Aliy

Protocol : MQTT v3.1

☐ Customize length for client ID/username/password

Client ID : 20 words

Username/password : 16 words

IP : 127 . 0 . 0 . 1 ☐ Use domain nam

Port : 1883 (e.g., 1883, 8000~90

Client ID : %2

%0 : HMI name
%2 : Random
%% : Character %

☐ Authentication

Keep alive time : 10 second(s)

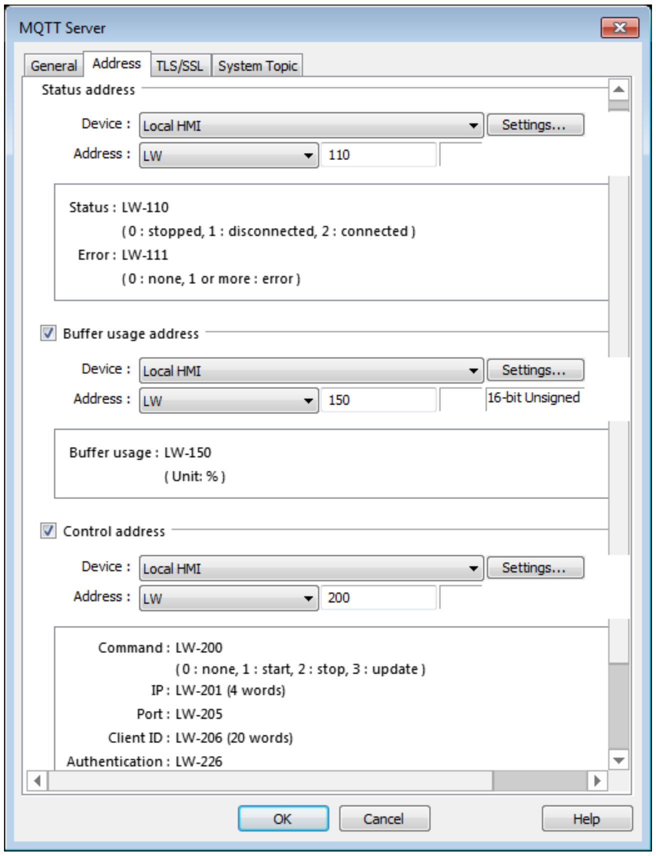
☐ Auto-connection

Setting	Description
Cloud service	**Normal** Use general publish-subscribe service. **AWS IoT** Use AWS IoT as a Broker, and use Thing Shadows service. For more information, see AWS IoT in the **Additional cMT Features** section of this manual. **Sparkplug B** Uses the Sparkplug B specification for use with Ignition SCADA software and other Sparkplug B module implementations. For more information, see Sparkplug B in the **Additional cMT Features** section of this manual.
Protocol	Supports MQTT v3.1 and v3.1.1.
Customize length for registration ID/ username/ password	Registration ID: The upper limit is 128 words. Username/Password: The upper limit is 256 words.
IP	Enter the MQTT Broker (Server) IP address or domain name for receiving the message. If 127.0.0.1 [Localhost] is used, the HMI will run an MQTT broker locally.
Use domain name	A domain name can be used as MQTT Broker's IP address.
Port	Enter the MQTT Broker port number for receiving the message. Commonly 1883 for standard MQTT connections, or 8883 for encrypted MQTT connections.
Client ID	Enter the client/registration ID.
Authentication	Check if selected MQTT Broker requires a [Username] and [Password].
Username	Enter the username for the MQTT Broker.
Password	Enter the password for the MQTT Broker.
Keep alive time	If the MQTT Broker does not receive a message from HMI beyond the specified time, the HMI will be identified as disconnected. Note: When running in simulation mode, messages may be delayed, but the delay will not exceed the [Keep alive time]. In production mode, the HMI sends messages immediately.
Auto-connection	In this mode, the connection will be automatically terminated if there's no data update for a specified period of time. The connection will resume once any data update occurs. The user can choose to publish initial values / topic list only at the first connection. In this mode, the start and stop commands are disabled.

Server Settings – Address Tab

LW addresses can be designated to dynamically control the MQTT connection or display MQTT status during HMI run time. After designating an address, its relative addresses (+1, +2, +3…etc.) will correspond to different attributes or parameters, as shown in the following EasyBuilder Pro settings dialog box. For instance, if MQTT_STATUS is set to LW-110, then LW-110 shows the status and LW-111 shows the error code.

Messages that have not been sent are stored in the buffer. The maximum buffer capacity is 10000 messages. When the buffer is full, the earliest message will be deleted.



Status address

Device : Local HMI Settings...

Address : LW 110

Status : LW-110
(0 : stopped, 1 : disconnected, 2 : connected)
Error : LW-111
(0 : none, 1 or more : error)

☑ Buffer usage address

Device : Local HMI Settings...

Address : LW 150 16-bit Unsigned

Buffer usage : LW-150
(Unit: %)

☑ Control address

Device : Local HMI Settings...

Address : LW 200

Command : LW-200
(0 : none, 1 : start, 2 : stop, 3 : update)
IP : LW-201 (4 words)
Port : LW-205
Client ID : LW-206 (20 words)
Authentication : LW-226

The MQTT_COMMAND control addresses can be designated to set control parameters, and the corresponding addresses include MQTT_COMMAND+1 ~ MQTT_COMMAND+70…etc. The status addresses can show connection status, and the corresponding addresses include MQTT_STATUS and MQTT_STATUS+1.

The MQTT settings can be changed dynamically during HMI run time by using the control addresses. By default, the HMI will automatically try to connect to the MQTT Broker on startup.

During HMI run time:

Setting MQTT_COMMAND to 1 connects HMI with the broker.

Setting MQTT_COMMAND to 2 disconnects HMI with the broker.

Setting MQTT_COMMAND to 3 after updating the control parameters will connect HMI with the broker using the new parameters.

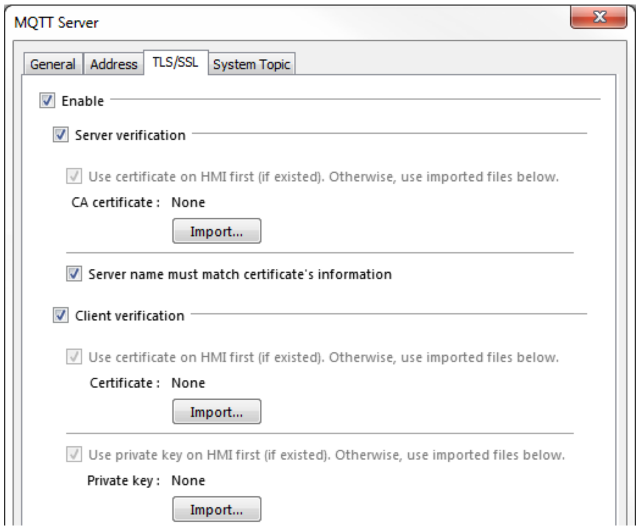
Setting	Description		
Status address	LW-n: Displays the connection status to MQTT Broker:		
		Value	Description
		0	Not attempting to connect to MQTT Broker.
		1	Disconnected and can’t connect to MQTT Broker.
		2	Connection succeeded.
	LW-n+1: Error indicator:		
		Value	Description
		0	No error
		1 or more	An error occurred
Buffer usage address	Messages that have not been sent are stored in the buffer. The maximum buffer capacity is 10000 messages. The buffer capacity is measured in percentage (%), rounded up. LW-n: Shows buffer usage.		
Control	LW-n: Displays the connection status to MQTT Broker:		
		Value	Description
		0	Ready
		1	Start
		2	Stop
		3	Update
	LW-n+1: Sets the IP address of MQTT Broker. LW-n+5: Sets the port number of MQTT Broker. LW-n+6: Sets the Registration ID for connecting MQTT Broker. LW-n+70: Enables / Disables authentication:		
		Value	Description
		0	Disable
		1	Enable
	LW-n+71: Sets the username for connecting MQTT Broker. LW-n+199: Sets the password for connecting MQTT Broker		

Server Settings – TLS/SSL Tab

Enabling TLS/SSL authentication opens two verification methods:

[Server verification]: Verify whether the server certificate is signed by CA (certificate authority) certificate.

[Client verification]: By providing a private key and certificate, the server can verify the client faster, skipping login by username or password.



MQTT Server

General | Address | TLS/SSL | System Topic

☑ Enable

☑ Server verification

☑ Use certificate on HMI first (if existed). Otherwise, use imported files below.

CA certificate : None

Import...

☑ Server name must match certificate's information

☑ Client verification

☑ Use certificate on HMI first (if existed). Otherwise, use imported files below.

Certificate : None

Import...

☑ Use private key on HMI first (if existed). Otherwise, use imported files below.

Private key : None

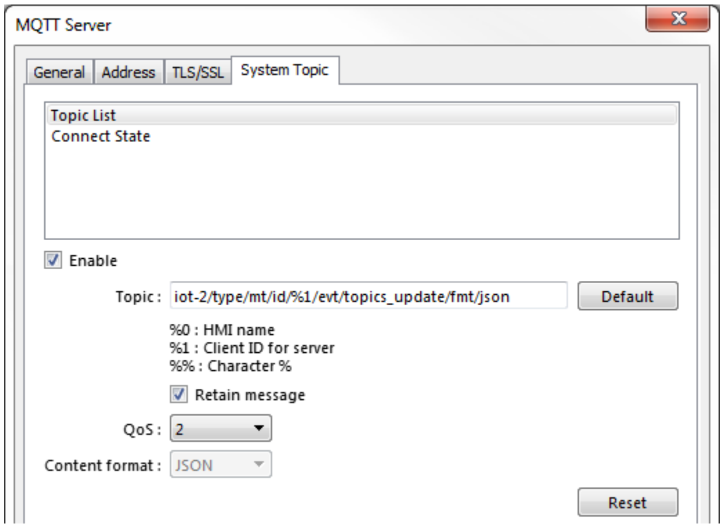
Import...

Server Settings – System Topic Tab

By default, when the HMI is the publisher and is connected to the broker for the first time, two default topics published at QoS2 will be sent from HMI. These messages can be edited or disabled.

1. iot-2/type/mt/id/<Client ID>/evt/topics_update/fmt/json
   This topic contains the messages sent from the HMI, and shows whether the messages are compressed.

2. iot-2/type/mt/id/<Client ID>/evt/status/fmt/json
   This topic shows the connection status between HMI and broker.



MQTT Server

General | Address | TLS/SSL | System Topic

Topic List
Connect State

☑ Enable

Topic : iot-2/type/mt/id/%1/evt/topics_update/fmt/json Default

%0 : HMI name
%1 : Client ID for server
%% : Character %

☑ Retain message

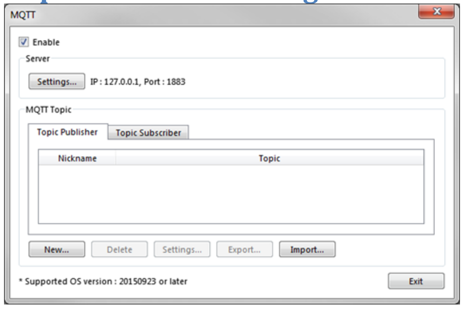
QoS : 2

Content format : JSON

Reset

Setting	Description
Topic List	**Enable** Selecting this checkbox puts the specified topic into the broker's topic list, which includes topics published by different HMIs. At the first time the subscriber connects to the broker, the broker will send the specified topic to the subscriber. Alternatively, the subscriber can subscribe to this topic to view the available topics in the broker. **Retain message** When this checkbox is selected, the MQTT broker will save the latest message.
Connect State	**Enable** Selecting this checkbox displays the connection state between the broker and the HMI (publisher). At the first time the subscriber connects to the broker, the broker will send the specified topic to the subscriber. Alternatively, the subscriber can subscribe to this topic to monitor the connection state between the broker and other HMIs (publisher). **Retain message** When this checkbox is selected, the MQTT broker will save the latest message.

Topic Publisher Settings



☑ Enable

Server

Settings... IP : 127.0.0.1, Port : 1883

MQTT Topic

Nickname	Topic

* Supported OS version : 20150923 or later

Click [New] to open General and Address settings, or click [Import] / [Export] to import or export an existing *.csv file. The maximum allowable number of topics is 255. Each topic contains a number of messages to be sent.

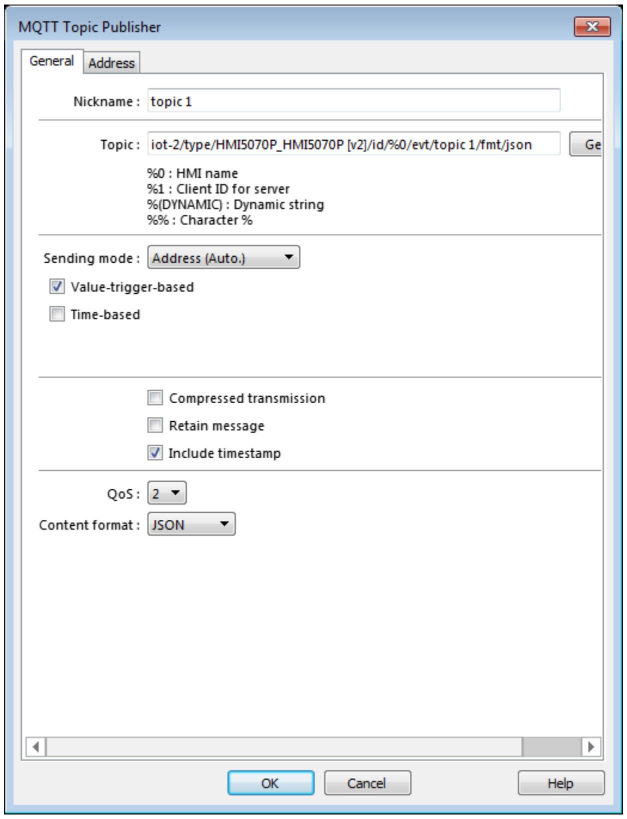
The name of the Topic can be user-defined, and by using the character % followed by certain codes, HMI name/Server setting can be used in Topic name as well.

For example: iot-2/type/cMT-SVR/id/%0/evt/topic 1/fmt/json

%0 stands for HMI name. To find out HMI name, open System Settings on HMI, or use system register LW-10884. If the HMI name is “Default HMI”, then the topic should be written as: iot-2/type/cMT-SVR/id/Default HMI/evt/topic 1/fmt/json

Sending Mode: If Trigger-based is selected, the MQTT message is sent when any value in the Topic changes. If Time-based is selected, data is published at a fixed time interval.

Apart from Address sending mode, when an Event Log is created, [Event Log] option can be found in the MQTT Topic Publisher settings dialog.



