MQTT User Guide

MQTT

This guide walks through the MQTT usage.

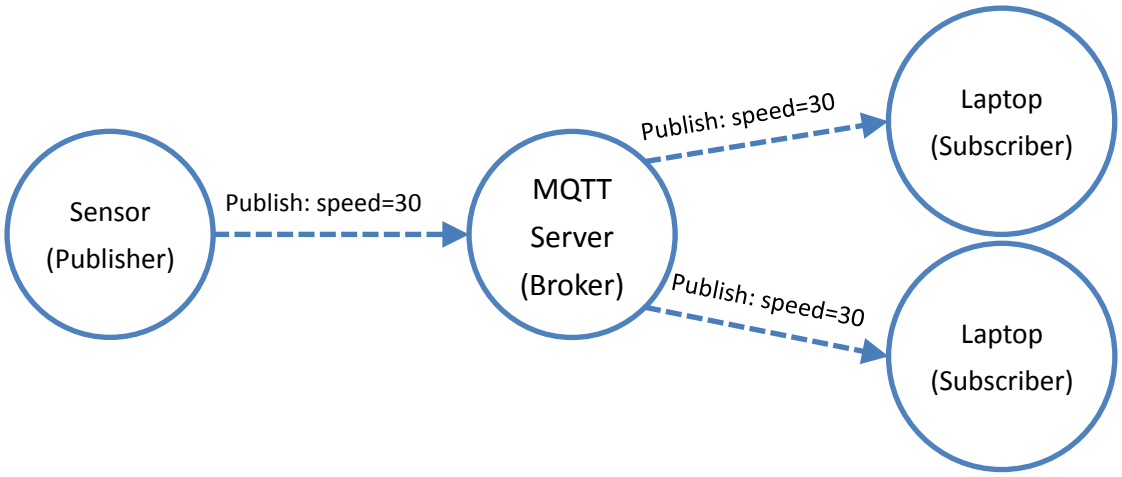
V3.00

Table of Contents

1. Overview

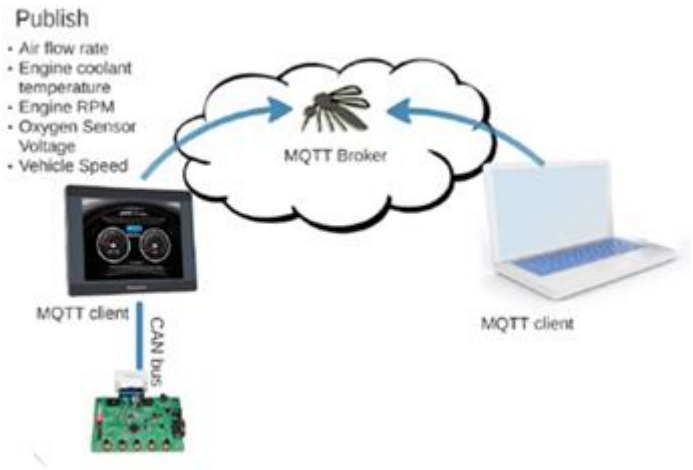
Designed to be light weight, open, and simple, MQTT is a subscriber/publisher messaging transport protocol that is considered a great solution for applications where small code footprint is required and/or network bandwidth is scarce. It is particularly suitable for continuous monitoring of sensory data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

Publisher, Subscriber, and Server (or Broker) are three important roles in MQTT communication protocol. The relationship between these roles is shown in the following figure. After the subscriber subscribes to a topic, when the Publisher publishes a message to the Server, the Server will deliver the message to the subscriber.

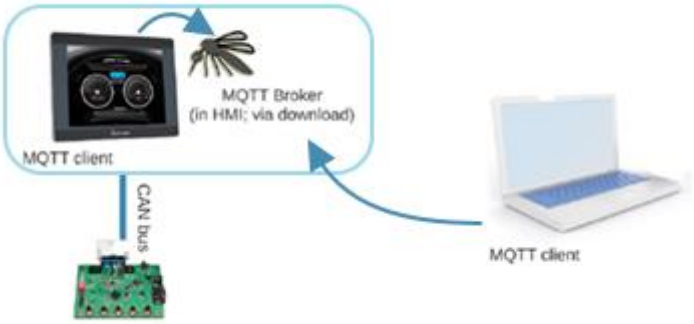


Weintek HMI and MQTT

HMI processes data from PLC and publishes messages to an MQTT Server, which will handle message delivery to the subscribers.



