

MQTT

USER MANUAL

Translation of the original instructions

Version: **1.0**

Date: **06/06/2023**

Index


1.	MQTT Configuration	4
	General parameters - MQTT Configuration	4
	MQTT function	6


VERSION	DATE	CHANGES
1.0	06/06/2023	-

Any information inside this manual can be changed without advice.

This handbook can be download freely from the website:
www.eelectron.com

Exclusion of liability:
Despite checking that the contents of this document match the hardware and software, deviations cannot be completely excluded. We therefore cannot accept any liability for this.
Any necessary corrections will be incorporated into newer versions of this manual.

Symbol for relevant information 

Symbol for warning 



1. MQTT Configuration

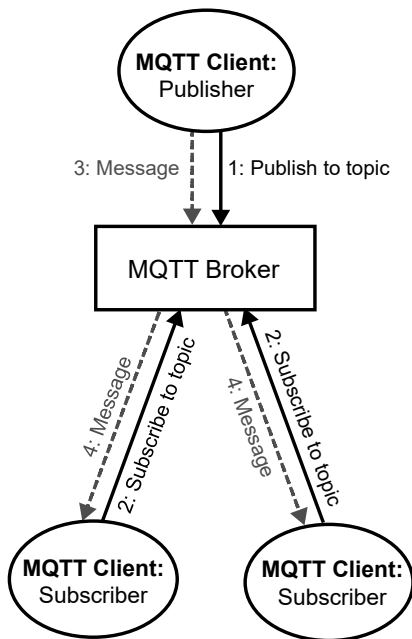
MQTT is a **messaging protocol** designed for transferring messages and based on a publish and subscribe model.

MQTT is often used in Internet of Things (IoT) and Machine-to-Machine (M2M) environments to allow devices to communicate with each other and with back-end servers.

MQTT is useful for enabling efficient and reliable communication between devices, particularly in environments where device resources are limited.

In MQTT a publisher publishes messages on a topic and a subscriber must subscribe to that topic to view the message.

MQTT requires the use of a central Broker and its as shown in the diagram below:



Main features:

- MQTT uses **TCP/IP** to connect to the broker.
- MQTT clients publish a **keepalive message** at regular intervals which tells the broker that the client is still connected
- Clients **do not have addresses** like in email systems, and messages are not sent to clients.
- Messages are **published to a broker on a topic**.
- The job of an MQTT broker is to **filter messages** based on topic, and then **distribute them to subscribers**.
- A client can receive these messages by subscribing to that topic on the same broker
- There is no direct connection between a publisher and subscriber.
- All clients can publish (broadcast) and subscribe (receive).
- MQTT brokers do not normally store messages.

MQTT can be used for:

- Transmit data between KNX or IoT devices via backend servers, for example to collect data from sensors or control actuators.
- Trigger real-time actions, such as notifying an app when a sensor detects a change in the state of the environment.
- Support two-way communication between devices, for ex-

ample to allow an app to send commands to a KNX or IoT device.

- Manage data security through the use of authentication and encryption.

General parameters - MQTT Configuration

Communication object involved:

"<General > Alarm Broker Connection"	1 bit	CRT
--------------------------------------	-------	-----

KNX PARAMETER	SETTINGS
Connection	
Client ID	ee%S
A Client is any device that runs an MQTT library and connects to an MQTT broker over a network. The Client ID is a unique identifier used in MQTT to identify each client connecting to the broker. The client ID is used by the broker to identify the messages sent by each client and to send the messages to the appropriate clients. The client ID must be unique for each client connecting to the broker, otherwise the broker may confuse messages sent by different clients with the same ID.	
Broker address	64 bytes allowed
An MQTT broker is a server that receives all messages from the MQTT clients and routes them to the appropriate destination clients. This parameter identifies the address of the MQTT broker to connect to.	
Broker port	1 ... 65535
It's the broker port used for connection (the default value is 8883).	
Broker connection interval [s]	0 ... 255
It defines the time interval during which the device tries to connect to the broker. In case of failure it waits that time to reconnect.	
Use TLS	no / yes
TLS (Transport Layer Security) is a security protocol that is used to secure communications on a network. TLS works by adding a layer of encryption to network communications so that only the sender and receiver can read the contents of the communications. In this way, TLS protects communications from being intercepted or modified by third parties.	