Nickname : topic 1

Topic : iot-2/type/HMI5070P_HMI5070P [v2]/id/%0/evt/topic 1/fmt/json

%0 : HMI name
%1 : Client ID for server
%(DYNAMIC) : Dynamic string
%% : Character %

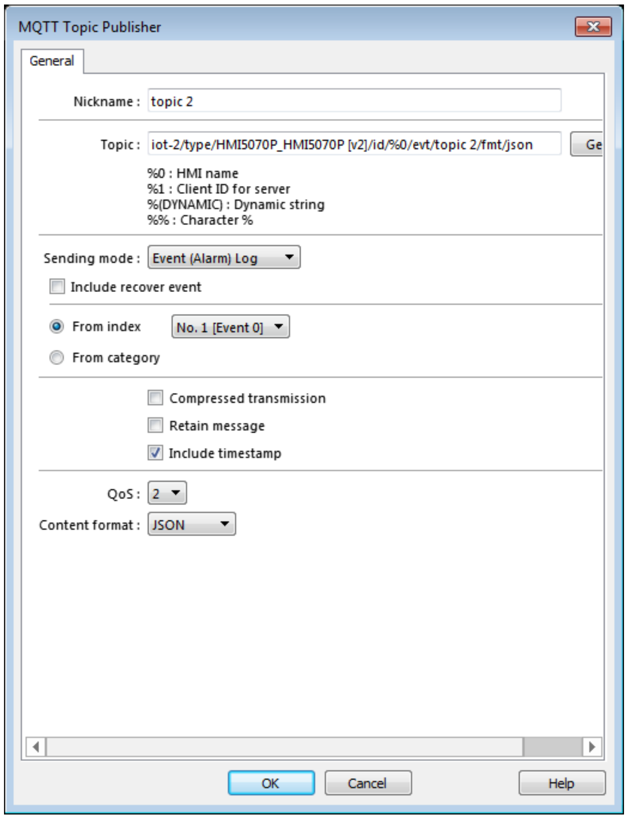
Sending mode : Address (Auto.)

☑ Value-trigger-based
☐ Time-based

☐ Compressed transmission
☐ Retain message
☑ Include timestamp

QoS : 2

Content format : JSON



Nickname : topic 2

Topic : iot-2/type/HMI5070P_HMI5070P [v2]/id/%0/evt/topic 2/fmt/json

%0 : HMI name
%1 : Client ID for server
%(DYNAMIC) : Dynamic string
%% : Character %

Sending mode : Event (Alarm) Log

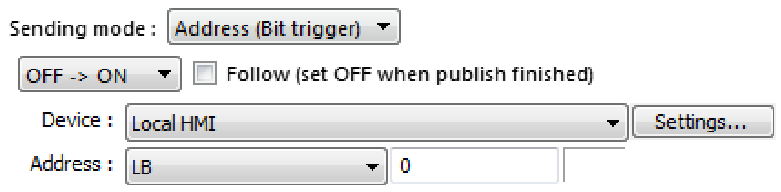
☐ Include recover event

● From index No. 1 [Event 0]
○ From category

☐ Compressed transmission
☐ Retain message
☑ Include timestamp

QoS : 2

Content format : JSON


Sending mode : Address (Bit trigger)

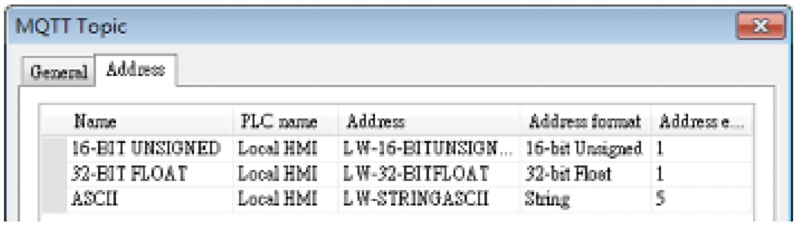
OFF -> ON ☐ Follow (set OFF when publish finished)

Device : Local HMI Settings...

Address : LB 0

Setting	Description
Nickname	Enter a nickname for easier reference.
Topic	Specify the format of the message topic sent to MQTT Broker.
Sending mode	**Address (Auto)** The HMI can publish data when any data in the topic changes [Value-trigger-based] or at a pre-defined time interval [Time-based], or both. **Address (Bit trigger)** The HMI only publishes data when a defined control bit changes. Available option include rising-edge detection, falling-edge detection, both with optional reset, or both rising and falling-edge detection (any change). **Event (Alarm) Log** The topic source can also be an Event Log.
Compressed transmission	The message will be compressed before being sent, and decompression is needed before reading the message. Messages in MQTT are compressed / decompressed with DEFLATE algorithm.
Retain message	If selected, the MQTT Broker will save the latest message.
Include timestamp	When the format used is JSON, selecting this option can include timestamp in the message.
QoS	MQTT provides three levels of reliability, which are known as qualities of service (QoS). The reliability of the message determines the persistence of the message. 0: At most once, messages are not persistent. 1: At least once. 2: Exactly once.
Content Format	The supported formats are: JSON and Raw Data.

When the sending mode is [Address], in Address tab, set the data composition that will be contained in the topic. The addresses can be consecutive or nonconsecutive, and of different data types and lengths.



MQTT Topic

General | Address

Name	PLC name	Address	Address format	Address e...
16-BIT UNSIGNED	Local HMI	LW-16-BITUNSIGN...	16-bit Unsigned	1
32-BIT FLOAT	Local HMI	LW-32-BITFLOAT	32-bit Float	1
ASCII	Local HMI	LW-STRINGASCII	String	5

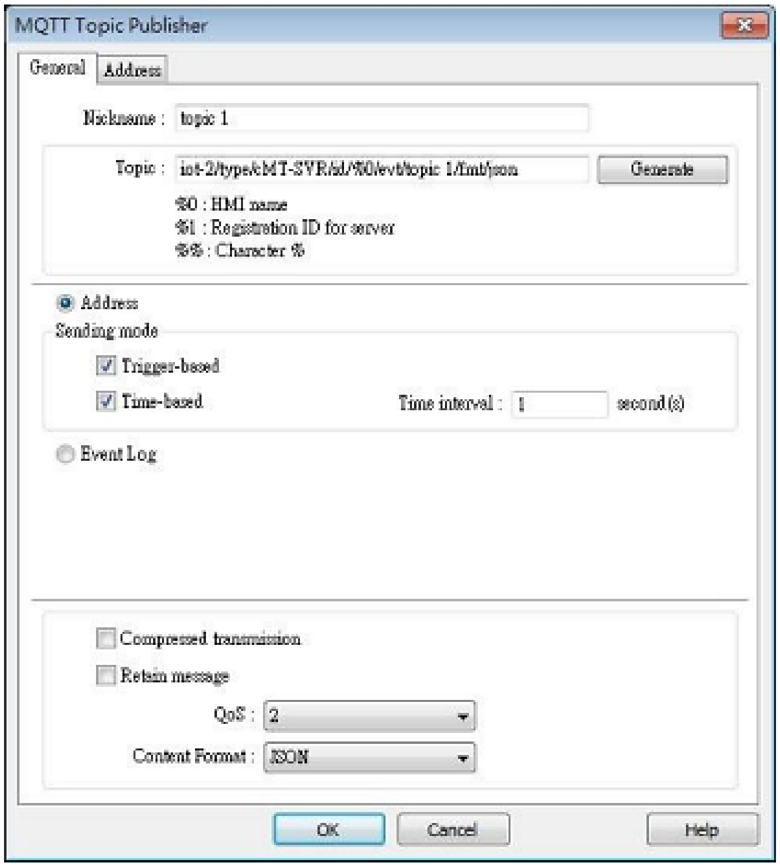
MQTT provides three levels of reliability, which are known as qualities of service (QoS). The reliability of the message determines the persistence of the message.

0: At most once, messages are not persistent.
1: At least once.
2: Exactly once.

For more information on QoS, click here.

[Compressed transmission]: The message will be compressed before being sent, and decompression is needed before reading the message. Messages in MQTT are compressed / decompressed with DEFLATE algorithm.

[Content format]: The supported formats are: JSON and Raw Data.



MQTT Topic Publisher

General | Address

Nickname : topic 1

Topic : iot-2/type/cMT-SVR/id/%0/evt/topic 1/fmt/json

%0 : HMI name
%1 : Registration ID for server
%% : Character %

(●) Address

Sending mode

[✓] Trigger-based

[✓] Time-based   Time interval : 1 second(s)

( ) Event Log

[ ] Compressed transmission

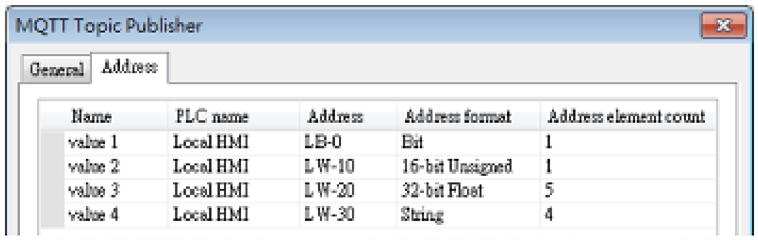
[ ] Retain message

QoS : 2

Content Format : JSON

Below is an example that shows the difference between JSON and Raw Data.

When publishing four values in the following address formats:



Name	PLC name	Address	Address format	Address element count
value 1	Local HMI	LB-0	Bit	1
value 2	Local HMI	LW-10	16-bit Unsigned	1
value 3	Local HMI	LW-20	32-bit Float	5
value 4	Local HMI	LW-30	String	4

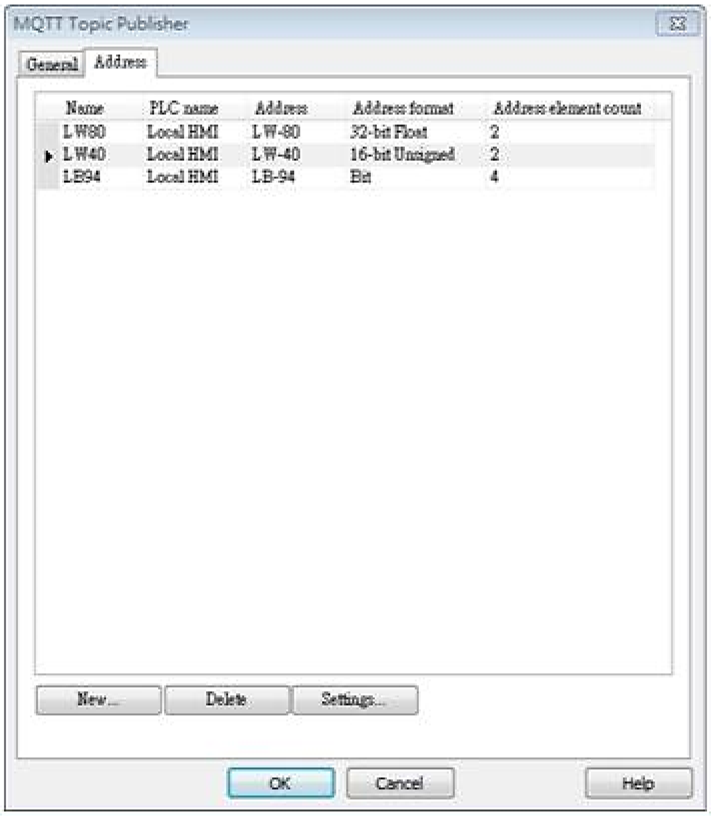
JSON:

```
{
    "d" : {
        "value 1" : [ false ],
        "value 2" : [ 2 ],
        "value 3" : [ 1.20000005, 0, 0, 0, 0 ],
        "value 4" : [ "ABCD" ]
    },
    "ts" : "2017-04-18T17:36:52.501856"
}
```

Raw data:

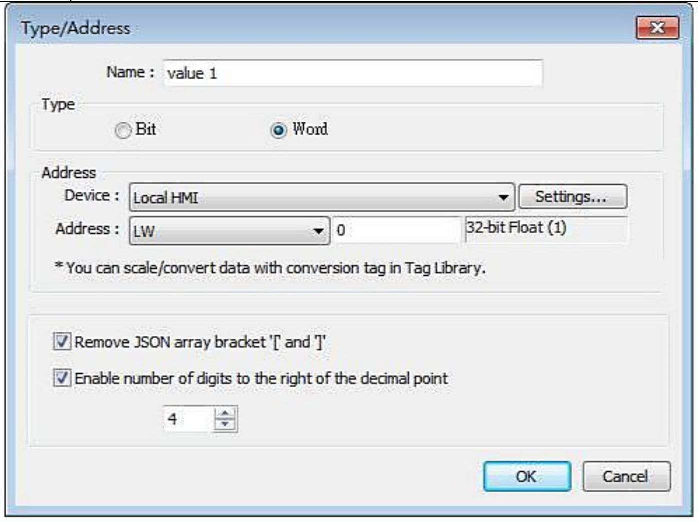
```
0002 009A 9999 3F00 0000 0000 0000 0000
0000 0000 0000 0041 4243 4400 0000 00
```

Topic Publisher – Address Tab



Name	PLC name	Address	Address format	Address element count
LW80	Local HMI	LW-80	32-bit Float	2
LW40	Local HMI	LW-40	16-bit Unsigned	2
LB94	Local HMI	LB-94	Bit	4

Setting	Description
New	Add the source of the topic. The length of each address can be specified respectively.
Delete	Delete the address.
Setting	Change the name and address.



Name : value 1

Type: Bit [✓ Word]

Address
Device : Local HMI
Address : LW 0 32-bit Float (1)

* You can scale/convert data with conversion tag in Tag Library.

[✓] Remove JSON array bracket '[' and ']'

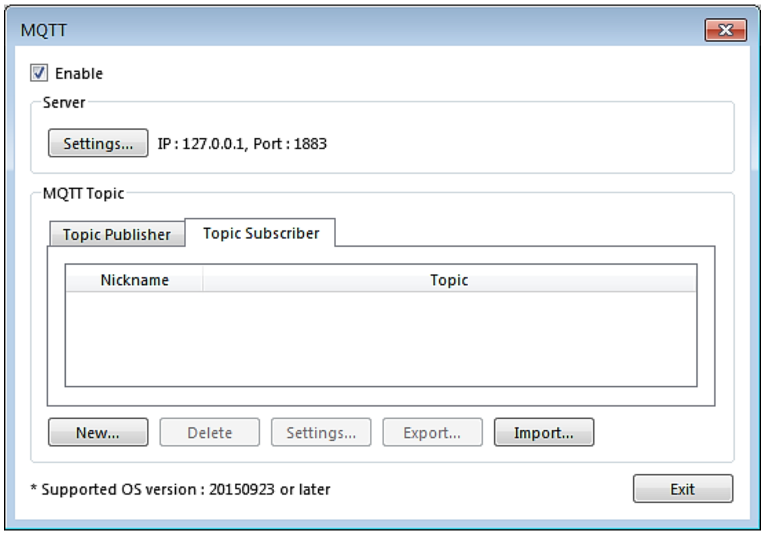
[✓] Enable number of digits to the right of the decimal point

4

Setting	Description
Remove JSON array bracket "[" and "]"	For JSON formatted messages, selecting this option can remove bracket "[" and "]".
Enable number of digits to the right of the decimal point	When data type is Float, the number of digits after the decimal point can be specified.