Alternatively, MQTT messages can be published internally to a built-in MQTT Server. That is, an external Server is not necessary; one can use an MQTT client to subscribe directly to the MQTT Server inside the HMI and receive message updates!



2. EasyBuilder Pro Settings

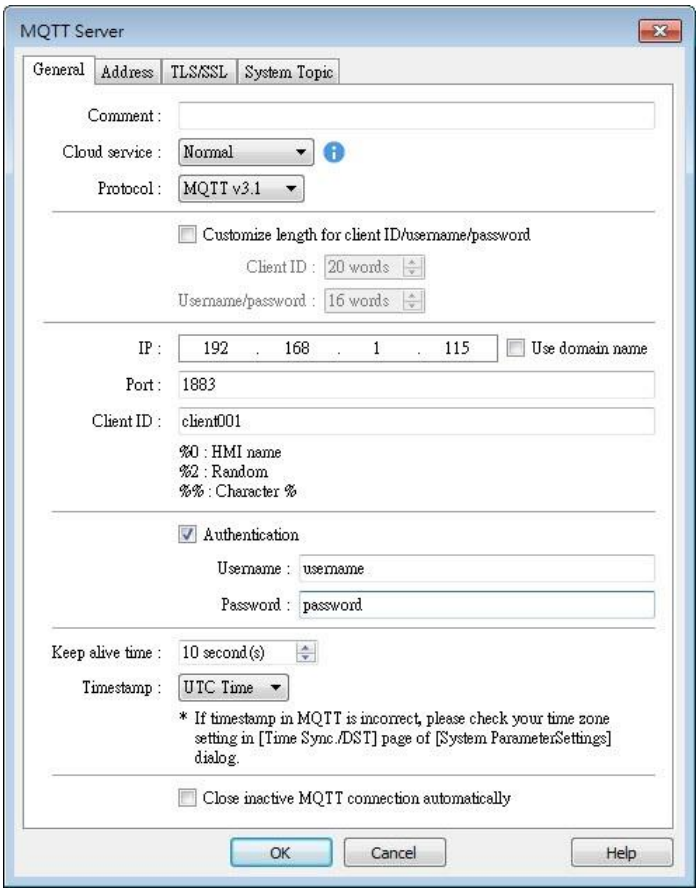
In EasyBuilder Pro, click [IIoT/Energy] » [MQTT] to setup MQTT in the project. Firstly and most importantly, MQTT server information must be entered.

Server Settings

Enter the IP address of the MQTT Server or use a domain name. When IP address 127.0.0.1 is used, the messages will be published to the built-in MQTT Server on HMI.

