DCC-EX MQTT Support MVP 1.0

@grbba, May 2021

This is a short guide on how to use the first MQTT implementation for the DccEX CommandStation.

Philosophy

MQTT as is a n to m communication protocol which is based on « topics » managed by a broker which handles all the traffic and topics between the clients. Any client connected to the broker can subscribe and/or publish to topics. A client will receive all messages from the topics he has subscribed.

The very first goal for the MQTT support was to get a foundation for 'MQTT Throttles' or the DccEX command-line-interface I am currently building. Throttles need by nature 1:1 communication for proper functioning

In order to provide '1:1' communications over MQTT such as you would get using the Ethernet or WiFi interface some additional logic on top has to be provided.

The MVP implementation of DccEX-MQ provides such a mechanism establishing a 'MQTT' tunnel. BUT be aware that today the tunnel is not 'secure' as in that there is no 100% guarantee that the 'tunnel' and associated topics are exclusively used by the client who created the tunnel in the first place. This is especially true for public brokers. Nevertheless, this risk can be minimized by various means which will be discussed later, the purpose of this document being a basic how-to.

What you need:

The following lists the basic requirements to complete the guide:

- **The MQTT branch of CommandStation-EX from GitHub**. This branch contains the MQTT code and is based on CommandStation-EX v3.1. It will NOT work from master…
  Install the branch inside your favorite IDE. I assume here that you know how to do this as well as how to compile the CommandStation-EX code as well as how to upload the compiled sketch. If you have issues leave me a message on discord either on dm or the #mqtt channel.

- Besides the Ethernet Library you need the **PubSubClient Library** which you can obtain from the library manager in the Arduino IDE or PlatformIO.

- **A CommandStation with an EthernetShield** (sorry no ENC based ones as the PubSubLibrary doesn't support those and the alternative for this chipset hasn't been maintained since 2015). No WiFi as the CommandStation-EX WiFi code doesn't yet provide a 'Client' class implementation which is required for the PubSubClient. So a 'classic' setup should work with the Arduino Mega (UNO is too short on resources and although the CommandStation-Ex code will compile the required code will not be added), One of the supported MotorShields and the Arduino Ethernet Shield (which has the W5500 chip-set) but any W5xx WizNet based shield shall do.

- **An MQTT client** to send/receive commands/results. I suggest MQTT Explorer (`http://mqtt-explorer.com/`) which is available for all major platforms. The remainder of this document will use MQTT explorer and show the steps to get up and running. Ideally, I would have an MQTT Throttle but that's not the case and I implement a Dcc-EX command line interface which will serve as reference implementation of the client part of the DccEX-MQ protocol.
- **An active Internet connection** reachable by the CommandStation

Some little point before you ask:

What you don't need is a MQTT broker as we will use for this guide a publicly available broker.

All is installed and ready to go?

Initial setup

Configuration

First thing to do is to create the config.h file.

Copy the config.example.h file to config.h in the same folder and open the file for editing

Make sure that:

- `WIFI_ENABLED` as well as `ETHERNET_ENABLED` are commented out
- `MQTT_ENABLED` is NOT commented.

As already mentioned, WiFi and Ethernet are incompatible with MQTT for now. The DccEX-MQ protocol does take care of the Ethernet connection independently of the CommandStation-EX EthernetInterface.

For the MQTT section your config.h file should look like this:

```
// ENABLE_MQTT: if set to true you have to have an Arduino Ethernet card (wired). This
// is not for Wifi. You will need the Arduino Ethernet library as well as the PubSub
// library from <add link here> or get via the library manager either from the IDE
// or PIO

// The following is only needed if the Broker requires it. cf broker descriptions below
#define MQTT_USER   "your broker user name"
#define MQTT_PWD    "your broker passwd"
#define MQTT_PREFIX "prefix if required by the broker"

// UNCOMMENT THE FOLLOWING LINE TO ENABLE MQTT
#define ENABLE_MQTT true

// Set the used broker to one of the configurations from MQTTBrokers.h where some
// public freely available brokers are configured

// DEFINE THE MQTT BROKER BELOW ACCORDING TO THE FOLLOWING TABLE:
//
//  DCCEX_MQTT_BROKER : DCCEX Team best effort operated MQTT broker;
//  pls apply for user/pwd on discord in the mqtt channel if you want to try it
//  DCCEX_MOSQUITTO   : Mosquitto.org public test broker no user / pwd required so anyone
//  can subscribe/publish to any topic here; good for testing only
//  DCCEX_HIVEMQ      : Provided by HiveMQ; Public no user / pwd required
//   |
//   +------------------------v

#define CSMQTTBROKER DCCEX_MOSQUITTO
```