Topic Subscriber Settings

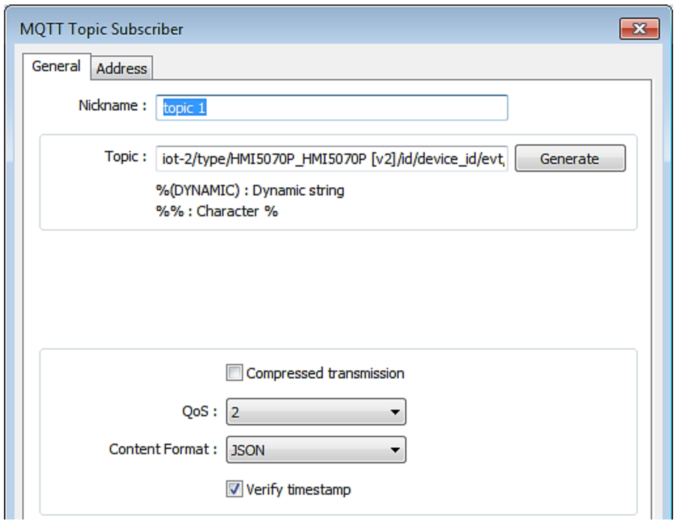
Advanced and cMT Series HMIs can subscribe topics from other MQTT Brokers. The settings are similar to Publisher settings demonstrated in the preceding chapters. Subscribing to Event Logs is not supported.



IP : 127.0.0.1, Port : 1883

* Supported OS version : 20150923 or later

Click [New] to open General and Address settings, or click [Import] / [Export] to import or export an existing *.csv file. The maximum allowable number of topics is 255.



Nickname : topic 1

Topic : iot-2/type/HMI5070P_HMI5070P [v2]/id/device_id/evt,

%(DYNAMIC) : Dynamic string
%% : Character %

Compressed transmission

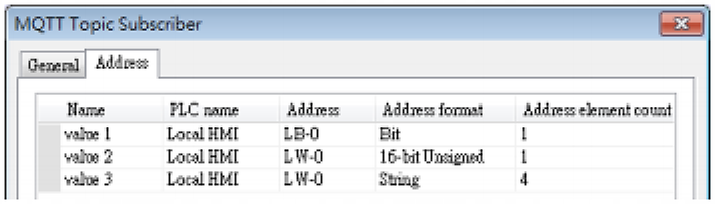
QoS : 2

Content Format : JSON

Verify timestamp

Setting	Description
Nickname	Enter a nickname for easier reference.
Topic	Subscribe to a topic in an MQTT Broker. The topic name can be dynamic.
Compressed transmission	Configure with the same setting as the MQTT Topic Publisher.
QoS	MQTT provides three levels of reliability, which are known as qualities of service (QoS). The reliability of the message determines the persistence of the message. 0: At most once, messages are not persistent. 1: At least once. 2: Exactly once.
Content Format	The supported formats are: JSON and Raw Data. Select a format that is supported by the publisher.
Verify timestamp	When a timestamp is included in the message, selecting this option will verify whether the timestamp is increasing, and will only update when the timestamp does increase. Otherwise, the message will be indicated as an earlier message and discarded.

The address format should be specified according to the subscribed topic. For example, if the topic contains the following address formats: Bit, 16bit-unsigned, String (length 4). The address settings should be:



Name	PLC name	Address	Address format	Address element count
value 1	Local HMI	LB-0	Bit	1
value 2	Local HMI	LW-0	16-bit Unsigned	1
value 3	Local HMI	LW-0	String	4

In this example, the order from value 1 to 3 should be: Bit, 16bit-unsigned, String. **The order cannot be changed, and the address element count should be identical.**

3. Choosing a Broker

Built-in Broker in HMI

To use the built-in broker in HMI, set the IP address to 127.0.0.1 in MQTT Server settings dialog box. MQTT will use the built-in broker in HMI, and client programs can connect to the broker using the IP address of the HMI.

The HMI's MQTT Broker must be downloaded to the HMI from EasyBuilder Pro. Select [Runtime] when downloading to set up an MQTT Broker.

External Broker

To use an external public broker, enter the IP address or domain name of the external broker when configuring MQTT server information.

For example, when using a public broker HiveMQ, the data of the broker is:

Host: broker.hivemq.com
Port: 1883
(http://www.hivemq.com/try-out/)

Self-built Broker

It is possible to host an MQTT Broker on your own server hardware using third party software. The following are some verified choices:

1. Mosquitto (http://mosquitto.org/download/)
2. HiveMQ (http://www.hivemq.com/downloads/) (test version provided)
3. EMQ (http://emqtt.io/)

Please visit their respective official websites for details on installation and restrictions.

4. Getting MQTT Data

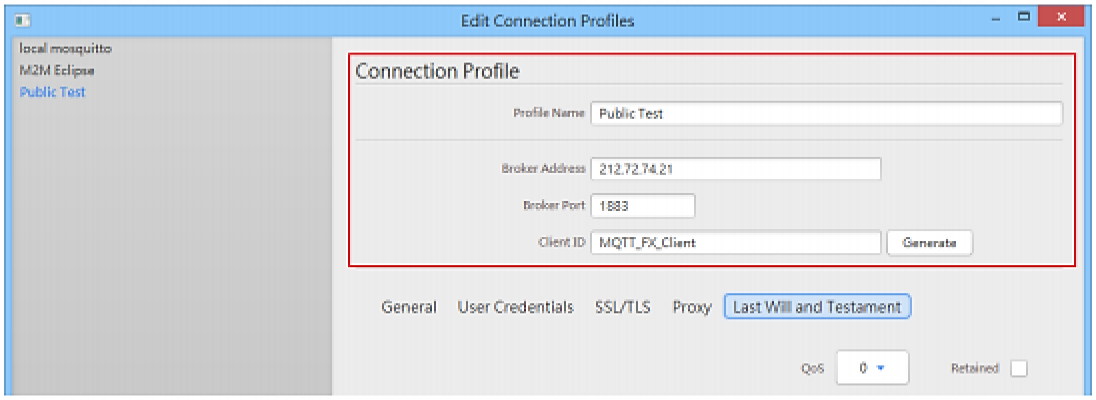
Getting MQTT data updates requires an MQTT client program. The client program connects to the broker and registers to receive data updates (subscribes to topics) from the HMI. Many free client programs can be found on the web. This chapter introduces one of the free programs for PC: MQTT.fx, available for free download here:

http://mqttfx.jensd.de/index.php/download

Client Program

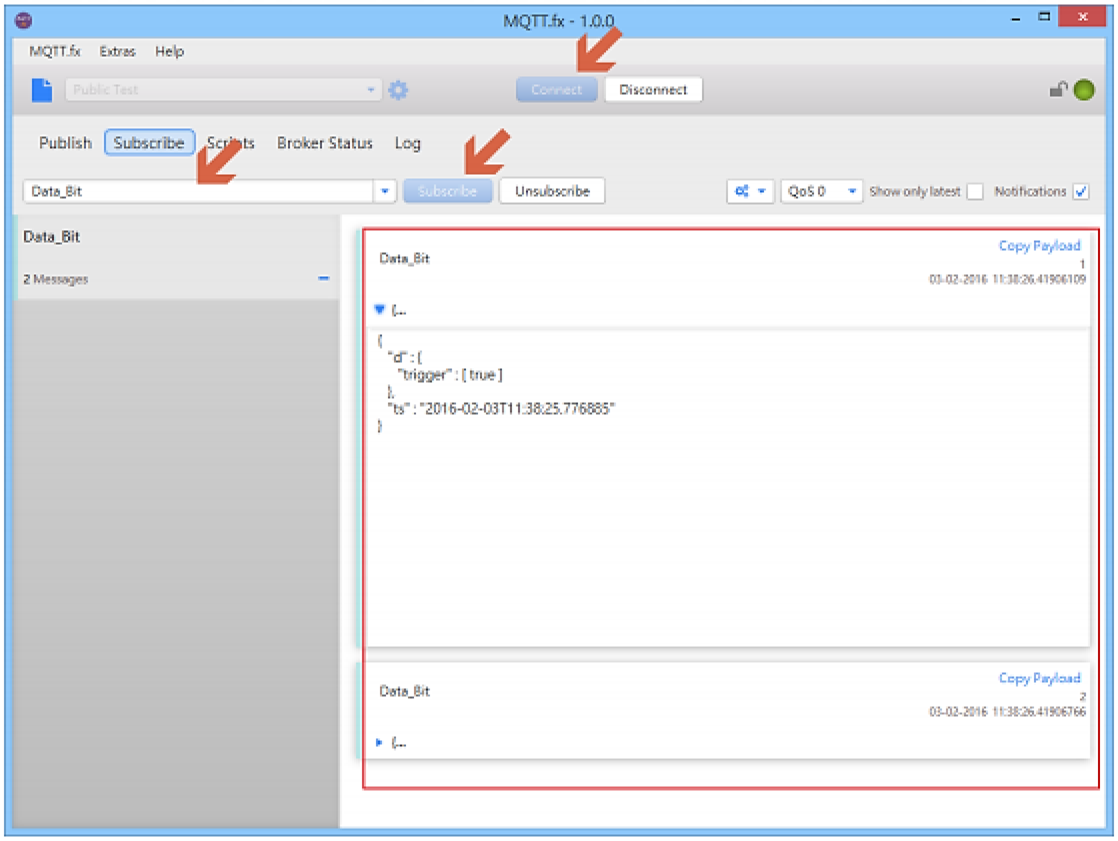
Many free MQTT client programs for PC and portable can be found on the web. The following briefly walks through the steps to use MQTT.fx on PC to connect to the broker in order to get message updates from HMI. For other programs, configuration steps may be similar.

1. Suppose an HMI is running MQTT Demo project. It's been configured to connect to MQTT Broker at broker.hivemq.com, port 1883. It has an MQTT topic called Data_Bit.
2. In MQTT.fx » Edit Connection Profiles window, the connection profile should be set as follows.
   Broker Address: broker.hivemq.com
   Broker Port: 1883
   Profile Name: user defined
   Client ID: user defined
   The rest of the settings can remain in default. The connection with broker will start after clicking [Connect].



Profile Name: Public Test
Broker Address: 212.72.74.21
Broker Port: 1883
Client ID: MQTT_FX_Client
QoS: 0
Retained: ☐

3. Open the Subscribe tab » enter Data_Bit in the field shown below » click [Subscribe] button. Once successfully subscribed, when there are MQTT message updates, they will be received and displayed on the main screen.



Data_Bit
1
03-02-2016 11:38:26.41906109

```
{
  "d" : {
    "trigger" : [ true ]
  },
  "ts" : "2016-02-03T11:38:25.776885"
}
```

Data_Bit
2
03-02-2016 11:38:26.41906766

Additional cMT Features

cMT Series HMIs and Gateways offer additional advanced MQTT features including support for Amazon AWS Things and Device Shadows, as well as support for the Sparkplug B payload specification.

AWS IoT

Overview of AWS IoT

AWS (Amazon Web Service) is a cloud platform widely used in the market, and AWS IoT (Internet of Things) supports the MQTT protocol. EasyBuilder Pro now supports AWS IoT service. Apart from using AWS IoT as a broker in the publish-subscribe mode, users can also create Amazon Things and Device Shadows offered by AWS IoT to make the most of MQTT.

Notes on AWS IIOT:

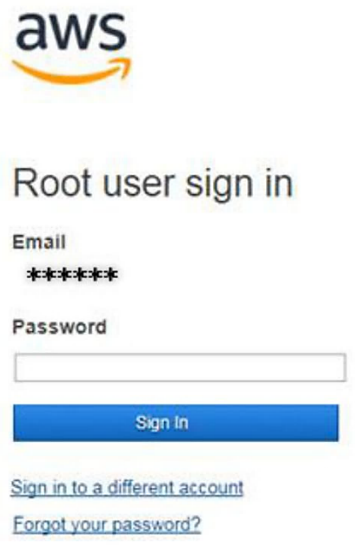
1. The maximum number of layers in a topic is 8 (iot-2/type equals to 2 layers).
2. Authentication via username and password in the General tab is not supported, you must use TLS/SSL.
3. AWS supports only QoS 0 and QoS 1.
4. Retaining the latest message in the MQTT broker is not supported.

This section goes through the steps to use AWS IoT and create a Thing and Shadow in EasyBuilder Pro.

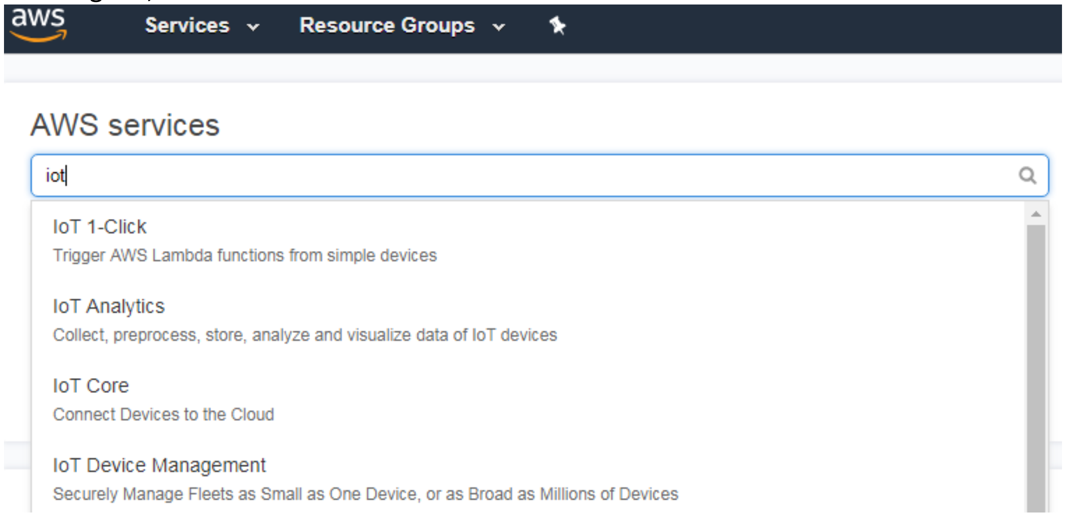
Hosting an MQTT Broker

AWS is a cloud platform; therefore, the settings are all configured on the web. You must sign up on the Amazon website before hosting an MQTT Broker. Amazon adds features to AWS frequently. The screenshots below may not be entirely accurate at the time of reading, and feature names may have changed.

1. Visit Amazon Web Services website at: https://aws.amazon.com
2. Sign up and sign in. Amazon offers a one year free evaluation trial for first time users.