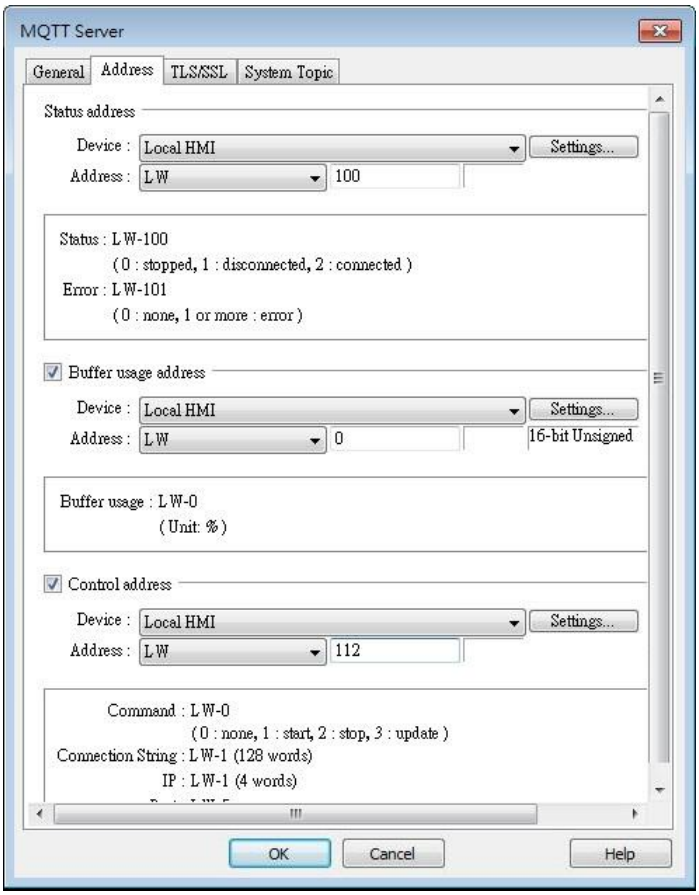
Selecting “Normal” as Cloud Service enables standard MQTT communication protocol. Other cloud services: AWS IoT, Sparkplug B, and Azure IoT Hub are extended applications (cMT Series only). For more information on the cloud services, please refer to corresponding manuals. Topics discussed in this manual are primarily for the standard MQTT communication protocol.

To manage access privileges to the server, please select [Authentication] and enter the username and password.

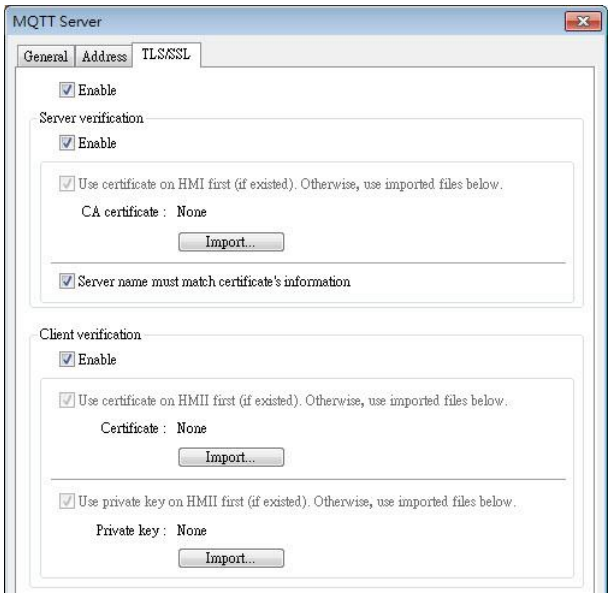


LW addresses can be designated to dynamically control MQTT or display MQTT status during HMI run time. After designating an address, its relative addresses (+1, +2,

+3…etc.) will correspond to different attributes or parameters, as shown in the following EasyBuilder Pro settings dialog box. If Status address is set to LW-100, then LW-100 shows the status and LW-101 shows the error code.
Messages that have not been sent are stored in the buffer whose maximum capacity is 10000 messages. When the buffer is full, the earliest message will be deleted.



Enabling TLS/SSL authentication opens two verification methods:
[Server verification]: Import server's certificate for client to verify whether the server is legitimate.
[Client verification]: Provide private key and certificate for server to verify the identity of the client.