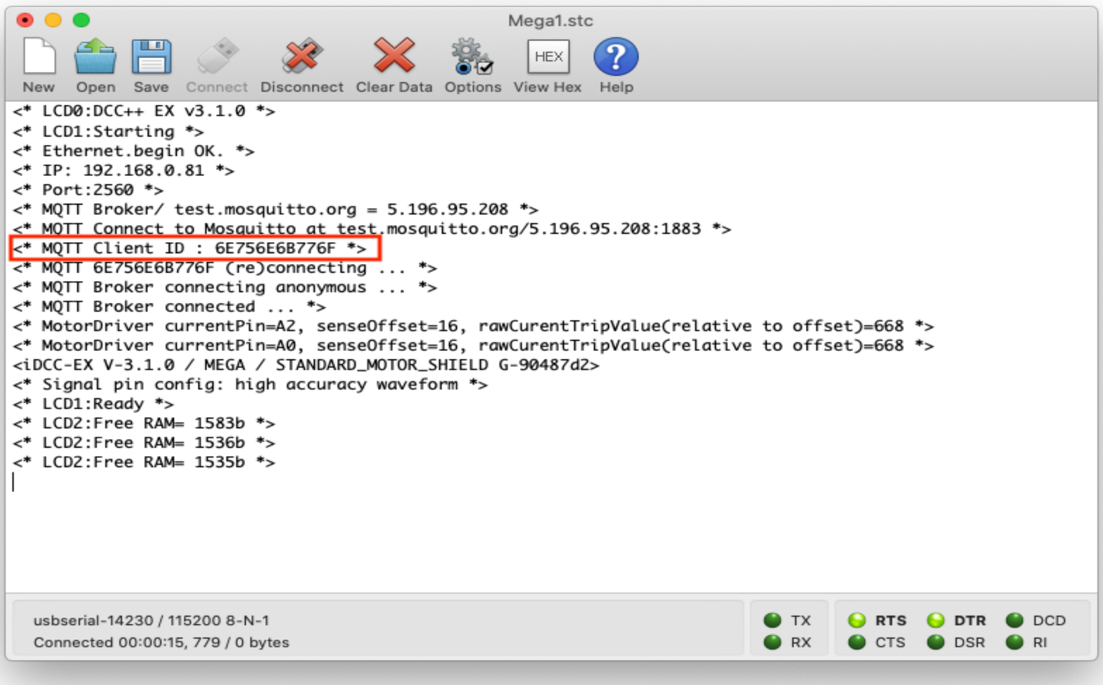
You don't need to touch any of the user / password defines as we use the public broker provided by mosquito.org. Alternatively, you can use `DCCEX_HIVEMQ` which is also preconfigured for the CommandStation.

Save the file and you are good to go for compilation and uploading the sketch! Done?

Run CommandStation-EX

The CommandStation code has been uploaded to your Arduino and the next step is to collect one important piece of information and that is the unique CommandStation Identifier.

Run a serial monitor of your choice e.g., the one integrated in the ArduinoIDE or PlatformIO and connect to the Arduino. I use CoolTerm here as independent Serial monitor (which can be found here: `http://freeware.the-meiers.org`)



```
<* LCD0:DCC++ EX v3.1.0 *>
<* LCD1:Starting *>
<* Ethernet.begin OK. *>
<* IP: 192.168.0.81 *>
<* Port:2560 *>
<* MQTT Broker/ test.mosquitto.org = 5.196.95.208 *>
<* MQTT Connect to Mosquitto at test.mosquitto.org/5.196.95.208:1883 *>
<* MQTT Client ID : 6E756E6B776F *>
<* MQTT 6E756E6B776F (re)connecting ... *>
<* MQTT Broker connecting anonymous ... *>
<* MQTT Broker connected ... *>
<* MotorDriver currentPin=A2, senseOffset=16, rawCurentTripValue(relative to offset)=668 *>
<* MotorDriver currentPin=A0, senseOffset=16, rawCurentTripValue(relative to offset)=668 *>
<iDCC-EX V-3.1.0 / MEGA / STANDARD_MOTOR_SHIELD G-90487d2>
<* Signal pin config: high accuracy waveform *>
<* LCD1:Ready *>
<* LCD2:Free RAM= 1583b *>
<* LCD2:Free RAM= 1536b *>
<* LCD2:Free RAM= 1535b *>
```

```
usbserial-14230 / 115200 8-N-1
Connected 00:00:15, 779 / 0 bytes
```

What we look for is the ClientID as shown in the figure above in the red rectangle. In this case our CommandStation Identifier is 6E756E6B776F.

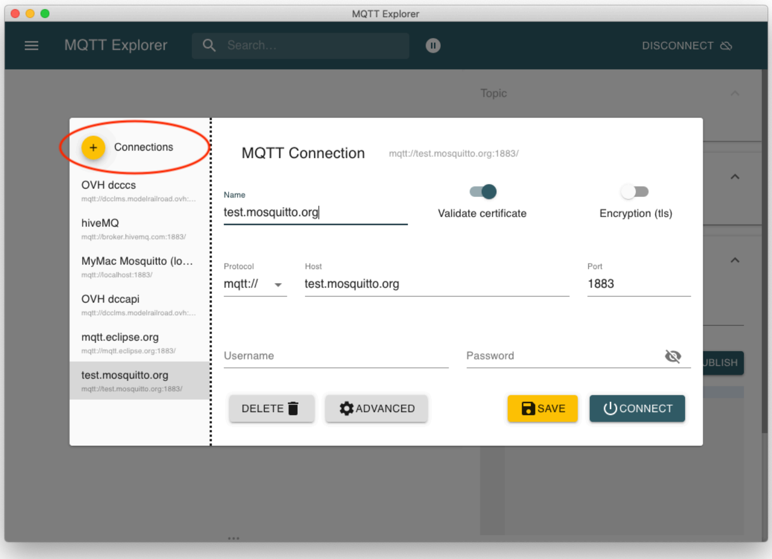
The two lines above the ClientID show to which MQTT Broker the CommandStation has connected to. In this case the Mosqiutto.org public test broker (which is preconfigured within the CommandStation). The clientID is important as this id corresponds to the initial topic the command station has subscribed to. Note the id down as we need it for the next steps.

The command station is now ready to initialize the DccEX-MQ protocol i.e create one or more tunnels (one for each client to connect to the command station) which we will do in the next steps starting with a MQTT client in our case MQTT-Explorer.