3. After sign in, browse for IoT Core.



AWS services

iot

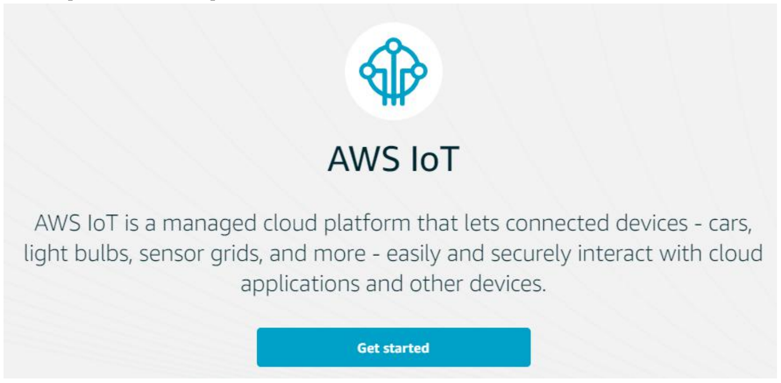
IoT 1-Click
Trigger AWS Lambda functions from simple devices

IoT Analytics
Collect, preprocess, store, analyze and visualize data of IoT devices

IoT Core
Connect Devices to the Cloud

IoT Device Management
Securely Manage Fleets as Small as One Device, or as Broad as Millions of Devices

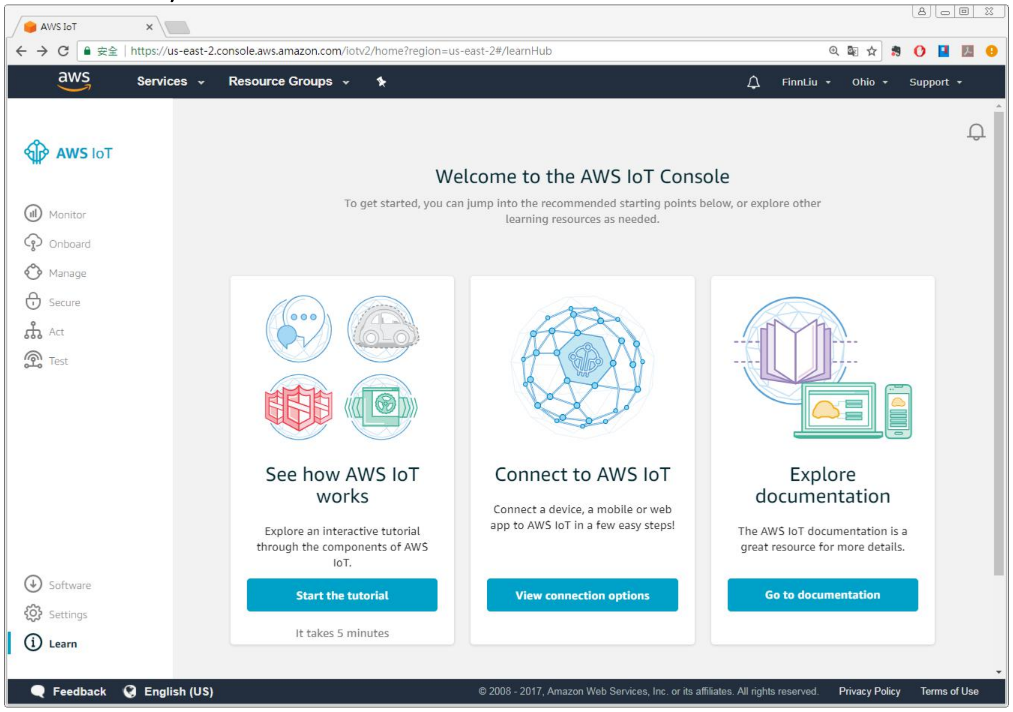
4. Click [Get started] to enter user interface.



AWS IoT

AWS IoT is a managed cloud platform that lets connected devices - cars, light bulbs, sensor grids, and more - easily and securely interact with cloud applications and other devices.

5. Create a Policy and a Certificate.



Welcome to the AWS IoT Console

To get started, you can jump into the recommended starting points below, or explore other learning resources as needed.

See how AWS IoT works

Explore an interactive tutorial through the components of AWS IoT.

It takes 5 minutes

Connect to AWS IoT

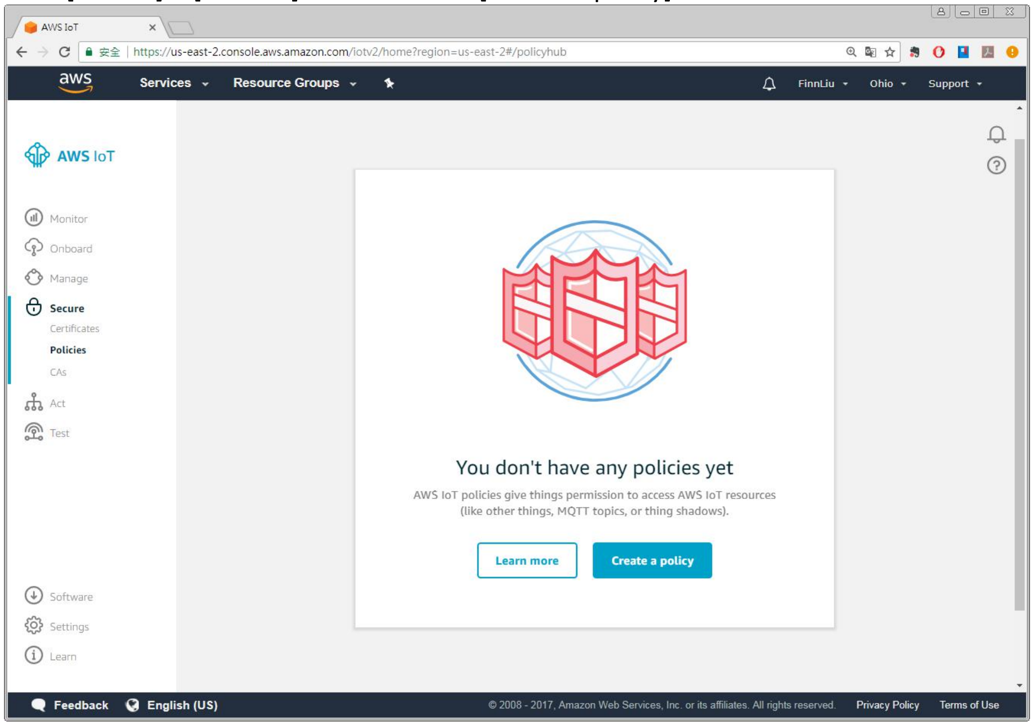
Connect a device, a mobile or web app to AWS IoT in a few easy steps!

Explore documentation

The AWS IoT documentation is a great resource for more details.

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

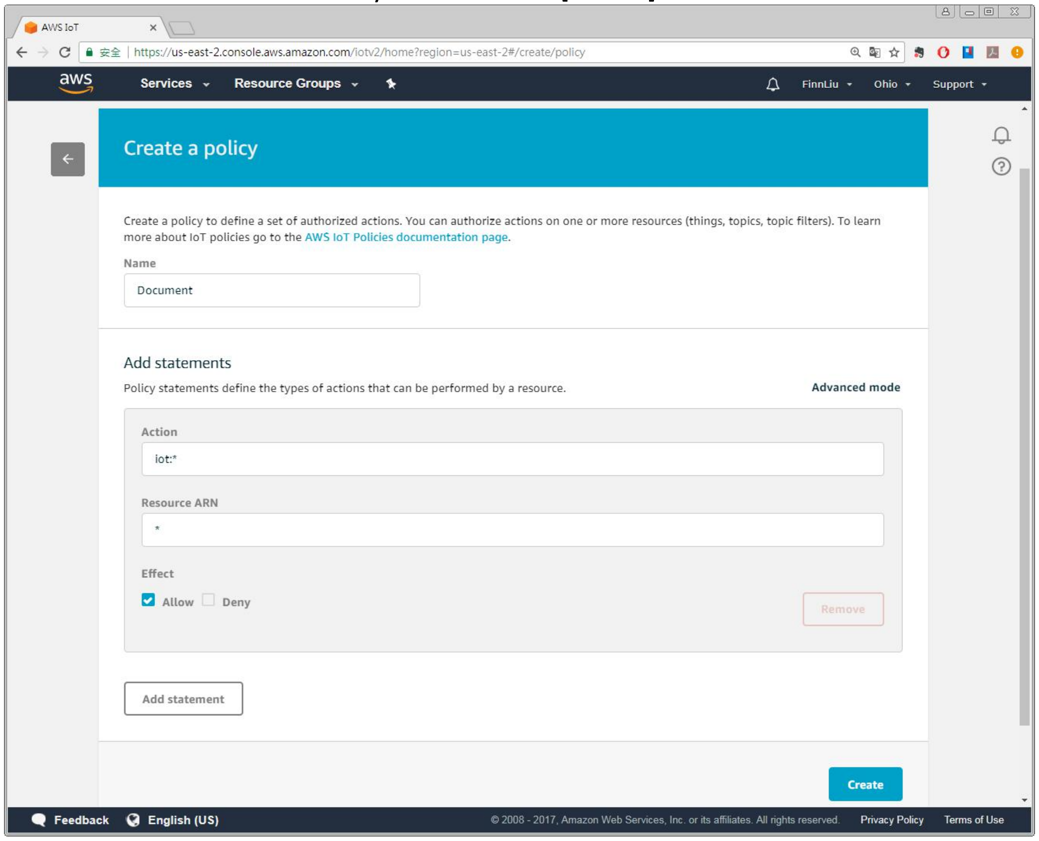
6. Click [Secure] » [Policies] and then click [Create a policy].



You don't have any policies yet

AWS IoT policies give things permission to access AWS IoT resources (like other things, MQTT topics, or thing shadows).

7. This page is for defining actions that can be performed by a resource. You may use the settings in the screenshot below or set your own. Click [Create] when finish.


AWS IoT
https://us-east-2.console.aws.amazon.com/iotv2/home?region=us-east-2#/create/policy
Services
Resource Groups
FinnLiu
Ohio
Support
Advanced mode
Remove
Add statement
Create
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the AWS IoT Policies documentation page.

Name: Document

Add statements

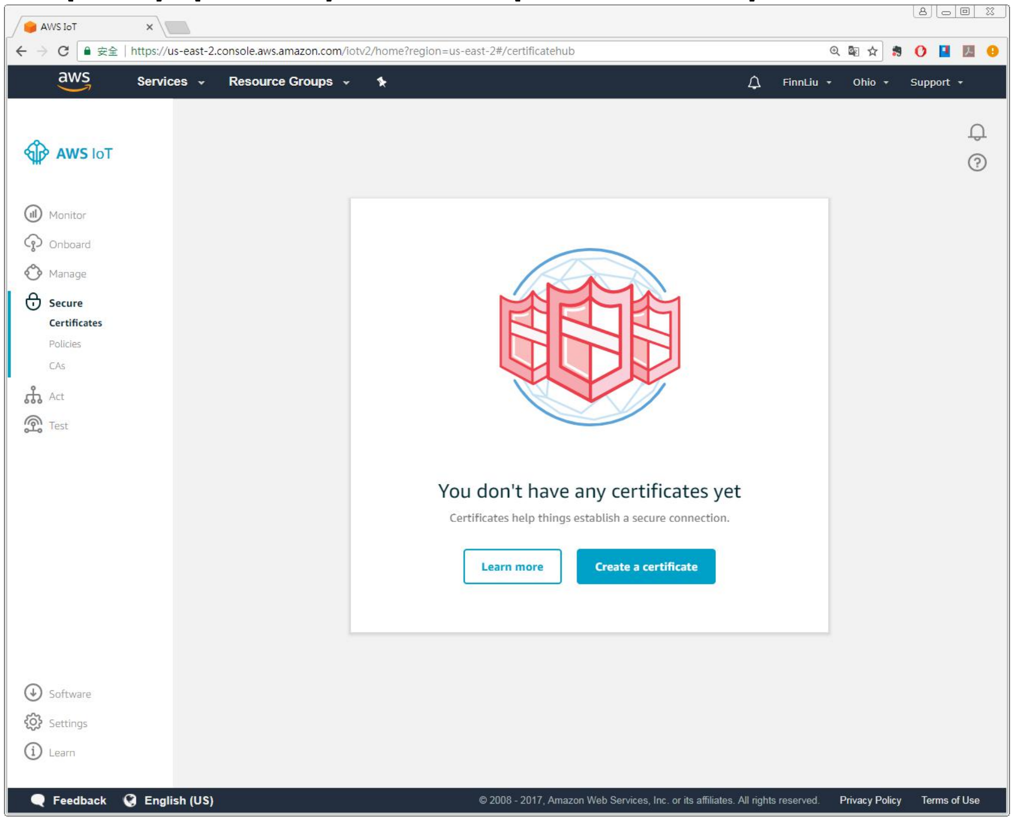
Policy statements define the types of actions that can be performed by a resource.

Action: iot:*

Resource ARN: *

Effect: [✓ Allow] Deny

8. Click [Secure] » [Certificate] and then click [Create a certificate].

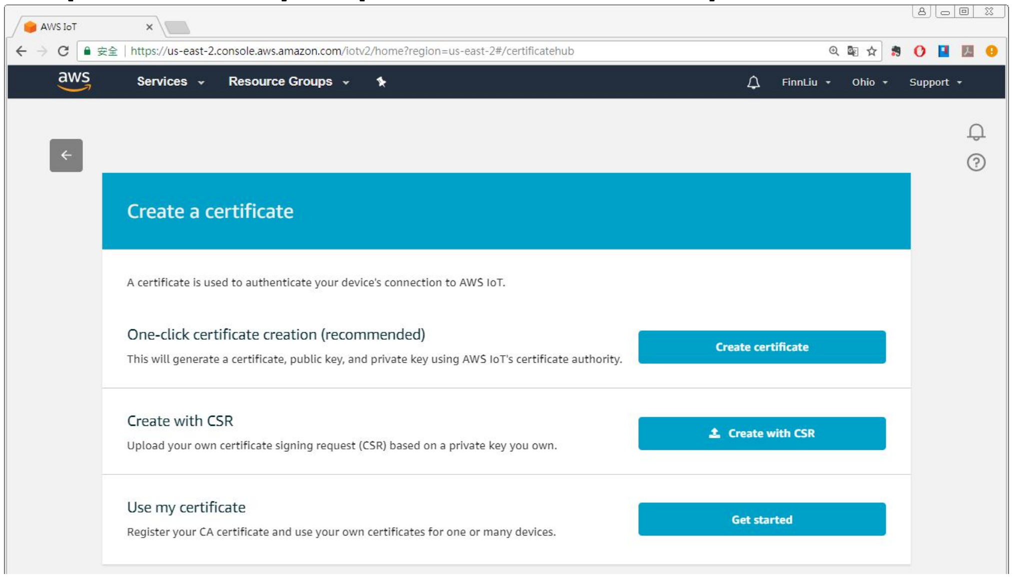


You don't have any certificates yet

Certificates help things establish a secure connection.

