DCC-EX MQTT Support MVP 1.0

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?

Inital 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 libray 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 avaiable 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

We need to collect one important piece of information and that is the unique CommandStation Identifier :

RoadMap / Brainstorm for additional features:

- 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 …)