Run MQTT-Explorer

Connect to the Broker

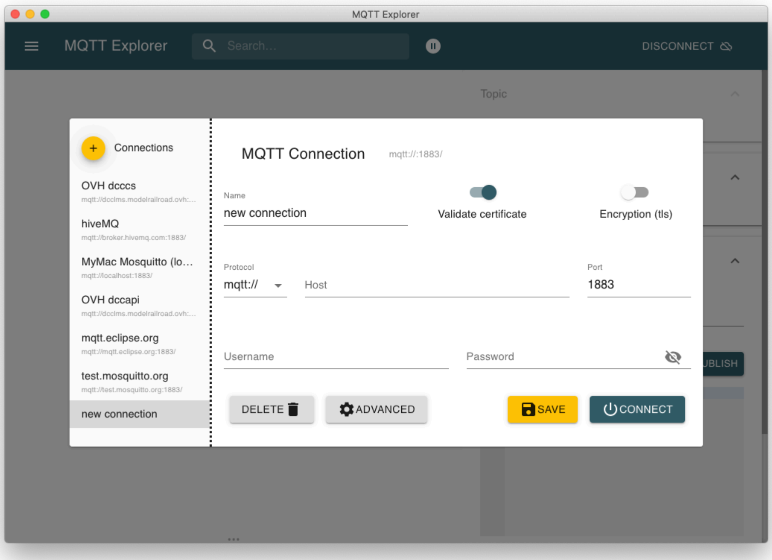
When starting MQTT-Explorer you will be presented with the following screen (if you use it for the first time the column on the left will be empty as well as all the fields for the connection details). Click on the yellow plus to create an empty new connection.



MQTT Connection mqtt://test.mosquitto.org:1883/

Name: test.mosquitto.org
Validate certificate
Encryption (tls)
Protocol: mqtt://
Host: test.mosquitto.org
Port: 1883
Username
Password

As you can see there are already some Brokers configured but for the exercise, we will configure a new connection. Remember we have been connecting the command station to the mosquito org test broker and the necessary configuration information can be found on the serial monitor as well just above the ClientID. Once the yellow plus has been selected you should see:



MQTT Connection mqtt://:1883/

Name: new connection
Validate certificate
Encryption (tls)
Protocol: mqtt://
Host
Port: 1883
Username
Password

1) Fill in the required fields:

- Name: CommandStationEX
- Protocol: Leave as is
- Host: test.mosquitto.org (as seen on the serial monitor)
- Port: 1883 (as seen on the serial monitor)

As this is a free public broker for testing purposes there is no Username / Password required to use the broker. All brokers support also a secure TLS encrypted transport using port 8883 (But that depends on the broker implementation and may change 1883 and 8883 are the default ports) but the Arduino and the EthernetShield aren't capable of handling the encryption necessary for this.)

2) Click on 'Save'

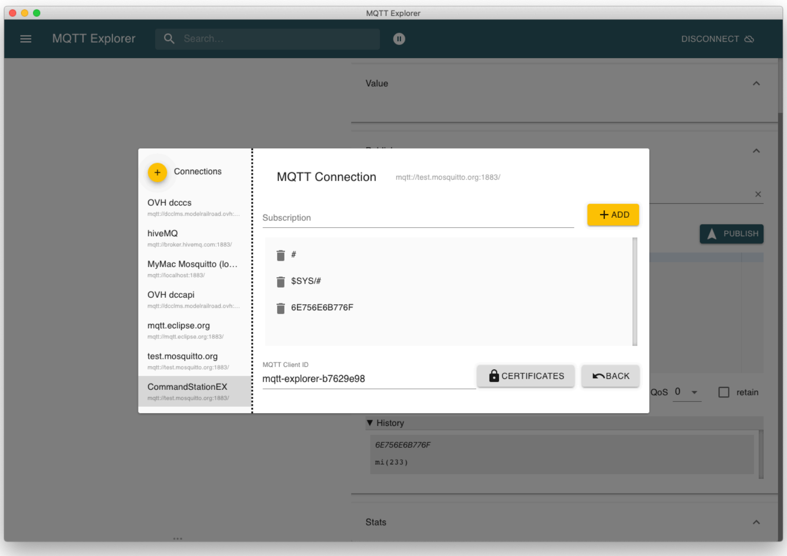
This registers your connection for future reuse in the left-hand panel.

At this point you are ready to connect MQTT-explorer to the broker but not yet capable to talk to the command station. One more step is required.

3) Click on 'Advanced'

You will see:

4) Fill in 'Subscription' the ClientID from your serial monitor (not the one I use here - your command station will not have the same ClientID) and press the yellow Add button and the window shall look like this:



MQTT Connection mqtt://test.mosquitto.org:1883/

Subscription

- #
- $SYS/#
- 6E756E6B776F

MQTT Client ID: mqtt-explorer-b7629e98

QoS 0

```
6E756E6B776F
mi(233)
```

The MQTT client is now subscribed to the same channel as the command station and can send each other messages over this subscription.

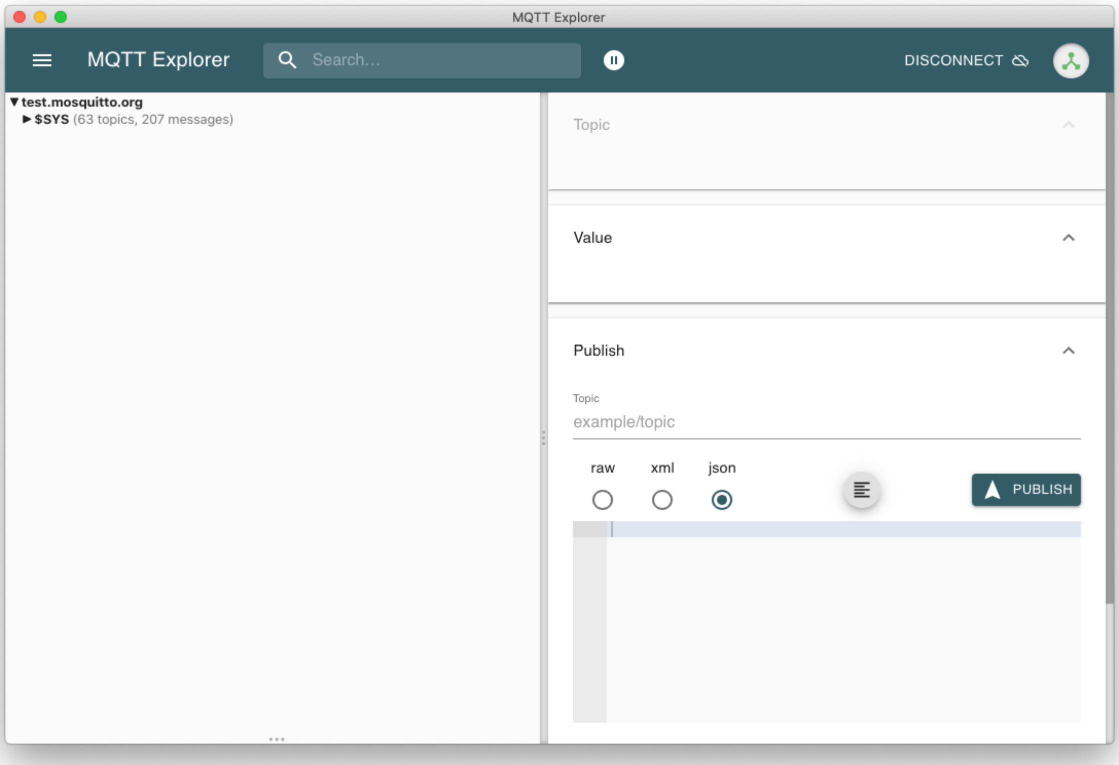
5) click 'Back' and then 'Save' to make sure the next time you connect to get the subscription again.