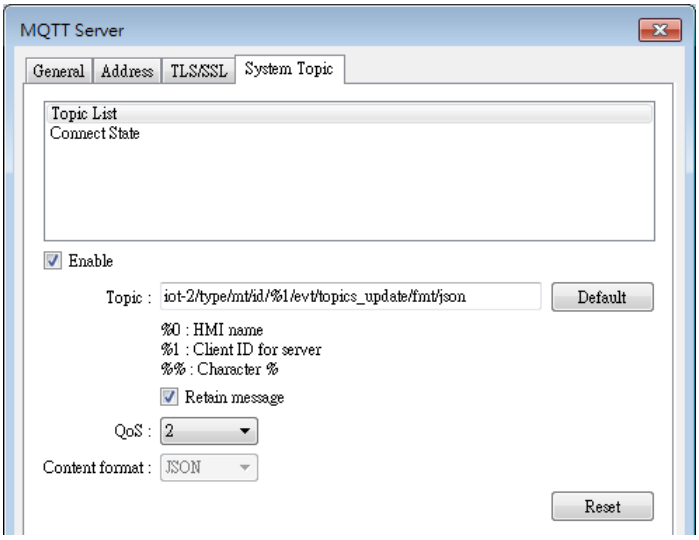


System Topic

When HMI is the publisher and is connected to the MQTT Server for the first time, two system topics at QoS2 will be sent from HMI.

1. iot-2/type/mt/id/<Client ID>/evt/topics_update/fmt/json
   This topic contains the all topics originating from the HMI, and 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 the server.

When using an MQTT Server that has restrictions to the topics (e.g. public cloud service), please do not enable System Topic, in order to avoid connection error.



Topic Publisher Settings

After setting MQTT Server, open MQTT Topic settings. 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.

e.g.: iot-2/type/cMT-SVR/id/%0/evt/topic 1/fmt/json
%0 stands for HMI name. If the HMI name is “Default HMI”, then the topic published will be: iot-2/type/cMT-SVR/id/Default HMI/evt/topic 1/fmt/json

%0: HMI name can be found by opening System Settings on HMI, or use system register LW-10884.
%1: Client ID for Server
%(DYNAMIC): Dynamic String
%%: character %

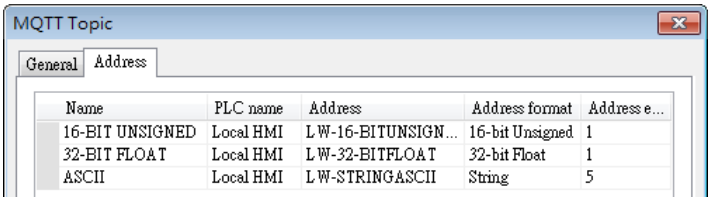
The next step is to select a Sending Mode from: Address (Auto.), Address (Bit trigger) or Event (Alarm) Log.

[Address (Auto.)]:
1. Value-trigger-based: MQTT message is sent when any value in the Topic changes.
2. Time-based: Data is published at a fixed time interval.

[Address (Bit trigger)]: MQTT message is sent when a designated bit address is triggered.

[Event (Alarm) Log]: MQTT message is sent when a selected event occurs. The event should be configured in Event Log first.

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

QoS 0: At most once, messages are not persistent.

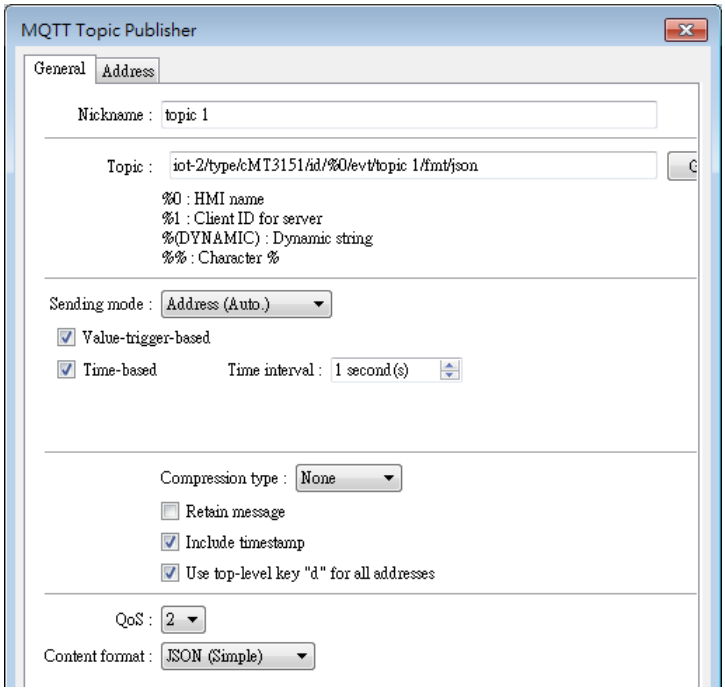
QoS 1: At least once.