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

9. Click [Create certificate] near [One-click certificate creation].



Create a certificate

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)

This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

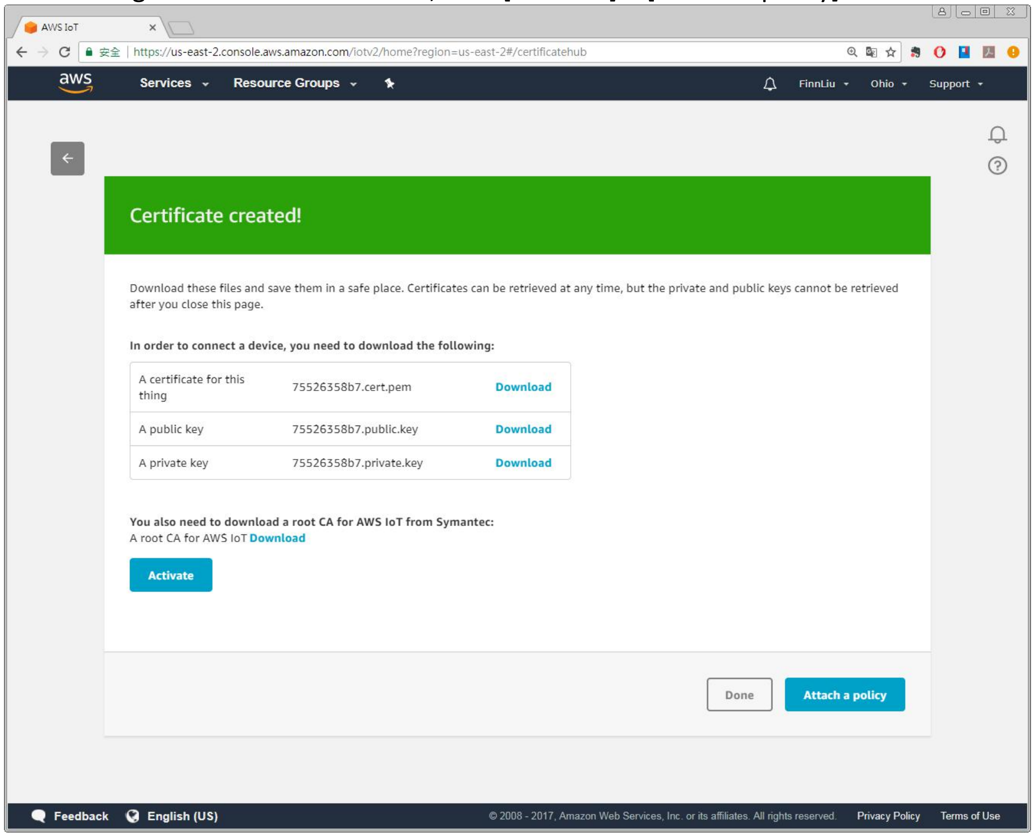
Create with CSR

Upload your own certificate signing request (CSR) based on a private key you own.

Use my certificate

Register your CA certificate and use your own certificates for one or many devices.

10. After saving the four download files, click [Activate] » [Attach a policy].



Certificate created!

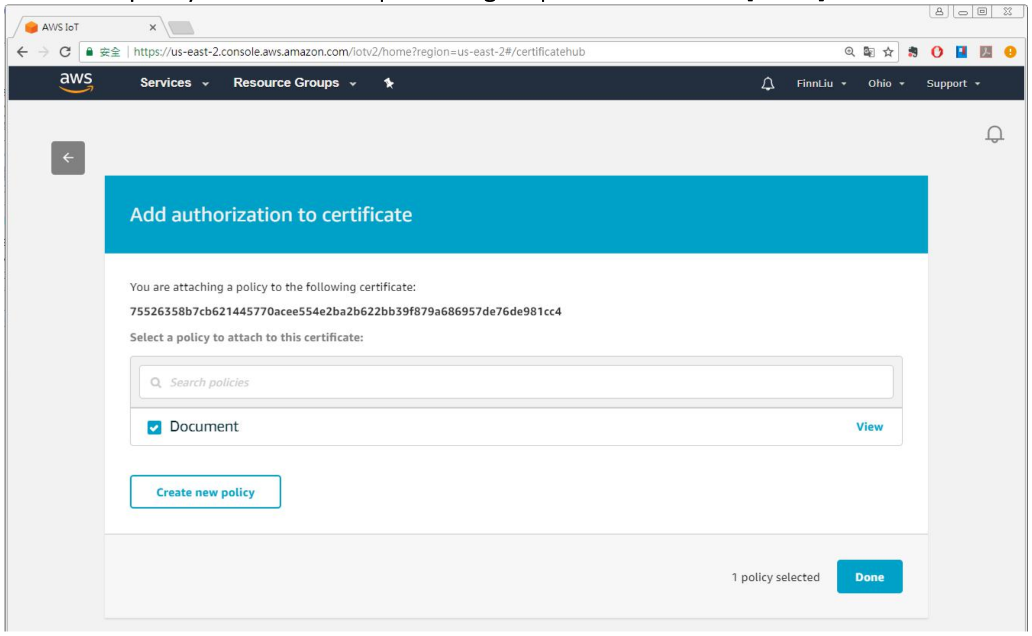
Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	75526358b7.cert.pem	Download
A public key	75526358b7.public.key	Download
A private key	75526358b7.private.key	Download

You also need to download a root CA for AWS IoT from Symantec:
A root CA for AWS IoT **Download**

11. Select the policy created in the preceding steps and then click [Done].

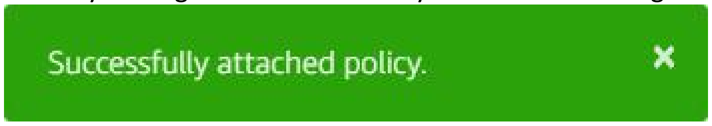


Add authorization to certificate

You are attaching a policy to the following certificate:
75526358b7cb621445770acee554e2ba2b622bb39f879a686957de76de981cc4

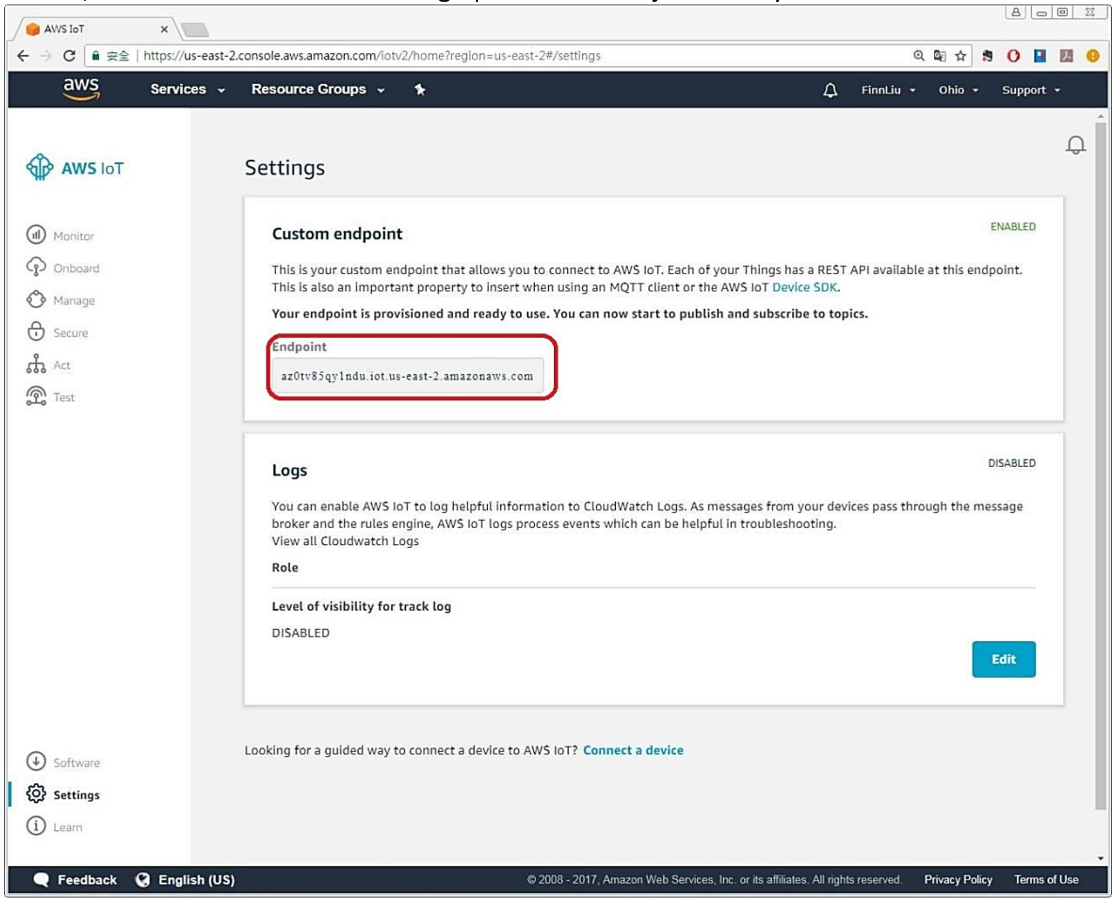
Select a policy to attach to this certificate:

12. Security setting is done successfully when the following box shows.



Successfully attached policy.

13. Click [Settings]. The URL marked in the red frame below is the domain name of your AWS IoT broker, and will be used when setting up the MQTT object in EasyBuilder Pro.


AWS IoT
https://us-east-2.console.aws.amazon.com/iotv2/home?region=us-east-2#/settings
aws
Services
Resource Groups
FinnLiu
Ohio
Support
AWS IoT
Monitor
Onboard
Manage
Secure
Act
Test
Software
Settings
Learn
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Settings

Custom endpoint

ENABLED

This is your custom endpoint that allows you to connect to AWS IoT. Each of your Things has a REST API available at this endpoint. This is also an important property to insert when using an MQTT client or the AWS IoT Device SDK.

Your endpoint is provisioned and ready to use. You can now start to publish and subscribe to topics.

Endpoint: az0tv85qy1ndu.iot.us-east-2.amazonaws.com

Logs

DISABLED

You can enable AWS IoT to log helpful information to CloudWatch Logs. As messages from your devices pass through the message broker and the rules engine, AWS IoT logs process events which can be helpful in troubleshooting.
View all Cloudwatch Logs

Role

Level of visibility for track log
DISABLED

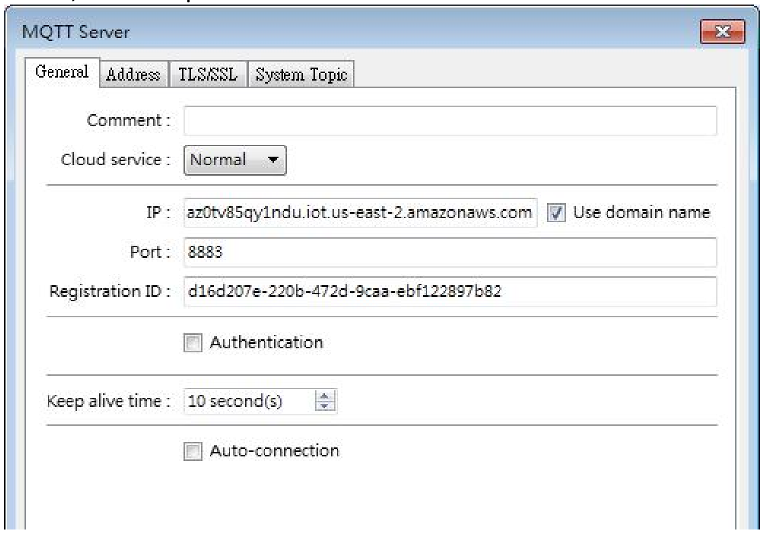
Edit

Looking for a guided way to connect a device to AWS IoT? Connect a device

EasyBuilder Pro Settings

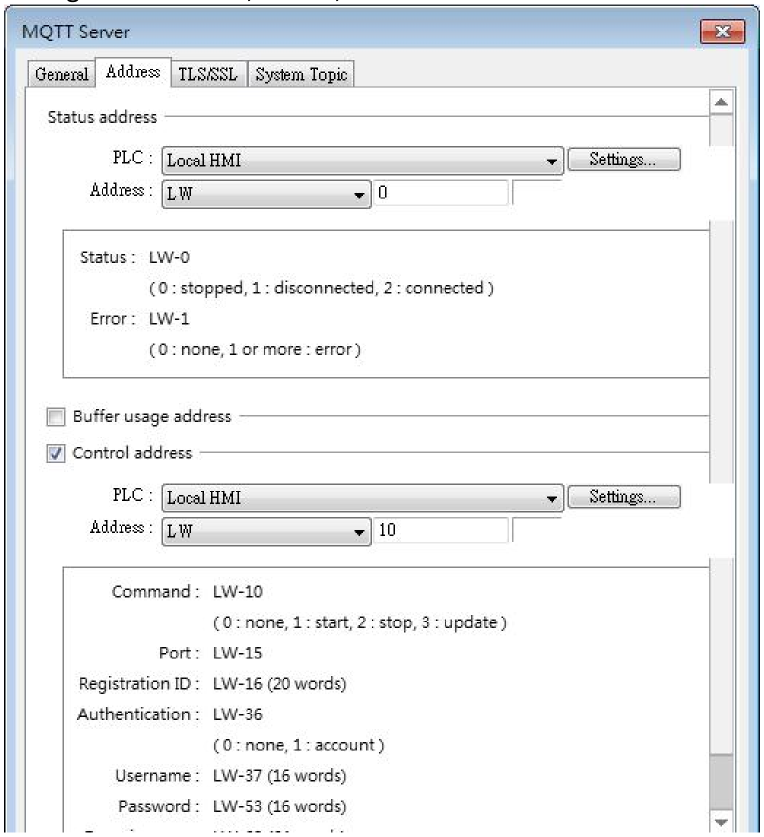
After hosting an AWS MQTT Broker, launch EasyBuilder Pro and open the MQTT settings window.

1. In the General tab, select [Normal] as cloud service to use publish-subscribe mode, or select [AWS IoT] to use Thing/Device Shadow mode. Use the URL obtained above for the domain name, and use port 8883.