Note: The test broker we use here is public and needs no username/password i.e., everyone with the knowledge of the CommandstationID can subscribe to the topic and send/receive messages to/from the command station.

We have set up the basic communication channel for now. The next step is getting a sort of « 1:1 » MQTT tunnel between the client and the command-station.

Initialize MQTT-Tunnel with the CommandStation

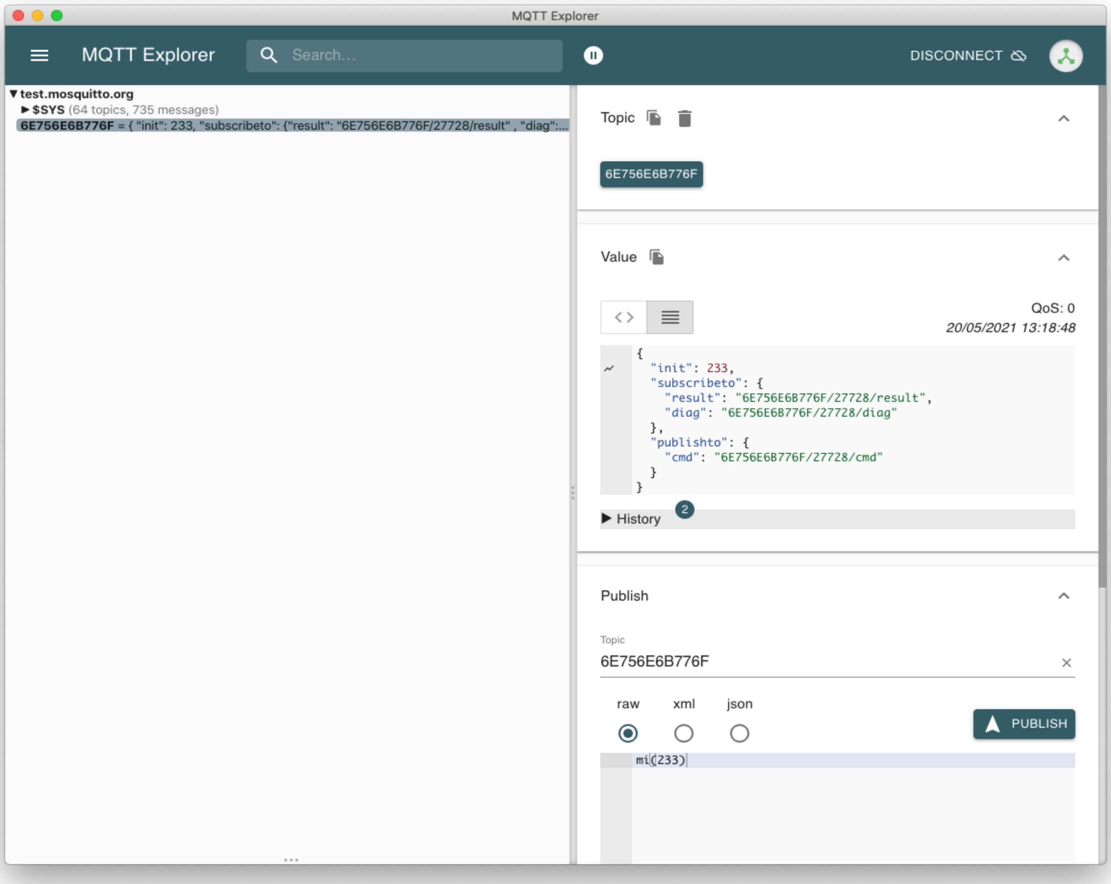
Now that we have the basic communication going it's time to send the first message to the command station in order to obtain a private tunnel in order to do that. Once connected to the broker from the MQTT-Explorer main screen you shall see this window:



In the publish section where it says 'Topic' enter your CommandstationID. Then select 'raw' as type of message and in the text entry field just below type the following: `mi(233)` `mi(number)` is a command the command station understands for a MQTT client wanting to initialize a channel. Number is a random number between 1 and 254 chosen by the client and used by the client to look for a response from the command station.

