MQTT Client Protocol (Message Queue Telemetry Transport)

Supported device types and versions
Communication line configuration
Communication line parameters
Communication station configuration
I/O tag configuration
Literature
Document revisions

Supported device types and versions

The protocol is an implementation of the MQTT 3.1.1 standard (October 2014). MQTT protocol is a client/server protocol of a subscribe/publish type. It is simple, has little overhead, and is easy to implement. It is used for M2M communication (Machine to Machine) and in the IoT context (Internet of Things). D2000 KOM implements the client part of the protocol. The protocol is implemented on a TCP/IP line.
For transfer of LoRaWAN data encapsulated within the MQTT protocol, see LoRaWan protocol description.

The communication was tested/deployed against:

- TheThings.Network cloud
- Loriot.io cloud
- Slovanet cloud (loralink.slovanet.sk)
- Pixii PowerShaper (energy storage solution)
- liveobjects.orange-business.com cloud

Note: communication with the cloud liveobjects.orange-business.com via websockets (wss://liveobjects.orange-business.com:443/mqtt) was also tested. The program https://github.com/jimparis/unwebsockify.git was used as a WSS wrapper. This program started with the parameters:
./unwebsockify.py --port 1883 --listen 172.16.0.1 wss://liveobjects.orange-business.com:443/mqtt
The D2000 KOM process connected to address 172.16.0.1 on port 1883. The WSS wrapper connected to the defined URL and wrapped the MQTT communication data in a websocket envelope.

Each PUBLISH message contains a topic (Topic), data (Payload), and level of confirmation (QoS). PUBLISH messages can be sent both by the client and the server. The clients at the beginning of the communication will use the SUBSCRIBE message to indicate what topics (parameter of Topic Filter protocol) they are interested in.
The protocol defines the following levels of confirmation of PUBLISH messages - QoS (Quality of Service):

- **QoS_0** - PUBLISH message is not confirmed, it may be lost
- **QoS_1** - PUBLISH message is confirmed by the other side's PUBACK, it may be duplicated
- **QoS_2** - PUBLISH message is confirmed by the other side's PUBREC which is then confirmed back by the PUBREL message and that one by a final PUBCOMP message.

The level of confirmation of the messages sent by the D2000 KOM process is defined by the protocol parameter Publish QoS. The D2000 KOM process considers the writing of the output tag to be successfully finished depending on the QoS:

- **QoS_0** - after the data is successfully sent via the TCP connection
- **QoS_1** - after receiving PUBACK
- **QoS_2** - after receiving PUBCOMP

The MQTT communication starts with the CONNECT message sent by the client (D2000 KOM). The message contains User Name, Password, and other parameters, from which only Clean Session Flag and Client ID can be modified (parameter *Will Flag* is not used, as well as *Will QoS* and *Will Retain*, parameter *Keep Alive* is set to 0). The server replies with a CONNACK message with a return code that contains information about the success of the connect operation.

Then the client sends a SUBSCRIBE message with a filter of topics (Topic Filter parameter), specifying which topics it is interested in, and with the required maximum level of confirmation (parameter Subscribe QoS).

The server responds with a return code that contains information about the success and maximum QoS that was assigned to the requested topics.

Then follows a phase of communication, during which both the client and the server send PUBLISH messages (the client with any topic, the server with topics relating to the filter of topics of the received SUBSCRIBE message) and confirm them according to the value of the QoS parameter of the received PUBLISH messages.

If the server does not send a message for longer than Ping Interval seconds, the client sends the PING request message, to which the server must respond with the PING response message (within the time specified by the Reply Timeout parameter).
If parameters change on the line, the connection is closed and re-created.

The communication has been tested with the MQTT server www.TheThings.network.

Communication line configuration

- Communication line category: TCP/IP-TCP.
- Host: IP address of MQTT server (or redundant addresses separated by a comma or semicolon).
- Port: the default port number is 1883 or 8883 for the encrypted SSL/TLS variant.
- Line number: unused, set the value to 0.

Note: The default port for the MQTT protocol is 1883 or 8883 for the SSL/TLS version. D2000 KOM does not contain an implementation of the SSL/TLS protocol variant, but it is possible to configure it by using the stunnel utility http://www.stunnel.org working in a client mode (client = yes). Stunnel running on the same computer as the D2000 KOM should listen to the 1883 local port and after connecting of D2000 KOM process to the port should encrypt the communication using SLL/TLS and send to the target MQTT server (typically on port 8883).