Validate certificate	no / yes
<p>The “Validate certificate” option for TLS (Transport Layer Security) is used to verify that the security certificate used by the server to which you are connecting is valid and reliable.</p> <p>When using TLS to establish a secure connection with a server, the server sends its security certificate to the client. The security certificate includes information about the identity of the server, such as the domain name and the name of the organization that manages it, as well as a cryptographic public key used to secure communications.</p> <p>The “Validate certificate” option allows you to verify that the security certificate sent by the server is valid and that it has not been altered or modified in any way.</p> <p>It should be considered that in some cases or states (communication debug) this option helps the configurer to carry out communication tests while maintaining an encrypted channel.</p> <p>It is recommended to activate this option when you finish configuring the device.</p> <p>The uploading of the certificates is managed by the software “Eelectron Certificate Loader” property of eelectron. Please refer to the specific user manual “Eelectron Certificate Loader”.</p>	
Username	
It's the username for authentication to the broker.	
Password	
It's the user password for logging in to the broker.	
Keep alive interval [s] (0 = disabled)	1 ... 65535
<p>The keep alive is a time interval used to manage the connection between the client and the broker. The keep alive is set by the client when it connects to the broker and is used to verify that the connection is still active.</p>	
Clean session	no / yes
<p>Clean session is a flag to indicate whether the client should keep or delete undelivered messages and subscriptions during logoff.</p> <ul style="list-style-type: none"> When clean session is set to yes, the client discards all undelivered messages and subscriptions when it disconnects from the broker. This way, the client does not receive messages sent during disconnection and must re-subscribe to the broker when it reconnects. When clean session is set to no, the client keeps undelivered messages and subscriptions during disconnection and receives them when it reconnects to the broker. This way, the client can receive all messages sent during the disconnect without having to retry them. <p>The value of clean session depends on the needs of the application and must be chosen appropriately to ensure good connection and message handling.</p>	
Will management	
<p>The “Will message”, also known as the “LWT message” (Last Will and Testament message), is a feature of the MQTT protocol that allows MQTT clients to specify a message to send automatically to other MQTT clients or to MQTT servers in case the client disconnects abnormally, without sending a disconnect message.</p> <p>The “will message” can be used for several purposes, for example:</p> <ul style="list-style-type: none"> Notify other devices or applications that a device has disconnected abnormally, so that other devices can take action accordingly. Update the MQTT server on the client's status so that the server can change its status and respond accordingly. Notify system operators that there is a problem with the device or application. 	

Will - topic	ee/%D/%S/status
<p>In MQTT, the topic name is a simple string that is hierarchically structured in levels that the broker uses to filter messages for each connected client. Each topic level is separated by a forward slash (topic level separator).</p> <div style="text-align: center;"> </div> <p>This parameter defines the topic for the will message.</p>	
Will - payload	30 bytes allowed
It defines the content of the message.	
Will - retain	no / yes
<p>If yes, the MQTT broker stores the last retained message and the corresponding QoS for the topic. Each client that subscribes to a topic pattern that matches the topic of the retained message receives the retained message immediately after they subscribe. The broker stores only one retained message per topic.</p> <p>If no, the message is not forwarded.</p>	
Will - Quality of Services (QoS)	0 / 1 / 2
<p>QoS (Quality of Service) is an attribute assigned to a single MQTT message, it is an agreement between sender and recipient that defines the way a message is delivered and transmitted. Allows clients to consider network reliability. The broker and client are able to retransmit messages and guarantee delivery, facilitating communication in unreliable networks.</p> <ul style="list-style-type: none"> (0) A message is delivered at most once. It may not be delivered at all. (1) The message is always delivered at least once. Receipt of the message must be confirmed. Failure to receive an acknowledgment will result in the message being resent. This process repeats until the message is acknowledged and can lead to the same message being sent and processed multiple times. (2) Sent messages are always delivered exactly once. It is the slowest and most reliable mode of transfer in an MQTT network. At least two transmission pairs are performed between the sender and the recipient before a message is deleted by the sender. <p>For more information consult: https://mqtt.org/</p>	
Birth management	
<p>The “Birth message” is a message that is automatically sent by the MQTT client to the MQTT broker when the client successfully connects to the broker. Unlike the “will message”, the “birth message” is sent only once, when the client connects to the MQTT broker.</p> <p>The “birth message” can be used for various purposes, for example:</p> <ul style="list-style-type: none"> Notify the MQTT broker or other MQTT clients that the client has just connected and is ready to receive messages. Update the MQTT broker or other MQTT clients on the status of the newly connected client. Send client information, such as name, ID, or other configuration information. 	
Birth - topic	ee/%D/%S/status
This parameter defines the topic for the birth message.	
Birth - payload	30 bytes allowed
It defines the content of the message.	
Birth - retain	no / yes
See “ Will - retain ”	
Birth - Quality of Services (QoS)	0 / 1 / 2
See “ Will - Quality of Services (QoS) ”	
General Topic	