Once entered press 'Publish' (be sure that the command station is actually running and connected t the broker by checking the serial monitor) cf. image below. That will send the command to the broker which distributes the message to the command station (remember the command station is subscribed to the topic named after is CommandstationID). In response the command station publishes the following JSON message (cf. image below):

```
`{
    "init": 233,
    "subscribeto":{"result":"6E756E6B776F/27728/result","diag":"6E756E6B776F/27728/diag"},
    "publishto": {"cmd": "6E756E6B776F/27728/cmd"}
}`
```



```
▼ test.mosquitto.org
  ► $SYS (64 topics, 735 messages)
  6E756E6B776F = { "init": 233, "subscribeto": {"result": "6E756E6B776F/27728/result" , "diag":...
```

Topic: 6E756E6B776F

QoS: 0
20/05/2021 13:18:48

```
{
  "init": 233,
  "subscribeto": {
    "result": "6E756E6B776F/27728/result",
    "diag": "6E756E6B776F/27728/diag"
  },
  "publishto": {
    "cmd": "6E756E6B776F/27728/cmd"
  }
}
```

Publish Topic: 6E756E6B776F

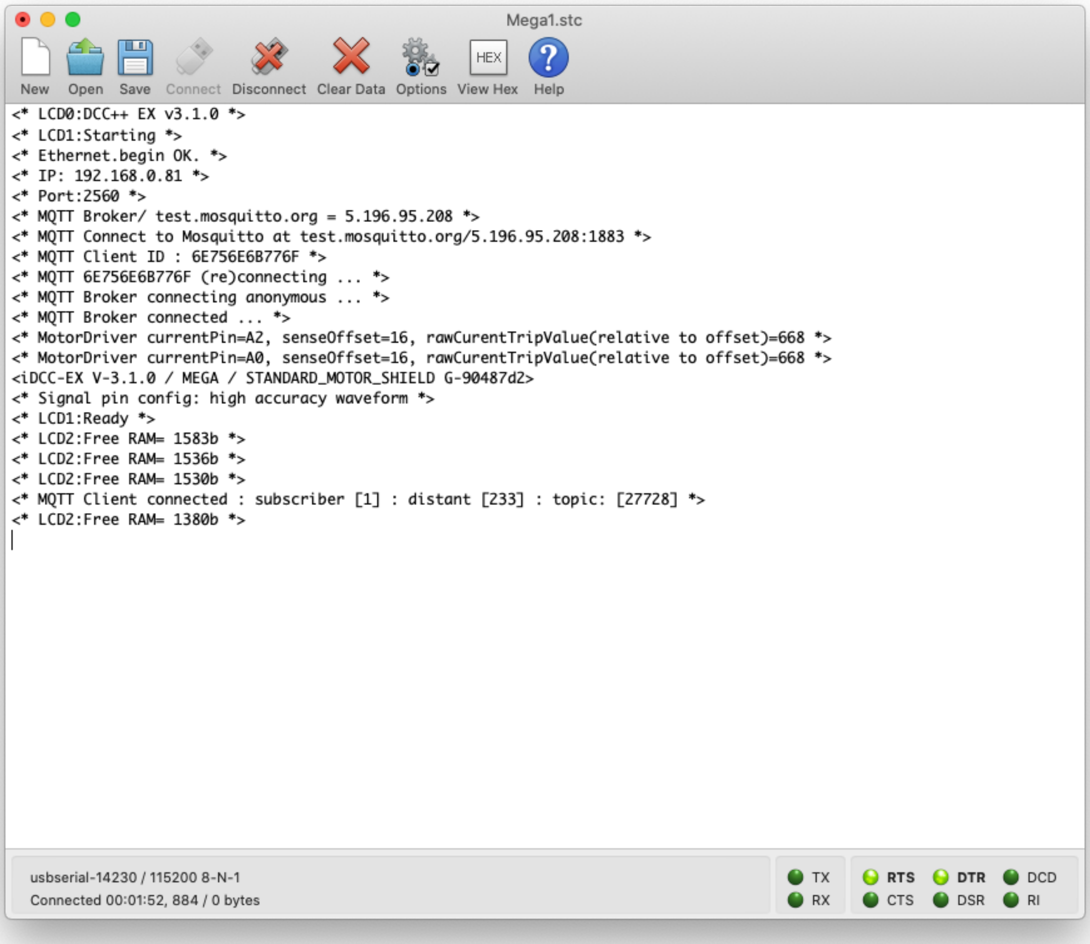
Format: [✓ raw] xml json

```
mi(233)
```

This return message signifies that the command station asks the client who send the `mi(233)` message to subscribe to two topics : `6E756E6B776F/27728/result` and `6E756E6B776F/27728/diag` as the command station will publish on those topics any result from a received command published by the client onto the topic mentioned in the « publishto » key in this case `6E756E6B776F/27728/cmd`

The Serial monitor should show the following and in particular the line:

```
<* MQTT Client connected: subscriber [1]: distant [233]: topic: [27728] *>
```



```
<* LCD0:DCC++ EX v3.1.0 *>
<* LCD1:Starting *>
<* Ethernet.begin OK. *>
<* IP: 192.168.0.81 *>
<* Port:2560 *>
<* MQTT Broker/ test.mosquitto.org = 5.196.95.208 *>
<* MQTT Connect to Mosquitto at test.mosquitto.org/5.196.95.208:1883 *>
<* MQTT Client ID : 6E756E6B776F *>
<* MQTT 6E756E6B776F (re)connecting ... *>
<* MQTT Broker connecting anonymous ... *>
<* MQTT Broker connected ... *>
<* MotorDriver currentPin=A2, senseOffset=16, rawCurentTripValue(relative to offset)=668 *>
<* MotorDriver currentPin=A0, senseOffset=16, rawCurentTripValue(relative to offset)=668 *>
<iDCC-EX V-3.1.0 / MEGA / STANDARD_MOTOR_SHIELD G-90487d2>
<* Signal pin config: high accuracy waveform *>
<* LCD1:Ready *>
<* LCD2:Free RAM= 1583b *>
<* LCD2:Free RAM= 1536b *>
<* LCD2:Free RAM= 1530b *>
<* MQTT Client connected : subscriber [1] : distant [233] : topic: [27728] *>
<* LCD2:Free RAM= 1380b *>
```

```
usbserial-14230 / 115200 8-N-1
Connected 00:01:52, 884 / 0 bytes
```

The message in the serial monitor tells us that there is 1 subscriber now which has send 233 as initializer and been assigned topic identifier 27728.