Forced disconnection: If all stations on the line are in the simulation mode or the communication is stopped for them, the line will be disconnected (the communication socket will be closed). If the simulation is disabled for at least one station and the communication is not stopped for it (the Parameters tab of the *Station* type object), the line will be connected again.

Communication line parameters

Dialog link configuration - **Protocol parameters** tab.
They affect some optional protocol parameters. The following protocol line parameters can be entered:

Table 1

Parameter	Description	Unit / size	Default value
Full Debug	Activates detailed debug information about sending and receiving values.	YES /NO	NO
User Name	User name used in a CONNECT message to connect to the MQTT server.	-	
Password	Password used in a CONNECT message to connect to the MQTT server.	-	
Topic Filter	The name of one topic or a multiple-topic filter sent within the SUBSCRIBE message. Using the filter the MQTT client specifies topics, within which it wants to receive messages. **Note:** topics are hierarchically sorted, a slash (/) is used as the separator, a plus (+) is used as a one-level mask, a hash (#) character is used as a mask for multiple levels. Examples of filter: a/b , level1/+ , # , +/+/+/up **Note:** the change of the *Topic Filter* parameter will be reflected after restarting the communication - e.g. due to the breakdown of the TCP connection, as long as all stations on the line are switched off (StOff) and switched on again, or after a restart of the KOM process. In the first two cases, the message UNSUBSCRIBE is sent to the original *Topic Filter* and then SUBSCRIBE to the new *Topic Filter* (this can be important in so-called persistent sessions, when the Client ID parameter is specified and the MQTT server remembers the state of the client even after the TCP connection is broken).	-	#
Subscribe QoS	The desired maximum level of validation (QoS) sent within the SUBSCRIBE message. The MQTT server can then send PUBLISH messages with such or lower levels of confirmation (but not higher). PUBLISH messages sent by the MQTT server will be confirmed by the D2000 KOM process according to the level of confirmation specified in them. The higher the level of confirmation, the more messages between the client and the server are exchanged (1 at QoS_0, 2 at QoS_1, and 4 at QoS_2).	QoS_0 QoS_1 QoS_2	QoS_1
Client ID	Unique client identifier (Client Identifier) sent within the CONNECT message. **Note:** it is possible to enter a blank string - in which case the server can assign a unique name to the client (if it supports such functionality) or return an error. However, if the Client ID is not specified, the Clean Session Flag parameter settings will be ignored (as the server will assign a unique name each time). The tested MQTT server (thethings.network) returned an error if the Client ID was blank and Clean Session Flag=NO. **Note**: a specific MQTT broker (PIXII.COM) identified clients only by *Client ID*. In practice, this meant that two different D2000 systems that connected to the same broker were considered as one client, and the broker closed an existing connection that it considered old when a new connection was established. After setting the *Client ID* to a unique value, the communications started to work without connection breakdowns.	-	
Clean Session Flag	Parameter Clean Session Flag of the CONNECT message. The *No* value means that the server uses the current session state (connection) - e. g. after collapse and recovery of the TCP connection. This means that all unconfirmed PUBLISH messages with QoS_1 and QoS_2 are resent (optionally also QoS_0, depending on the implementation). The Yes value means that the session is re-created and unconfirmed PUBLISH messages are not repeated.	YES /NO	NO
Publish QoS	Level of confirmation (QoS) used to send PUBLISH messages through the D2000 KOM process. Sending the PUBLISH message is the outcome of writing into the output tag with the OUT_VALUE address. The higher the confirmation level, the more messages between the client and server are exchanged (1 for QoS_0, 2 for QoS_1, and 4 for QoS_2).	QoS_0 QoS_1 QoS_2	QoS_0
Publish Retain	Setting the Retain flag used when sending PUBLISH messages by the D2000 KOM process. Activating the Retain flag causes the last message sent by the D2000 KOM process to be available on the MQTT server to other clients immediately after they are connected, as well as after the D2000 KOM process is disconnected.	YES /NO	NO
Ping Interval	If the MQTT server did not send any message during the specified time interval, the D2000 KOM process sends a PING request and waits for a PING response (until time Reply Timeout). A value of 0 turns off sending the PING request messages. The parameter allows detection of TCP connection failure.	sec	60