Comment :
Cloud service : Normal
IP : az0tv85qy1ndu.iot.us-east-2.amazonaws.com [✓] Use domain name
Port : 8883
Registration ID : d16d207e-220b-472d-9caa-ebf122897b82
[ ] Authentication
Keep alive time : 10 second(s)
[ ] Auto-connection

2. Configure the Status, Buffer, and Control addresses as normal.



Status address

PLC : Local HMI
Address : LW 0

Status : LW-0
(0 : stopped, 1 : disconnected, 2 : connected)
Error : LW-1
(0 : none, 1 or more : error)

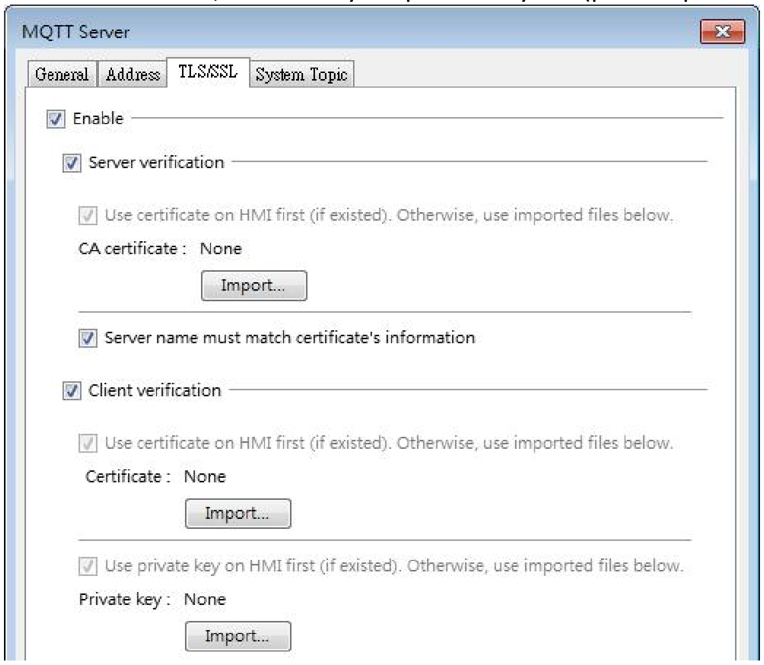
☐ Buffer usage address

☑ Control address

PLC : Local HMI
Address : LW 10

Command : LW-10
(0 : none, 1 : start, 2 : stop, 3 : update)
Port : LW-15
Registration ID : LW-16 (20 words)
Authentication : LW-36
(0 : none, 1 : account)
Username : LW-37 (16 words)
Password : LW-53 (16 words)

3. In the TLS/SSL tab, import the files generated when creating the AWS certificate.
Server verification, CA certificate: Import a .pem file.
Client verification, Certificate: Import a .crt file. (certificate.pem.crt)
Client verification, Private key: Import a .key file (private.pem.key)



☑ Enable

☑ Server verification

☑ Use certificate on HMI first (if existed). Otherwise, use imported files below.
CA certificate : None

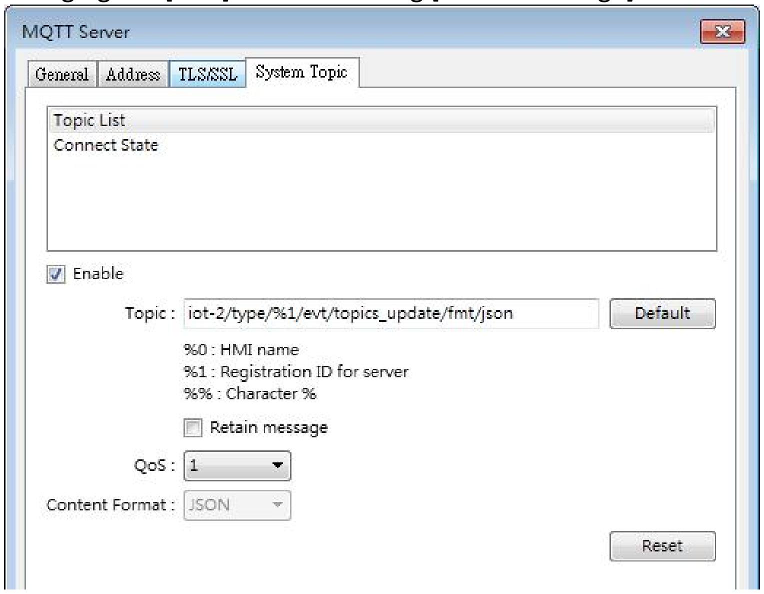
☑ Server name must match certificate's information

☑ Client verification

☑ Use certificate on HMI first (if existed). Otherwise, use imported files below.
Certificate : None

☑ Use private key on HMI first (if existed). Otherwise, use imported files below.
Private key : None

4. The System Topic tab includes Topic List and Connection State topics that cMT device will automatically send once it connects to broker. Be sure to conform to AWS restriction by changing the [QoS] and unchecking [Retain message] for each topic.



[✓] Enable

Topic : iot-2/type/%1/evt/topics_update/fmt/json

%0 : HMI name
%1 : Registration ID for server
%% : Character %

[ ] Retain message

QoS : 1

Content Format : JSON

Restrictions of using AWS IoT as MQTT Broker:

a. Only QoS 0 and QoS 1 are available.

b. Retain message is not supported.

c. The maximum number of layers is 8.

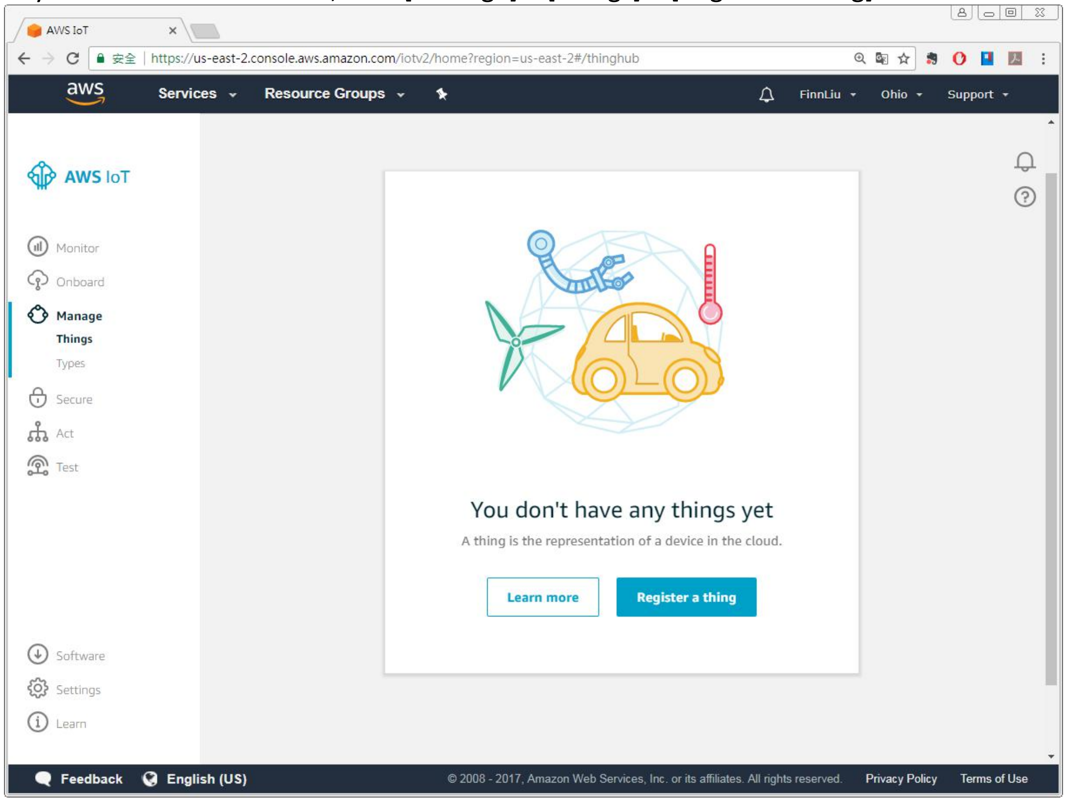
5. If using AWS as a [Normal] cloud service broker, configure the Topic Publisher and Topic Subscriber as described earlier in this manual. If using Things/Device Shadows ([AWS IoT] selected for cloud service on the General tab), see the next section for Publish/Subscribe settings.

Thing and Device Shadow

With AWS IoT, Publisher->Broker-> Subscribe is no longer the only path that data is accessed over MQTT. By introducing the Device Shadow service, a Thing (a device, app…etc) can interact with cloud applications and other devices connected to AWS IoT. The device shadow (sometimes referred to as a thing shadow) is used as a communication layer between your mobile/cloud application and the devices (HMIs) connected to AWS IoT.

The shadow is a persistent, virtual representation of the HMI or Gateway. Because it always has a point-in-time view of the state of your device, it’s easy to write applications that interact with your devices through device shadows. A Shadow can be maintained for each Thing (device/HMI) connected to AWS IoT. The Shadow can be used to get/set the state of a Thing over MQTT, regardless of whether the Thing is connected to the Internet.

1. In your AWS IoT Dashboard, click [Manage] » [Things] » [Register a thing].

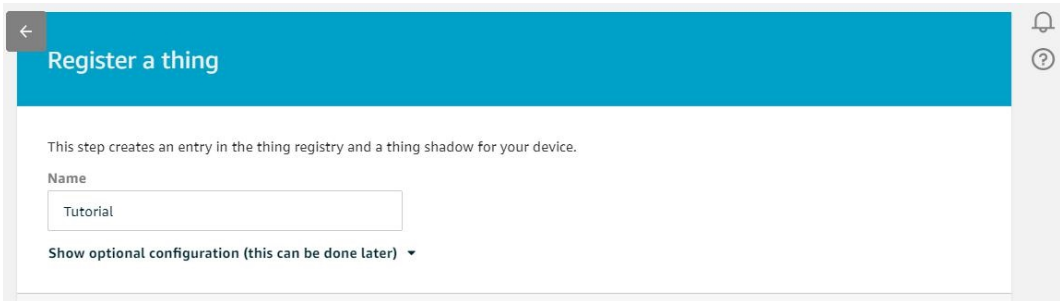


You don't have any things yet

A thing is the representation of a device in the cloud.

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

2. Select [Create a single thing] and enter the name for the Thing. The additional Thing options can be ignored. Click [Next].

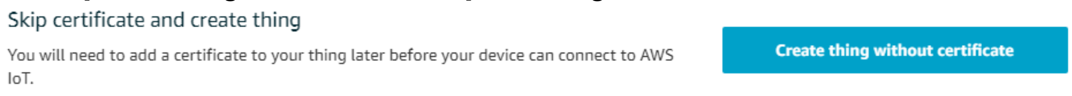


Register a thing

This step creates an entry in the thing registry and a thing shadow for your device.

Name: Tutorial

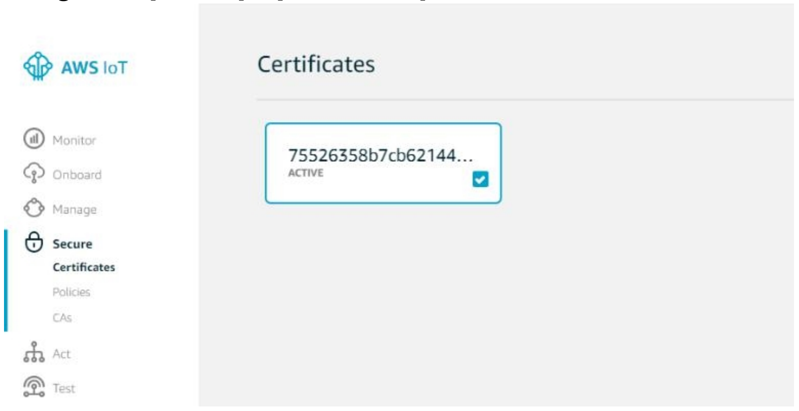
3. Select [Create thing without certificate]. The Thing is created.



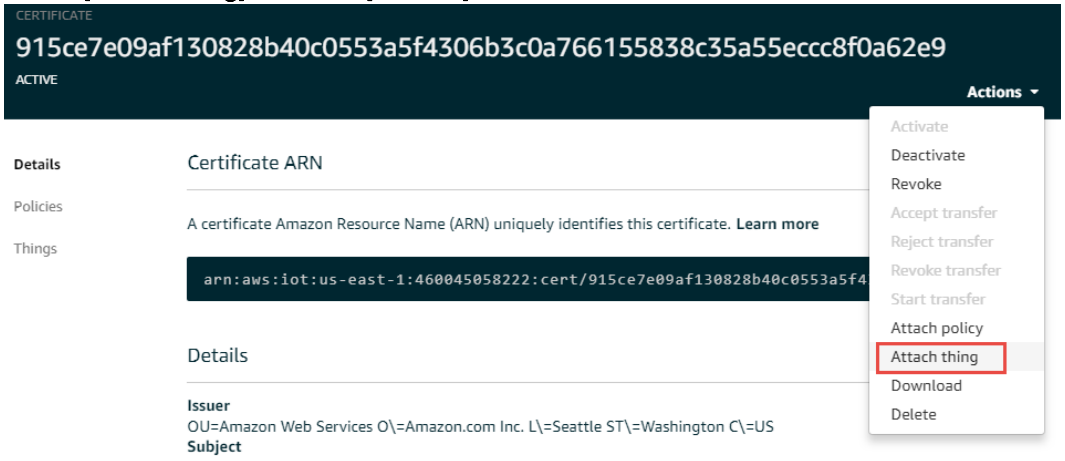
Skip certificate and create thing

You will need to add a certificate to your thing later before your device can connect to AWS IoT.

4. Navigate to [Secure] » [Certificates] and select the certificate created previously.



5. Select [Attach thing] from the [Actions] menu.

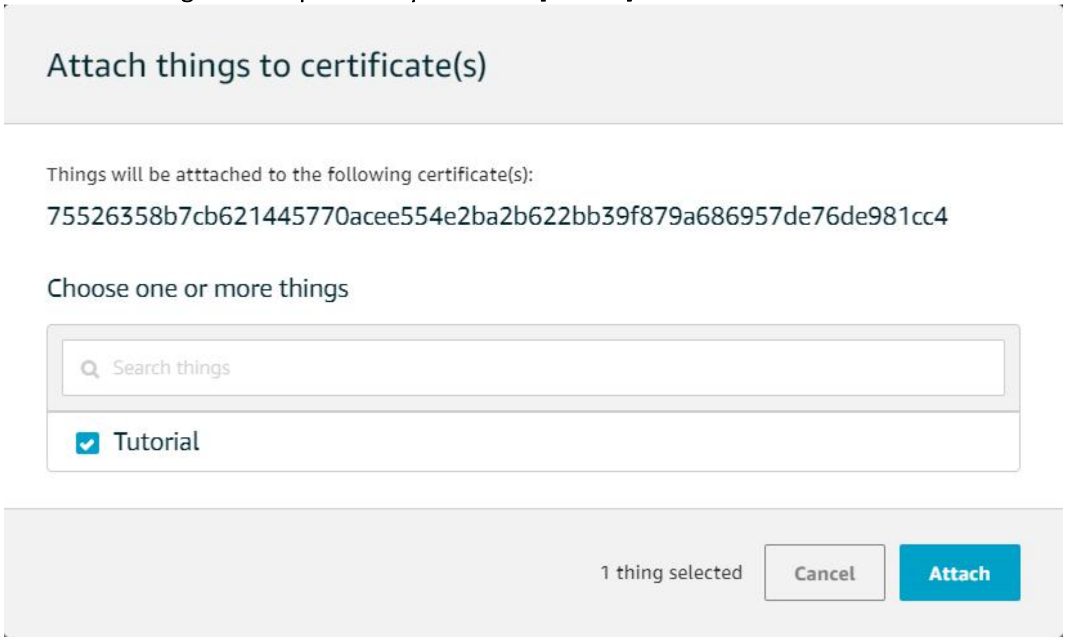


A certificate Amazon Resource Name (ARN) uniquely identifies this certificate. **Learn more**

```
arn:aws:iot:us-east-1:460045058222:cert/915ce7e09af130828b40c0553a5f4
```

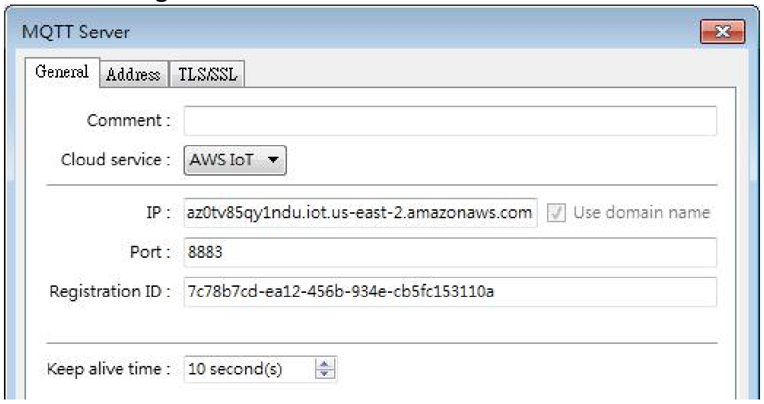
Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject

6. Select the Thing created previously and click [Attach] to finish.



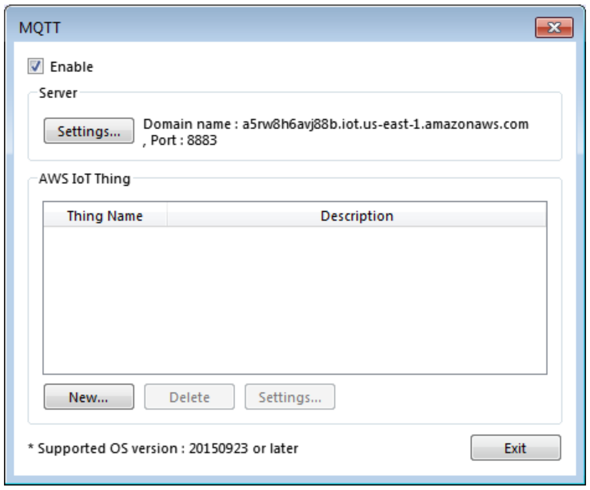
Things will be atttached to the following certificate(s):
75526358b7cb621445770acee554e2ba2b622bb39f879a686957de76de981cc4

7. In EasyBuilder Pro, make sure that [AWS IoT] is selected as the cloud service option in the MQTT Server settings window.