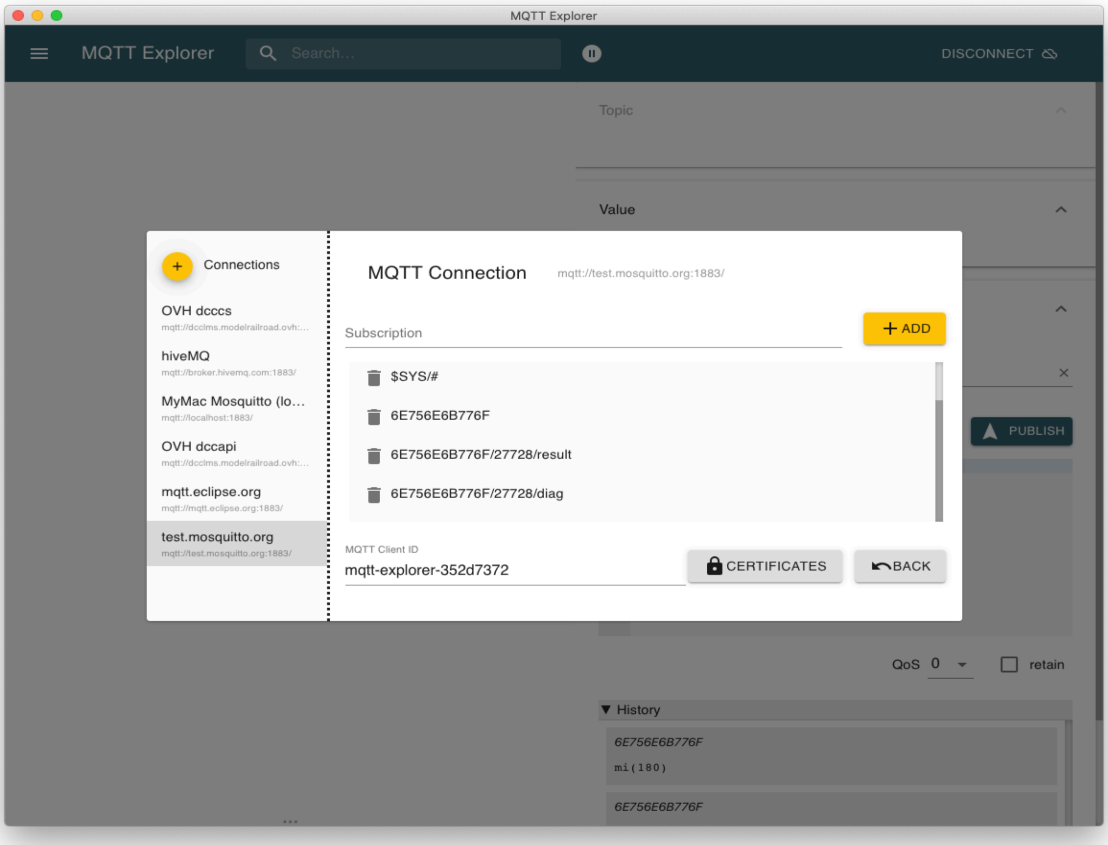
Note: Obviously the random number 233 and any random number used, can be reused any time by any client to connect and obtain its topics It is therefore important if you implement an MQTT client to ignore all « init » responses in order to avoid confusion as they shouldn't belong to you anymore but another client. There is a slight chance of collision for the initialization process when actually two clients send the same random number in a « mi » at more or less the same time and the replies arrive in the same time window it needs for the command station to process them. [1]

In an MQTT command station client the subscription changes would be handled automatically upon reception of the JSON message and having determined that the init

[1] The command station does not yet issue a warning in this case. This is on the roadmap

value corresponds to the one high has been send but in the case of MQTT Explorer we have to do this by hand editing the subscription page in the connection setup.

So, disconnect by clicking 'Disconnect' at the top right-hand corner which brings you back to the connection screen. Again, press here 'Advanced' and add the topics received in the « subscribeto » key. The subscribe screen should look like this now. Again click 'Back' and then 'Save' and reconnect.



OVH dcccs
mqtt://dcclms.modelrailroad.ovh:...
hiveMQ
mqtt://broker.hivemq.com:1883/
MyMac Mosquitto (lo...
mqtt://localhost:1883/
OVH dccapi
mqtt://dcclms.modelrailroad.ovh:...
mqtt.eclipse.org
mqtt://mqtt.eclipse.org:1883/
test.mosquitto.org
mqtt://test.mosquitto.org:1883/

MQTT Connection mqtt://test.mosquitto.org:1883/

Subscription
$SYS/#
6E756E6B776F
6E756E6B776F/27728/result
6E756E6B776F/27728/diag

MQTT Client ID
mqtt-explorer-352d7372

6E756E6B776F
mi(180)
6E756E6B776F

Note: For testing purposes using a client like MQTT-Explorer you can use always the same 'random' number for connection in order to avoid having to change the subscriptions accordingly every time. In a normal MQTT command station client this should be truly randomized as the topic setup should also be done automatically

Note: You can try to send the same mi message a second time as you will see in the serial monitor the command station will consider this as a new client connecting to it and create the topic for this one as well.