Payload Type	The setting of message parsing: • Text only - the message is not parsed, it is assigned to the I/O tag with address IN_TOPIC • JSON - the message is parsed as JSON data. If there is an I/O tag with address IN_TOPIC, the whole message will be assigned to it. If there are I/O tags with addresses JA=json_address, they will be populated with the appropriate data from the JSON message. If no such addresses exist in the message, the I/O tags will be invalidated.	Text only JSON	Text only
Time Field Name	If Payload Type=JSON, the name of the field with a timestamp. If the field name is not specified or the field is not found, the current time is assigned to the values. For more information on the field name format, see I/O tags with addresses JA=json_address.	-	-
Time Mask	Mask for parsing a value in the field with a timestamp. **Note:** from settings of time station parameters depends whether the time is interpreted as local or UTC with configured offset. Special masks are: • **UNIX** - the numeric value represents the number of seconds from epoch 00:00:00 01.01.1970 UTC. • **UNIXMS** - the numeric value represents the number of milliseconds from epoch 00:00:00.000 01.01.1970 UTC.	-	yyyy-mm-dd hh:mi:ss.mss
Ignore Missing Time	Ignoring a missing timestamp - if it is not present in the JSON payload, no warning will be issued.	YES /NO	NO
Will Flag	Parameter Will Flag of a CONNECT message. A value of Yes means that the server will send a Last Will message to interested parties if the connection to the D2000 KOM process is lost.	YES /NO	NO
Will QoS	The acknowledgment level (QoS) used when sending a Last Will message in the event of a loss of connection to the D2000 KOM process.	QoS_0 QoS_1 QoS_2	QoS_0
Will Retain	The setting of the Retain flag used when sending a Last Will message if the connection to the D2000 KOM process is lost.	YES /NO	NO
Will Topic	The topic used to send the Last Will message if the connection to the D2000 KOM process is lost.	-	
Will Message	Contents of the Last Will report if the connection to the D2000 KOM process is lost.	-	
Reply Timeout	If the MQTT server does not respond to the SUBSCRIBE, UNSUBSCRIBE, and PING requests within the required time or the D2000 KOM process fails to read a complete message (and only part of it is read), the D2000 KOM process declares an error, closes the connection, and opens it again. Value 0 turns off the timeout. The parameter enables the handling of problematic behavior of the MQTT server.	sec	20
Wait Timeout	A timeout of a single reading from a TCP connection. D2000 KOM repeats reading of spontaneous data Max. Wait Retry times and if no data is read, the reading is timeouted and finished (and may be followed by a further reading or writing). By lowering Wait Timeout and Max. Wait Retry parameters, it is possible to achieve a faster writing response of the D2000 KOM process at the expense of a higher CPU load when the MQTT server has no data. Note: if a lot of messages come from the MQTT server and the D2000 KOM also needs to write values, we recommend setting a lower parameter value (e.g. 0.005 sec) so that writing is not blocked by reading (in any case, after 10 received messages, there is an interruption during which the accumulated writes can be performed).	sec	0.100
Max. Wait Retry	The number of repetitions of reading from TCP connection. See the description of the Wait Timeout parameter.	-	3

Communication station configuration

- Communication protocol "**MQTT Client Protocol**".
- Station address: the station address corresponds to the Topic field in the PUBLISH message received from the MQTT server. The address can be a specific Topic, a regular expression, a # character representing all Topics, or a topic **.*** representing all Topics that are not suitable for other stations. The processing priority is as follows:
  - If there is a station with address # on the line, all messages are directed to its I/O tags and no further search is performed.
  - Otherwise, all other stations on the line are searched (with the exception of the **.*** address). If the Topic matches the address of a station, the message is directed to that station and no further search is performed.
  - Otherwise, all other stations on the line are searched (with the exception of the **.*** address), and their address is evaluated as a regular expression. If the Topic matches the station address, the message is directed to that station and no further search is performed.
    Stations are searched in descending order (by station address), so more specific terms go first (e.g., *status/battery* before *status/batt.**)
  - Finally, if there is a station with a **.*** address, the message is addressed to it.
- Polling parameters on the *Time parameters* tab - recommended value is Delay=0.