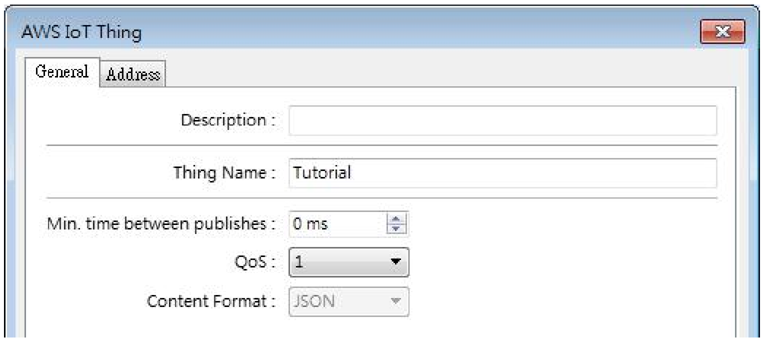
Comment :
Cloud service : AWS IoT
IP : az0tv85qy1ndu.iot.us-east-2.amazonaws.com [✓] Use domain name
Port : 8883
Registration ID : 7c78b7cd-ea12-456b-934e-cb5fc153110a
Keep alive time : 10 second(s)

8. Click new to add a device.



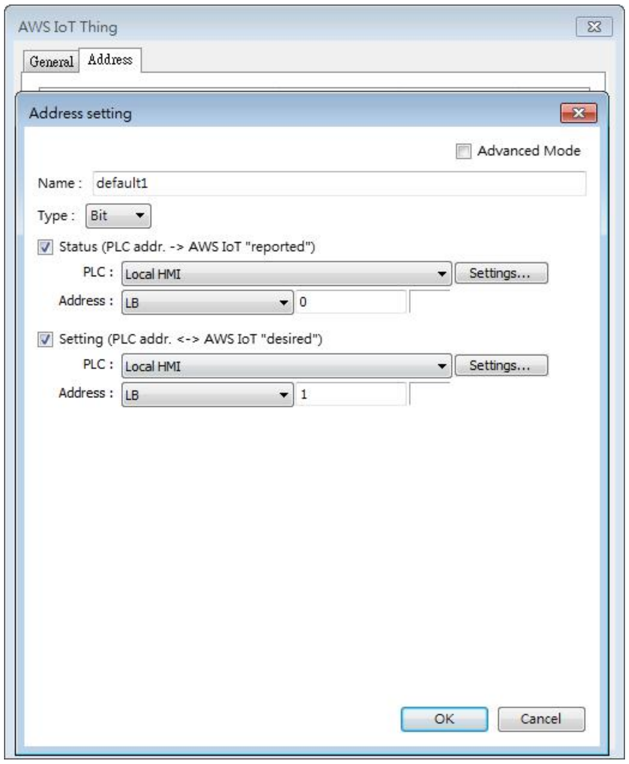
[✓] Enable
Domain name : a5rw8h6avj88b.iot.us-east-1.amazonaws.com , Port : 8883
* Supported OS version : 20150923 or later

9. Enter the Thing name and set a minimum time between publishes. Only QoS 0 and 1 are available.



Description :
Thing Name : Tutorial
Min. time between publishes : 0 ms
QoS : 1
Content Format : JSON

10. In the Address tab, click [New] and select addresses for "reported" status and "desired" setting. The arrows "->" and "<->" indicate the direction in which data is transmitted between the HMI and AWS. By default, the "reported" data is only sent from the HMI to AWS and "desired" data can be sent in both directions.



Advanced Mode [ ]

Name : default1

Type : Bit

[✓] Status (PLC addr. -> AWS IoT "reported")

PLC : Local HMI Settings...

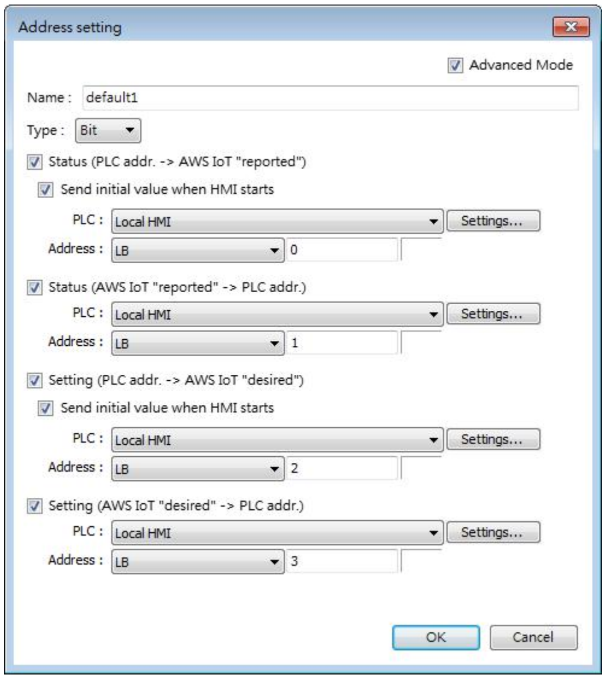
Address : LB 0

[✓] Setting (PLC addr. <-> AWS IoT "desired")

PLC : Local HMI Settings...

Address : LB 1

11. If the Advanced Mode checkbox is selected, the Status (reported) and Setting (desired) can use different addresses for each data transmission direction to allow more granular control from AWS.



[✓] Advanced Mode

Name : default1

Type : Bit

[✓] Status (PLC addr. -> AWS IoT "reported")

[✓] Send initial value when HMI starts

PLC : Local HMI Settings...

Address : LB 0

[✓] Status (AWS IoT "reported" -> PLC addr.)

PLC : Local HMI Settings...

Address : LB 1

[✓] Setting (PLC addr. -> AWS IoT "desired")

[✓] Send initial value when HMI starts

PLC : Local HMI Settings...

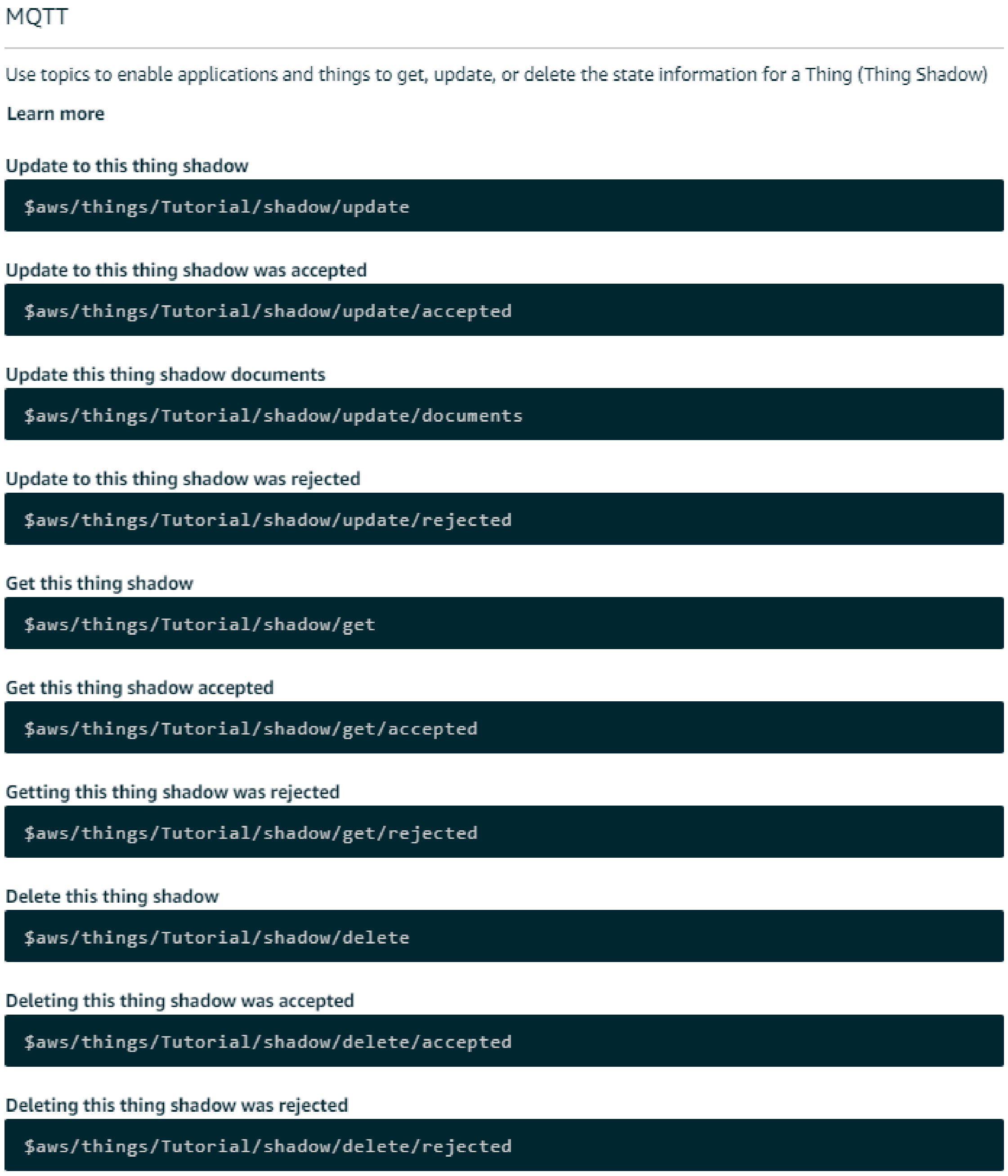
Address : LB 2

[✓] Setting (AWS IoT "desired" -> PLC addr.)

PLC : Local HMI Settings...

Address : LB 3

12. You should now be able to subscribe and publish to your Thing with the following topics:



MQTT

Use topics to enable applications and things to get, update, or delete the state information for a Thing (Thing Shadow)

Learn more

Update to this thing shadow

```
$aws/things/Tutorial/shadow/update
```

Update to this thing shadow was accepted

```
$aws/things/Tutorial/shadow/update/accepted
```

Update this thing shadow documents

```
$aws/things/Tutorial/shadow/update/documents
```

Update to this thing shadow was rejected

```
$aws/things/Tutorial/shadow/update/rejected
```

Get this thing shadow

```
$aws/things/Tutorial/shadow/get
```

Get this thing shadow accepted

```
$aws/things/Tutorial/shadow/get/accepted
```

Getting this thing shadow was rejected

```
$aws/things/Tutorial/shadow/get/rejected
```

Delete this thing shadow

```
$aws/things/Tutorial/shadow/delete
```

Deleting this thing shadow was accepted

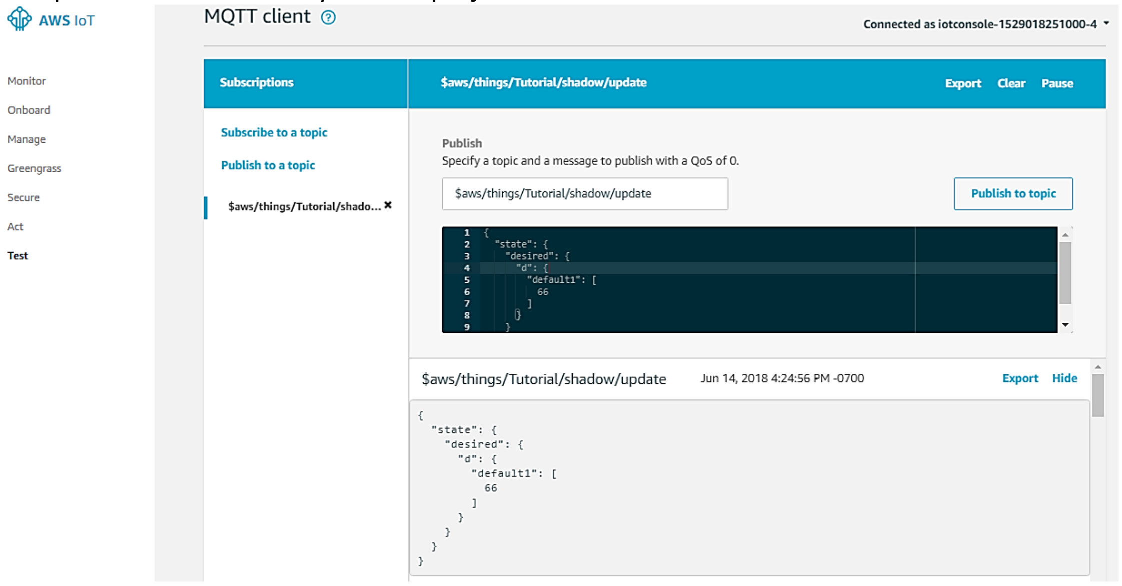
```
$aws/things/Tutorial/shadow/delete/accepted
```

Deleting this thing shadow was rejected

```
$aws/things/Tutorial/shadow/delete/rejected
```

This information is found in the [Interact] section of the Thing Dashboard.