QoS 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. Supported compression algorithms are zlib, gzip, and DEFLATE .

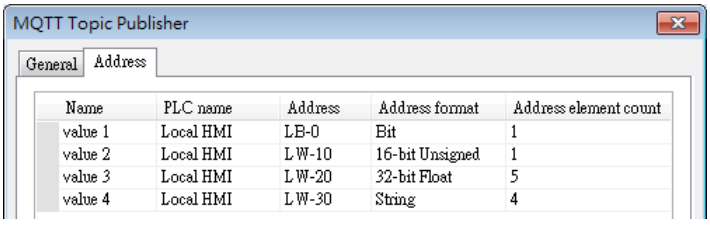
[Content format]: The supported formats include: Raw data, JSON (Simple) and JSON (Advanced).



Content formats

Raw data:

When publishing four values in the following address formats:

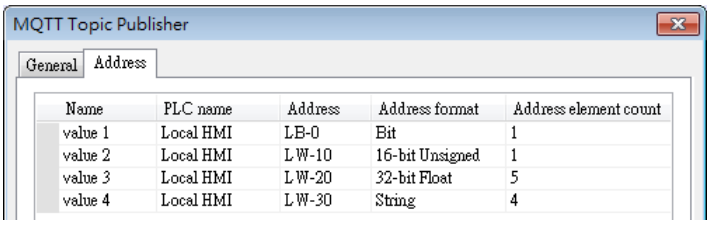


The data published from HMI will be:

0002 009A 9999 3F00 0000 0000 0000 0000

0000 0000 0000 0041 4243 4400 0000 00

JSON (Simple):
When publishing four values in the following address formats:



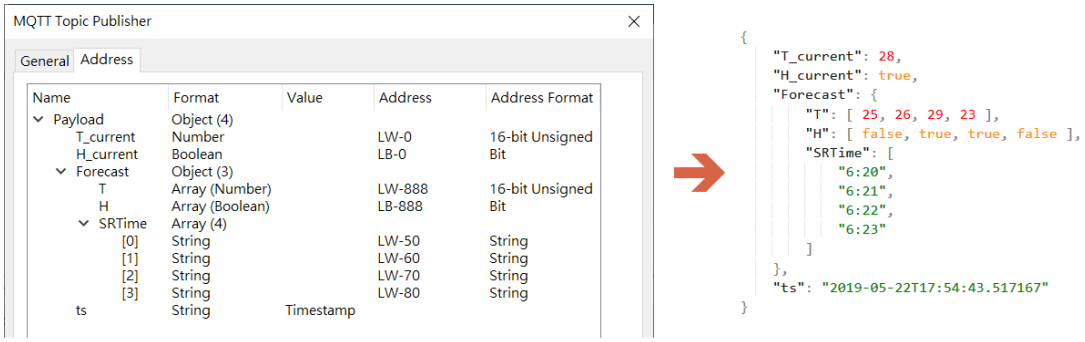
The data published from HMI will be:

```
{
    "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"
}
```

The data published will be in the common key:value format when [Include time stamp] and [Use top-level key “d” for all addresses] options are both disabled in the General tab, and [Remove JSON array bracket “[“ and “]”.] option is selected for each address.
Example data:

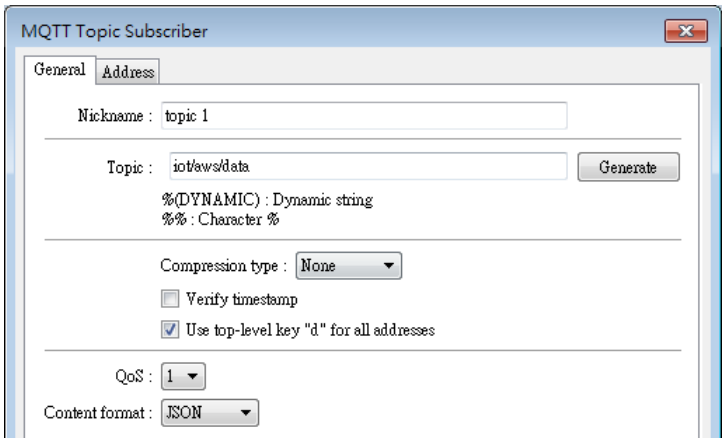
```
{
   "W": 7,
   "B": true
}
```

JSON (Advanced): This mode allows users to configure more complex JSON format, including nested format.
As shown in the following illustration, settings on the left will generate the JSON formatted data on the right.



Topic Subscriber Settings

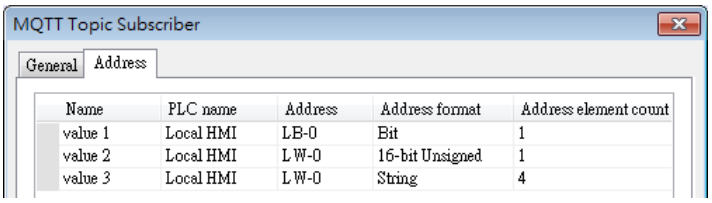
MQTT Topic Subscriber settings are similar to Publisher settings demonstrated in the preceding chapters. The topic and the corresponding addresses must be configured. The subscribed data will be written into the corresponding addresses when the message is successfully received.



[Verify timestamp]: With this option selected, whether the timestamp of each JSON message is increasing will be checked. Only when the time stamp does increase will the data be written into the corresponding addresses after receiving the message.

Content Format can be JSON or Raw Data.
The format subscribed by the HMI must be identical to the format of the message received; otherwise the data will not be written to the corresponding address after receiving the message.

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



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

Subscribing to the messages published by another Weintek HMI can usually be successfully achieved when the settings in Subscriber matches the settings in Publisher. To receive messages from other sources, please check message settings carefully, for example, whether [Include timestamp] or [Use top-level key “d” for all addresses] is enabled, and make sure to configure settings in the subscriber in accordance.

Project and Application

In MQTT settings, a control address and a status address can be designated to control MQTT during HMI run time. The information of the addresses can be found in EasyBuilder Pro settings dialog box.

The following is an example showing how MQTT can be controlled dynamically during HMI run time when LW-102 is designated as the control address, and LW-100 is designated as the status address:
During HMI run time:
Setting LW-102 to 1 connects HMI with MQTT server.
Setting LW-102 to 2 disconnects HMI with MQTT server.
Setting LW-102 to 3 after updating the control parameters (LW-107, LW-108) will connect HMI with MQTT server using the new parameters.
The status address LW-100 can show connection status, and errors are indicated in LW-101.
The MQTT settings can be changed dynamically during HMI run time by using the control addresses mentioned above.

3. Choosing an MQTT Server

Using Built-in MQTT Server on HMI

To use HMI's built-in MQTT server, use IP address: 127.0.0.1. The client program can connect to the MQTT server using the IP address of the HMI.

At the first time using HMI's MQTT server, please downloaded it to HMI using EasyBuilder Pro. Please select [Runtime] when downloading MQTT server.

Using a Public Cloud MQTT Server

To use a public cloud MQTT server, enter the IP address or website address, and the communication port of the cloud service when configuring MQTT server.
e.g. AWS IoT's MQTT server, Alibaba Cloud's MQ server.

Using Self-build MQTT Server

The user can host an MQTT server. The following are some choices:

1. Mosquitto (http://mosquitto.org/download/)
2. EMQ (http://emqtt.io/)

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

4. References

HMI MQTT is integrated into AWS IoT, allowing users to use AWS IoT in the following ways:

1. Use AWS IoT as a standard MQTT server.
2. Use AWS IoT as an entrance for data to be transmitted to other AWS cloud services.
3. Deep integration with AWS IoT Thing Shadow service.

Please see more information in the following documents:
HMI MQTT meets AWS IoT
HMI MQTT meets AWS IoT Walkthrough