Note: The command station accepts up to 20 connections for now where in theory 255 are possible. This will nevertheless not work due to resource limitations on the underlying Arduino hardware

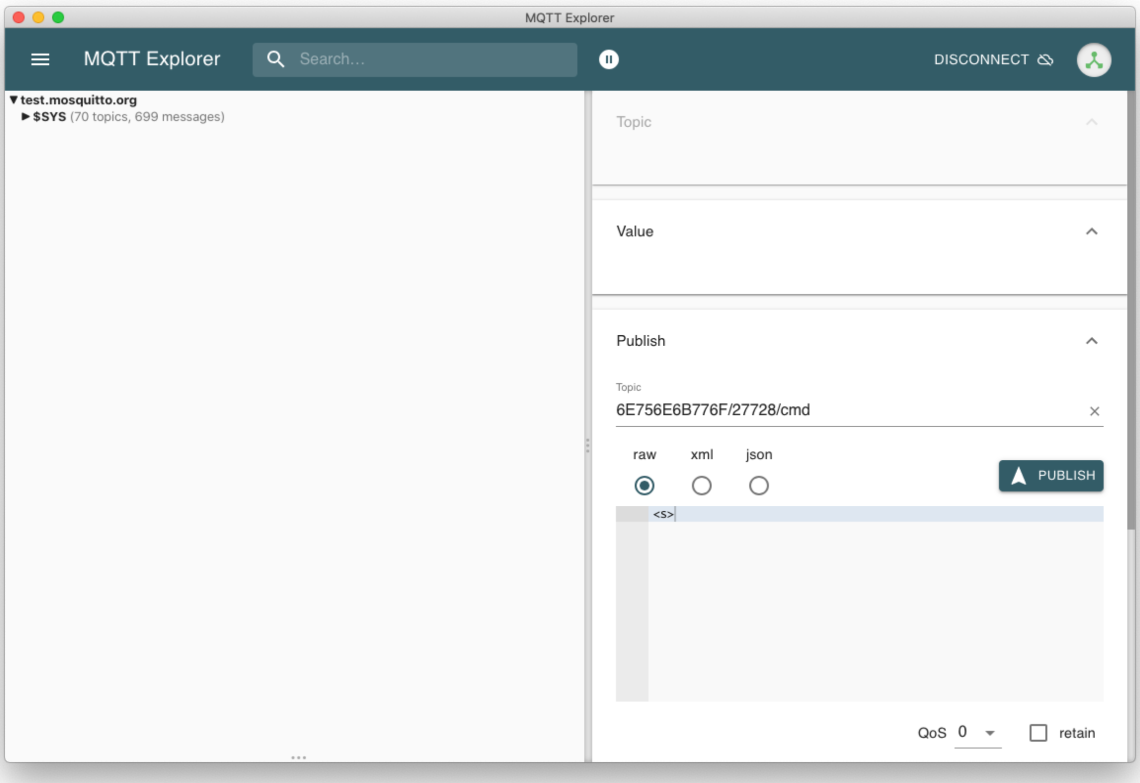
Note: The handshake process shown manually here using MQTT explorer has to be implemented by any MQTT enabled client in order to communicate with the command station. In case of the reference implementation in the ongoing development of the DccEX

Command line interface written in C++ we use the Eclipse Paho C++ MQTT library. Multiple language bindings exist though for the Paho MQTT library should you which to implement a client yourself or integrate MQTT into an existing client for CommandStation Ex whatever the technology is you chose.

Now we have the link between the client and the command station successfully created let's send some commands

Send / Receive commands / results

In the same pane as before when issuing the 'mi' command on the CommandstationID topic now enter the topic you received in the « publishto »key e.g., like this:



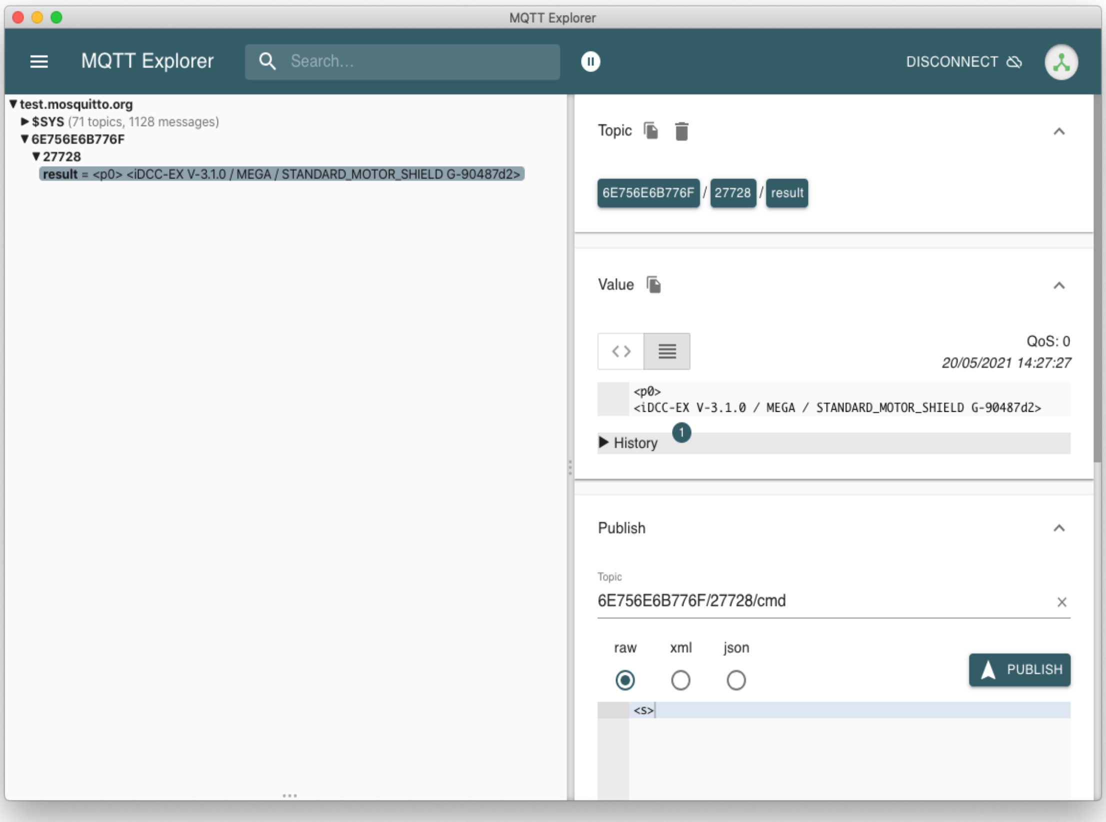
Topic: 6E756E6B776F/27728/cmd

[✓ raw] xml json

```
<s>
```

QoS 0

The <s> command has been edited in the text pane as raw message. Once you have entered all this press 'Publish' and you should see the following result:



```
▼test.mosquitto.org
  ▶$SYS (71 topics, 1128 messages)
  ▼6E756E6B776F
    ▼27728
      result = <p0> <iDCC-EX V-3.1.0 / MEGA / STANDARD_MOTOR_SHIELD G-90487d2>
```