13. From the [Test] section of the AWS IoT Dashboard, you should be able to subscribe and publish to topics and interact with your test project.


AWS IoT
Monitor
Onboard
Manage
Greengrass
Secure
Act
Test
MQTT client
Connected as iotconsole-1529018251000-4
Subscriptions
Subscribe to a topic
Publish to a topic
$aws/things/Tutorial/shado...
$aws/things/Tutorial/shadow/update
Export
Clear
Pause
Publish
Specify a topic and a message to publish with a QoS of 0.
$aws/things/Tutorial/shadow/update
Publish to topic
$aws/things/Tutorial/shadow/update
Jun 14, 2018 4:24:56 PM -0700
Export
Hide

```
{
  "state": {
    "desired": {
      "d": {
        "default1": [
          66
        ]
      }
    }
  }
}
```

Sparkplug B

Overview of Sparkplug B

Sparkplug is a specification for MQTT enabled devices and applications to send and receive messages in a stateful way. Sparkplug provides a mechanism for ensuring that remote device or application data is current and valid by using device lifecycle messages such as the required birth and last will & testament messages that must be sent to ensure the device lifecycle state and data integrity.

It also is utilized in connecting with the Inductive Automation Ignition Platform utilizing the Cirrus Link MQTT modules. The Sparkplug specification provides the necessary details for any MQTT enabled device to connect to MQTT servers and integrate with zero configuration into Ignition via the Cirrus Link MQTT Engine Module or other Sparkplug supported applications.

Sparkplug B features include:

- Complex data types using templates
- Datasets
- Richer metrics with the ability to add property metadata for each metric
- Metric alias support to maintain rich metric naming while keeping bandwidth usage to a minimum
- Historical data
- File data

The Sparkplug B specification can be found here. Topics are formatted as:

spBv1.0/group_id/message_type/edge_node_id/[device_id]

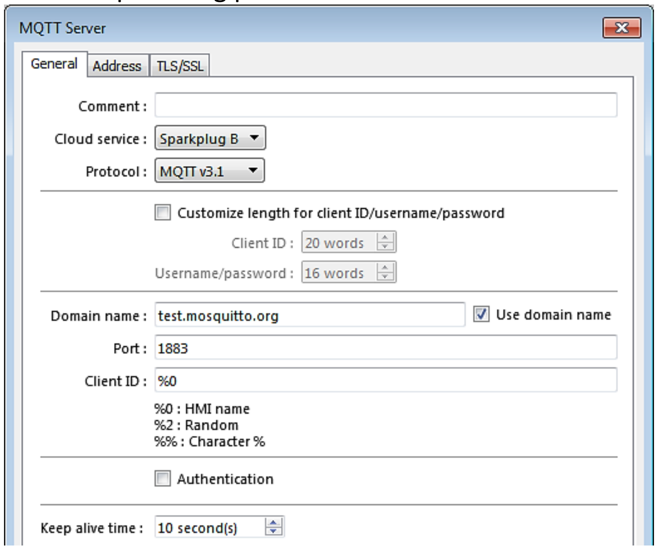
The following *message_type* elements are defined for the Sparkplug Topic Namespace:

- NBIRTH – Birth certificate for MQTT EoN nodes.
- NDEATH – Death certificate for MQTT EoN nodes.
- DBIRTH – Birth certificate for Devices.
- DDEATH – Death certificate for Devices.
- NDATA – Node data message.
- DDATA – Device data message.
- NCMD – Node command message.
- DCMD – Device command message.
- STATE – Critical application state message.

EasyBuilder Pro Settings

To configure a Sparkplug B payload, launch EasyBuilder Pro and open the MQTT settings window.

1. In the General tab, select [Sparkplug B] for the cloud service. In this example, we will use the test.mosquitto.org public broker.


MQTT Server

General | Address | TLS/SSL

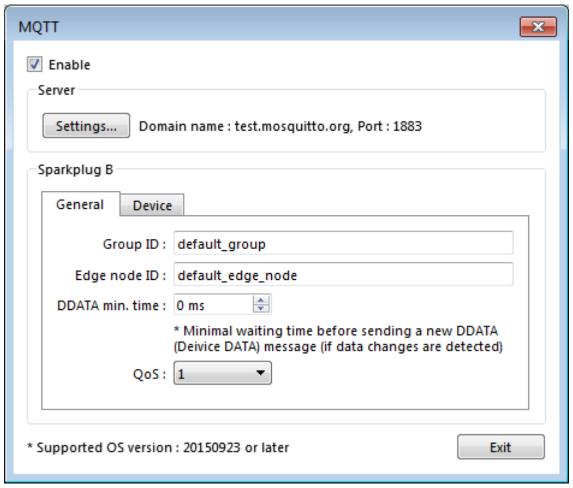
Comment :
Cloud service : Sparkplug B
Protocol : MQTT v3.1

☐ Customize length for client ID/username/password
Client ID : 20 words
Username/password : 16 words

Domain name : test.mosquitto.org ☑ Use domain name
Port : 1883
Client ID : %0
%0 : HMI name
%2 : Random
%% : Character %

☐ Authentication

Keep alive time : 10 second(s)

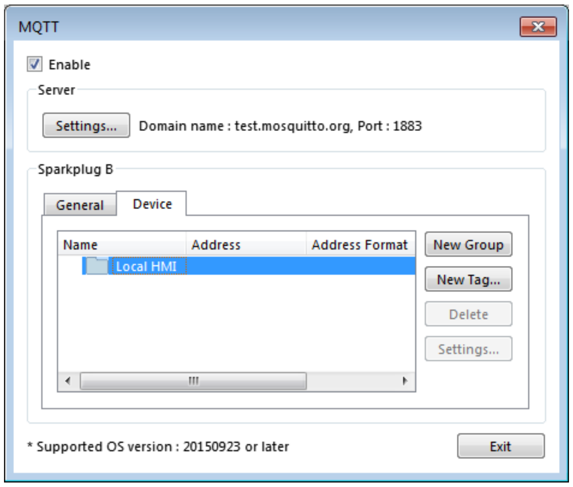

MQTT

☑ Enable

Server
Settings... Domain name : test.mosquitto.org, Port : 1883

Sparkplug B
General | Device
Group ID : default_group
Edge node ID : default_edge_node
DDATA min. time : 0 ms
* Minimal waiting time before sending a new DDATA (Deivice DATA) message (if data changes are detected)
QoS : 1

* Supported OS version : 20150923 or later
Exit


MQTT

☑ Enable

Server
Settings... Domain name : test.mosquitto.org, Port : 1883

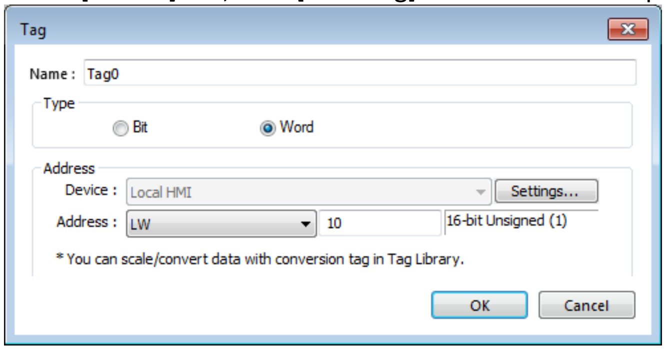
Sparkplug B
General | Device

Name	Address	Address Format
Local HMI		

New Group
New Tag...
Delete
Settings...

* Supported OS version : 20150923 or later
Exit

2. In the [Device] tab, click [New Tag] to add data to the payload.

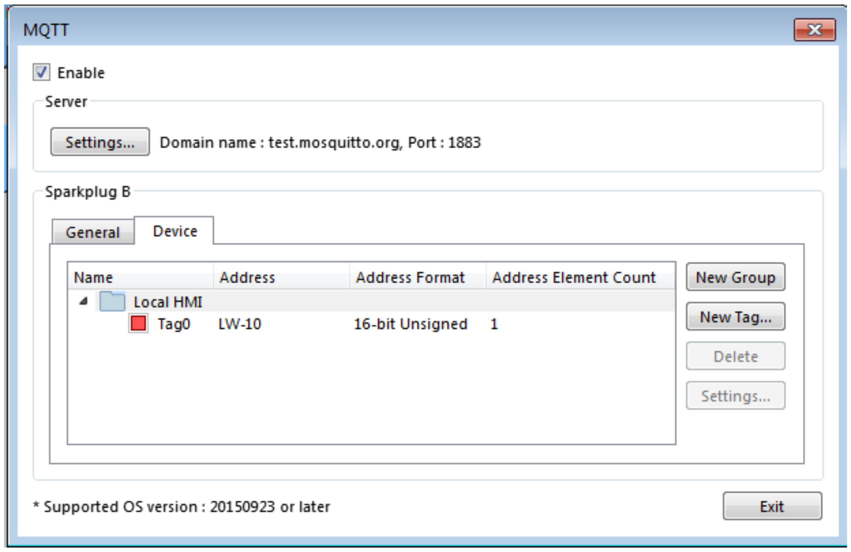

Tag

Name : Tag0

Type
Bit [✓ Word]

Address
Device : Local HMI Settings...
Address : LW 10 16-bit Unsigned (1)
* You can scale/convert data with conversion tag in Tag Library.

OK Cancel



MQTT

☑ Enable

Server

Settings... Domain name : test.mosquitto.org, Port : 1883

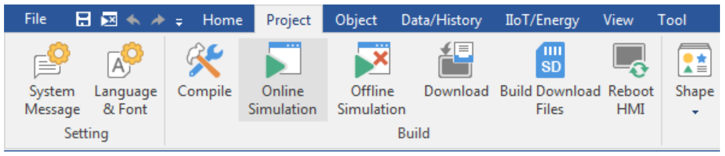
Sparkplug B

General | Device

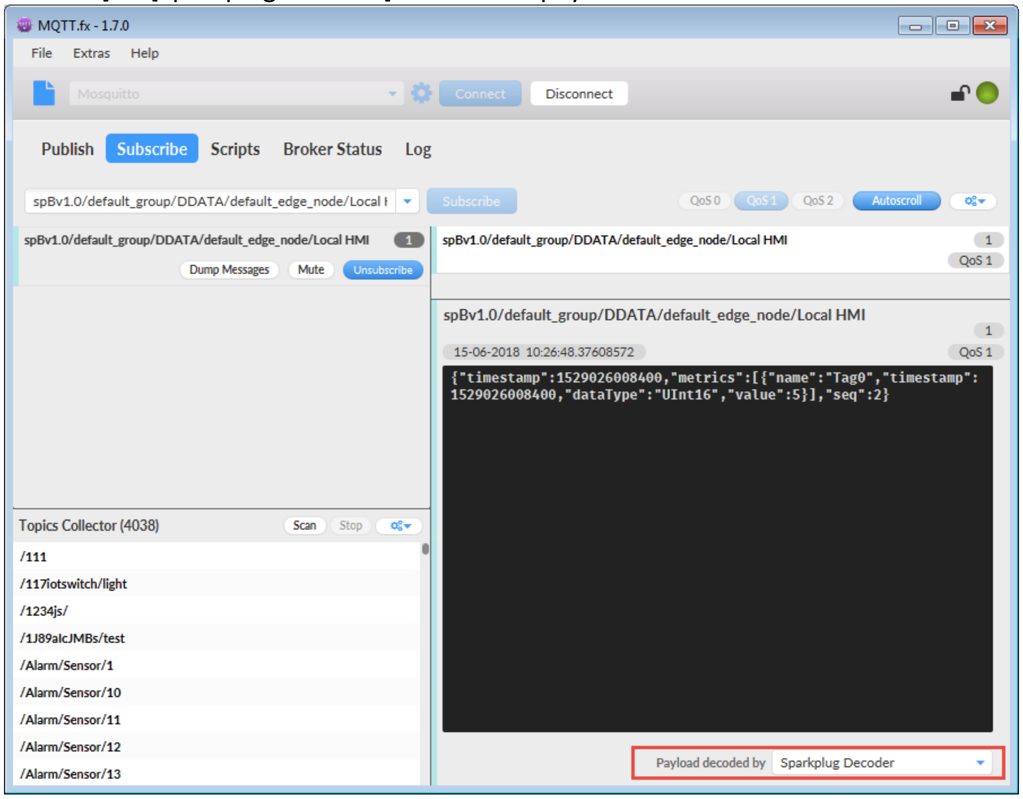
Name	Address	Address Format	Address Element Count
Local HMI			
🟥 Tag0	LW-10	16-bit Unsigned	1

* Supported OS version : 20150923 or later

3. Add a way to change the specified tag to a screen and run the project in [Online Simulation] mode.



4. Open the MQTT.fx client program and connect to test.mosquitto.org. The preceding setup results in data being updated on topic [spBv1.0/default_group/DDATA/default_edge_node/Local HMI]. Be sure to set the [Payload decoder] to [Sparkplug Decoder] to view the payload in a readable format.



spBv1.0/default_group/DDATA/default_edge_node/Local HMI

spBv1.0/default_group/DDATA/default_edge_node/Local HMI 1

spBv1.0/default_group/DDATA/default_edge_node/Local HMI 1 QoS 1

spBv1.0/default_group/DDATA/default_edge_node/Local HMI 1

15-06-2018 10:26:48.37608572 QoS 1

```
{"timestamp":1529026008400,"metrics":[{"name":"Tag0","timestamp":
1529026008400,"dataType":"UInt16","value":5}],"seq":2}
```

Topics Collector (4038)

/111
/117iotswitch/light
/1234js/
/1J89alcJMBs/test
/Alarm/Sensor/1
/Alarm/Sensor/10
/Alarm/Sensor/11
/Alarm/Sensor/12
/Alarm/Sensor/13

Payload decoded by Sparkplug Decoder

References

EasyBuilder Pro User Manual Chapter 42 IIoT:
https://www.maplesystems.com/supportcenter/SCManuals.htm

IBM Qualities of service provided by an MQTT client:
https://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.dev.doc/q029090_.htm

The Seven Best MQTT Client Tools:
http://www.hivemq.com/blog/seven-best-mqtt-client-tools

MQTT Server Software:
Mosquitto (http://mosquitto.org/download/)
HiveMQ (http://www.hivemq.com/downloads/)
EMQ (http://emqtt.io/)

MQTT Client Software:
MQTT.fx (http://mqttfx.jensd.de/index.php/download)

Amazon Web Services website:
https://aws.amazon.com

How does AWS IoT platform work:
https://aws.amazon.com/tw/iot-platform/how-it-works/

Sparkplug B Specification:
https://s3.amazonaws.com/cirrus-link-com/Sparkplug+Topic+Namespace+and+State+ManagementV2.1+Apendix++Payload+B+format.pdf

Using Ignition MQTT Engine with Maple Systems HMIs Tech Note (how to configure Ignition to work with an Advanced HMI):
https://www.maplesystems.com/cgi-bin/beimatdongon/TechNote/09075119.pdf

Your Industrial Control Solutions Source

www.maplesystems.com

1010-1068 Rev. 00