I/O tag configuration

Possible value types of I/O tags: **Ci**, **Co**, **TxtI**, **TxtO, Qi, Ci, Co, Ai, Ao, Di, Do, TiR, ToR, TiA, ToA**.

Type of I/O tag	Address	Description
I/O tags for reading data sent by MQTT server through PUBLISH message. **Note:** values of I/O tags are set by the D2000 KOM process in the order IN_TOPIC, IN_DATA and IN_ID. It is not necessary for configuration to contain all three I/O tags.		

TxtI	IN_TOPIC	Topic (Topic) of received PUBLISH message.
TxtI	IN_DATA	Data (Payload) of received PUBLISH message.
Ci	IN_ID	Identifier of a packet (Packet Identifier) of PUBLISH message that depends on the level of validation (QoS). For messages sent with QoS_0, the identifier is zero, for QoS_1 and QoS_2, it is a positive 16-bit number. **Note:** if the MQTT server sends also messages with the QoS_0 level of validation and the ACK_ID I/O tag is configured, then we recommend activating the option *New value when changing time* in the *Filter* tab, so that repeated writing of the value 0 will cause a new value that differs only in a timestamp to be generated.
I/O tags for parsing JSON messages		
TxtI, TxtO, Qi, Ci, Co, Ai, Ao, Di, Do, TiR, ToR	JA=*json_address*	If Payload Type=JSON, the message is parsed as JSON data. The *json_address* value specifies the name of the JSON field whose value is to be assigned to the I/O tag. For JSON messages that can be structured, the syntax *level1.level2.level3* ... is supported, e.g. *rx.current*, and if they contain fields (indexed from 1) also syntax *level1[index1].level2[index2].level3* ... is possible, e.g. *rx.gwrx[1].time*. Since the JSON message itself can be an array, the address can also start with an index, e.g. *JA=[1].batt_cell_v_avg* For other examples, see the description of the LoRaWAN protocol's Envelope type I/O tags.
I/O tag to confirm the received data to the MQTT server.		
Co	ACK_ID	If an output I/O tag with ACK_ID address is defined, the D2000 KOM expects confirmation of the processing of each message by writing a copy of the value of the IN_ID tag. Only after, it sets values from the next received PUBLISH message (if it was received in the meantime) into the IN_TOPIC, IN_DATA, and IN_ID I/O tags (in this order). In the case of the QoS_0 level of confirmation, it is, therefore, necessary to repeatedly set the value of I/O tag ACK_ID to 0. If the I/O tag ACK_ID does not exist, the values are written into the IN_TOPIC, IN_DATA, and IN_ID I/O tags immediate ly after the PUBLISH message is received and processed. **Note:** for the messages received with the QoS_0 level of validation, no confirmation is sent to the MQTT server, only the values of the received PUBLISH message will be published.
I/O tags for sending values to the MQTT server through PUBLISH message. **Note:** in order for the D2000 KOM process to send the PUBLISH messages to the MQTT server, both I/O tags must be defined within one station.		
TxtO	OUT_TOPIC	The topic of the PUBLISH message being sent.
TxtO	OUT_VALUE	Data (Payload) of the PUBLISH message being sent. **Note:** sending the message is performed out as a result of writing to the OUT_VALUE I/O tag (i.e. if the Topic does not change then it is sufficient to set the OUT_TOPIC point once - e.g. by using default value).

Literature

Links
Official website of MQTT protocol http://mqtt.org

Specifications and Standards
MQTT 3.1.1 specification http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
ISO/IEC 20922:2016 http://www.iso.org/standard/69499.html

Descriptions of Data Formats and API
www.loriot.io - Application API Data Format https://www.loriot.io/home/documentation.html#docu/app-data-format
www.thethingsnetwork.org - API Reference https://www.thethingsnetwork.org/docs/applications/mqtt/api.html

Blog

You can read a blog about the MQTT protocol

- Communication - MQTT in practice

Document revisions

- Ver. 1.0 - August 8th, 2017 - document creation.
- Ver. 1.1 - October 15th, 2021 - support LastWill and Retain parameters
- Ver. 1.2 - October 27th, 2021 - support for parsing of JSON messages
- Ver. 1.3 - February 1st, 2022 - support for timestamps in JSON messages

Súvisiace stránky:

Communication Protocols