

35 MQTT

Introduction

MQTT (short for MQ Telemetry Transport) is an open OASIS and ISO standard (ISO/IEC PRF 20922) lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP, although its variant, MQTT-SN, is used over other transports such as UDP or Bluetooth. It is designed for connections with remote locations where a small code footprint is required or the network bandwidth is limited.

The broker acts as a post office, MQTT doesn't use the address of the intended recipient but uses the subject line called "Topic", and anyone who wants a copy of that message will subscribe to that topic. Multiple clients can receive the message from a single broker (one to many capability). Similarly, multiple publishers can publish topics to a single subscriber (many to one).

Each client can both produce and receive data by both publishing and subscribing, i.e. the devices can publish sensor data and still be able to receive the configuration information or control commands. This helps in both sharing data, managing and controlling devices.

With MQTT broker architecture the devices and application becomes decoupled and more secure. MQTT might use Transport Layer Security (TLS) encryption with user name, password protected connections, and optional certifications that requires clients to provide a certificate file that matches with the server's. The clients are unaware of each others IP address.

The broker can store the data in the form of retained messages so that new subscribers to the topic can get the last value straight away.

The main advantages of MQTT broker are:

- Eliminates vulnerable and insecure client connections
- Can easily scale from a single device to thousands
- Manages and tracks all client connection states, including security credentials and certificates
- Reduced network strain without compromising the security (cellular or satellite network)

Each connection to the broker can specify a quality of service measure. These are classified in increasing order of overhead:

- At most once - the message is sent only once and the client and broker take no additional steps to acknowledge delivery (fire and forget).
- At least once - the message is re-tried by the sender multiple times until acknowledgement is received (acknowledged delivery).
- Exactly once - the sender and receiver engage in a two-level handshake to ensure only one copy of the message is received (assured delivery).

Using WCC Lite as MQTT Client

MQTT serves as an alternative for protocols conforming to IEC standards, for example, to send data to a cloud infrastructure that supports MQTT instead of IEC-60870-5-104.

✔ WCC Lite supports MQTT messaging compatible with MQTT v3.1 standard (starting from version **v1.4.0**). Such messaging is possible via mapping of Redis and MQTT data therefore data can be transmitted from any protocol that is supported by WCC Lite.

All standard functions, except for data encryption, are supported. Encrypted messages are not supported yet, therefore to ensure security a user would have to use a VPN service. A user can choose from three different Quality of Service levels, select if messages are to be retained, authenticate users and optionally send Last Will messages.

To configure WCC Lite a user can fill in the needed parameters in Excel configuration. These parameters are shown in two tables below.

Table. MQTT parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|-----------|--------|---------------------------|----------|---------------------------------------|-------|-----|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |

| | | | | | | |
|-------------------|---------|--|--------------------------------|------|------|---|
| device_alias | string | Device alias to used in configuration | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 0 | 0 | 1 |
| protocol | string | Selection of protocol | Yes | | MQTT | |
| host | string | MQTT broker IP address selection | Yes | | | |
| port | integer | MQTT broker port selection | No | 1883 | | |
| enable_threshold | boolean | A parameter to determine if identical values should not be sent multiple times in a row. | No | 1 | 0 | 1 |
| gi_interval_sec | integer | Parameter to determine how frequently should all values be sent at once. Disabled if equal to 0. | No | 0 | | |
| mqtt_qos | integer | MQTT Quality of Service for message as in standard | No | 0 | 0 | 2 |
| mqtt_retain | boolean | Selecting if MQTT broker should retain last received messages | No | 0 | 0 | 1 |
| user | string | MQTT user name | Yes | | | |
| password | string | MQTT user password | Yes | | | |
| use_last_will | boolean | Selecting if MQTT should use Last Will and Testament functionality (Default: False) | No | 0 | 0 | 1 |
| last_will_topic | string | Topic to which an MQTT message would be sent if the device abruptly disconnected message broker | Yes (If use_last_will=True) | | | |
| last_will_message | string | Message to be sent over MQTT if the device abruptly disconnected message broker | No | | | |
| last_will_qos | integer | MQTT Quality of Service selection as in standard | No | 0 | | |
| last_will_retain | boolean | Selecting if MQTT broker should retain last will message | No | 0 | 0 | 1 |

To map the signal to send through MQTT client, it should have its device_alias and signal_alias mapped to source_device_alias and source_signal_alias respectively.

If MQTT is configured but does not send data, a user can use command line interface to debug transmission. All options for MQTT process which transmits data over MQTT (called mqtt-client as it

Table. MQTT parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|-----------|------|-------------|----------|---------------------------------------|-------|-----|
| | | | | | Min | Max |

| | | | | | | |
|-----------------------|---------|---|-----|---|---|---|
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Device alias from a Devices tab | Yes | | | |
| signal_alias | string | Unique signal name to be used | Yes | | | |
| source_device_alias | string | device_alias of a source device | No | | | |
| source_signal_aliases | string | signal_alias of a source signal | No | | | |
| enable | boolean | Enabling/disabling of an individual signal | No | 1 | 0 | 1 |
| log | integer | Allow signal to be logged. Log signal with 1 and no logging with 0. | No | 0 | | |
| topic | string | Topic name to override the value built by default | No | | | |

MQTT data format

The format of a MQTT message is a bit different than Redis messages. Redis messages are supported as CSV strings: value,timeStamp,flags (where value can be float, integer or nan; timeStamp - Unix timestamp in milliseconds; flags contain additional information about a measurement). MQTT messages are supported as value,timeStamp,quality (where value can be float, integer or nan; timeStamp - Unix timestamp in milliseconds; quality shows if a value is to be considered as valid). Quality parts of a string is always equal to 1 except for Redis messages containing invalid (IV), non-topic (NT) and/or overflow (OV) flags.

As mentioned, MQTT client acts as an adapter between Redis and MQTT, therefore data from topic in Redis is written to a topic in MQTT. Therefore mqtt-client has to know the mapping table before starting. This table is saved at /etc/elseta-mqtt.json. Every Redis topic name is constructed as tag/[device_alias]/[signal_alias]/[direction]. Prefix tag/ is always used before the rest of argument. device_alias and signal_alias represent columns in Excel configuration. Direction can have one of four possible values - rout, out, in, rin; all of which depend on the direction data is sent or acquired protocol-wise. The same Redis topic structure is preserved in MQTT by default making it easier to find matching signals, however, as no recalculation is done by MQTT and only PUBLISH messages are now supported, only Redis signals with in direction have their MQTT mappings.

A user can create and select his own topic name in Excel configuration, in topic column. As no recalculation is done by MQTT and only PUBLISH messages are now supported, only Redis signals with in direction have their MQTT mappings.

Debugging a MQTT protocol

If configuration for MQTT is set up, handler for protocol will start automatically. If configuration is missing or contains errors, protocol will not start. It is done intentionally to decrease unnecessary memory usage.

MQTT Client command line debugging options

mqtt-client

```
-h [ -help ] Display help information
-c [ -config ] Configuration file location (default - /etc/elseta-mqtt.conf)
-V [ -version ] Show version
-d<debug level> [ -debug ] Set debugging level
-r [ -redis ] Show REDIS output
-m [ -mqtt ] Show MQTT output
```



If MQTT Client does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from command line interface and find out why link is not functioning properly.



To launch a debugging session, a user should stop mqtt-client process and run mqtt-client command with respective flags as was shown above.