Topic: 6E756E6B776F / 27728 / result

QoS: 0
20/05/2021 14:27:27

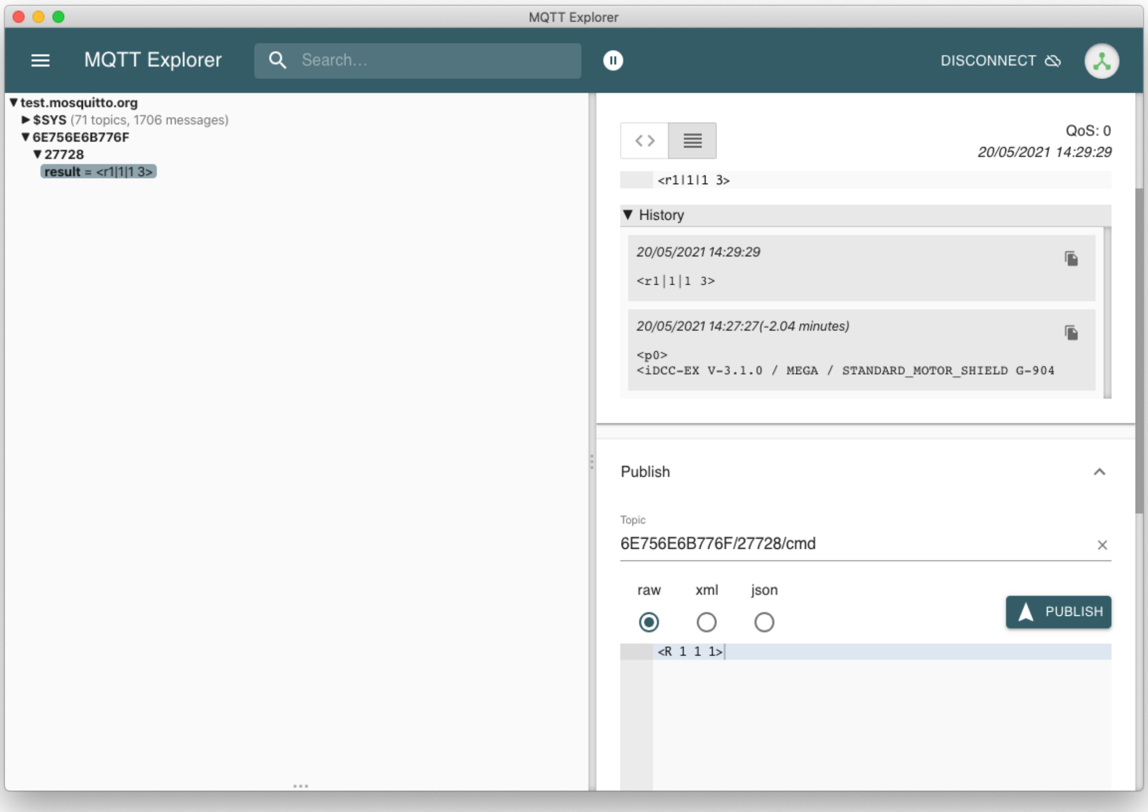
```
<p0>
<iDCC-EX V-3.1.0 / MEGA / STANDARD_MOTOR_SHIELD G-90487d2>
```

Topic: 6E756E6B776F/27728/cmd

[✓ raw] xml json

```
<s>
```

You may not see the full tree until the result immediately on the left-hand pane just click on the triangles right next to the topics to open the tree. You can also see the result in the history on the right-hand side. Another example is given below sending a read command to the command station together with the result shown.



```
▼test.mosquitto.org
  ▶$SYS (71 topics, 1706 messages)
  ▼6E756E6B776F
    ▼27728
      result = <r1|1|1 3>
```

QoS: 0
20/05/2021 14:29:29

```
<r1|1|1 3>
```

20/05/2021 14:29:29

```
<r1|1|1 3>
```

20/05/2021 14:27:27(-2.04 minutes)

```
<p0>
<iDCC-EX V-3.1.0 / MEGA / STANDARD_MOTOR_SHIELD G-904
```

Topic
6E756E6B776F/27728/cmd

[✓ raw] xml json

```
<R 1 1 1>
```

This concludes the guide for initial MQTT setup allowing to communicate the DccEX protocol, i.e., < > commands, using multiple clients with the command station over MQTT. I hope you found this helpful, and any comment is welcome. You can contact me on Discord @grbba via DM or the #mqtt channel.

- grbba

Things to do / RoadMap / Brainstorm for additional features:

- Tee diagnostic output to the diagnostic channel as well
  - Rename ClientID to CommandstationID in the serial output for more clarity
  - Provide a client class for the WifiInterface of CommandStation-EX
  - Enable concurrent use of MQTT / Ethernet / WiFi which may be help full sometimes but will eat into the available memory
  - Telemetry of sensors/turnouts/accessories etc. (commands for those are already handled but feedback / status checks have yet to come)
  - Broadcasting to all connected clients (e.g. acquisition/release of a loco, etc …)
  - WiFi over ESP enabling SSL traffic with the broker

- Issue a warning/error message in case of an init collision
- Last Will Messages for graceful exit/disconnect (from the client side - e.g. what to do if the connection is lost like issue an emergency stop - or from the command station side for the client to handle)