Prefix	ee/%D/%S
The topic prefix is used to create a hierarchy of subtopics, for example to send messages to devices in a certain geographic area or to devices that belong to a certain category. For example, the prefix "home/room 1/" could be used to send messages only to devices in room 1 of a house.	
Subscribe rule	
Adds a level to the topic, which depends on the Rule position parameter, which identifies whether the message is of the command type (Received from the broker and sent to the bus).	
Publish rule	
Adds a level to the topic, which depends on the Rule position parameter, which identifies if the message is of type status (Received from the bus and sent to the broker).	
Rule position	end of topic / end of prefix
It's where to add the level that identifies the type of message.	
Minimum delay between MQTT object messages [ms]	30 / 40 / 50 / 75 / 100 / 150 / 200 / 250
It defines the minimum time interval that elapses between two MQTT messages.	
String pattern x (x = 1 ... 8)	16 bytes allowed
It allows to associate a string to a pattern x (1...8). The string can be call back into topics by inserting "%x" .	
Example Pattern 1 = home Topic= ee/%D/%S/%1/cmd Result = ee/bridge/006c12345678/home/cmd	
Alarm Configuration	
Broker connection alarm	disabled / enabled
It enables the object "<General > Alarm Broker Connection" to notify if the connection with broker is active.	
Alarm telegram	telegram "0" / telegram "1"
It defines the telegram sent on the object "<General > Alarm Broker Connection" when alarm is active.	

MQTT function

Communication object involved:

"<MQTT x > Object"	1 bit ... 4 bytes	CRT/CW/CRWT
----------------------	-------------------	-------------

KNX PARAMETER	SETTINGS
MQTT function	none subscribe (from broker to BUS) publish (from BUS to broker) subscribe/publish
With this parameter it's possible to set the mode of use of MQTT : none : not use subscribe (from broker to BUS) : the device subscribes to the topic and sends the payload of messages received from the broker in string format to the bus, according to the set DPT. Object flags: CRT publish (from BUS to broker) : the device publishes the topic to the broker and creates the payload by transforming the value of the object into a string. Object flags: CW Example for publish Prefix = ee/%D/%S Publish role = /sts Role position = end of topic Mqtt topic = /1 Value = 1 The final topic will be: ee/%D/%S/1/sts with payload 1 with Role position = end of prefix The final topic will be: ee/%D/%S/sts/1 with payload 1 subscribe / publish : Automatically create two topics following the rules described above for subscribe and publish Object flags: CRWT	
MQTT name	16 bytes allowed
This parameter defines the name of the MQTT module; the name can be used to rapidly identify the functionality.	
MQTT topic	30 bytes allowed
It identifies the topic. Please refer to: "Eelectron Certificate Loader" page 5 .	
DPT size	
This parameter defines the DPT of the MQTT function. The telegram can be: <ul style="list-style-type: none"> • 1 bit • 4 bit • 1 byte (signed, unsigned) • 2 bytes (signed, unsigned, float) • 4 bytes (signed, unsigned, float) 	
DPT type	0 - 255 0 - 100% angle HVAC mode
In case of 1 byte unsigned, it specifies the DPT of the object.	
Retain	no / yes
See " Will - retain ".	

Quality of Services (QoS)	0 / 1 / 2
<p>QoS (Quality of Service) is an attribute assigned to a single MQTT message, it is an agreement between sender and recipient that defines the way a message is delivered and transmitted. Allows clients to consider network reliability. The broker and client are able to retransmit messages and guarantee delivery, facilitating communication in unreliable networks.</p> <ul style="list-style-type: none"> • (0) A message is delivered at most once. It may not be delivered at all. • (1) The message is always delivered at least once. Receipt of the message must be confirmed. Failure to receive an acknowledgment will result in the message being resent. This process repeats until the message is acknowledged and can lead to the same message being sent and processed multiple times. • (2) Sent messages are always delivered exactly once. It is the slowest and most reliable mode of transfer in an MQTT network. At least two transmission pairs are performed between the sender and the recipient before a message is deleted by the sender. <p>For more information consult: https://mqtt.org/</p>	

Example

Let's assume the following data model:

- temperature data communication from BUS KNX to MQTT client.
- communication of the temperature setpoint from the MQTT client to the KNX BUS.
- 1 bit on/off light command communication.
- communication of a percentage command for shutters control

Step 1 : ETS configuration

https://download.eelectron.com/ETS%20-%20configuration_eg.mp4

Step 2 : MQTT client configuration

https://download.eelectron.com/MQTT%20Explorer_eg.mp4



The value sent from MQTT client to BUS KNX is the absolute value of the communication object e.g. command % 100% = 100 , command RGB FFF = FFF)

Only in case of 4 bit DPT, the value to write on the MQTT client must have this formatting:

[0 - 1] ; [0 - 7], where 0 is decrease, 1 is increase and 7 :

0	break	
1	1	100.0%
2	2	50.0%
3	4	25.0%
4	8	12.5%
5	16	6.3%
6	32	3.1%
7	64	1.6%



For a detailed analysis of the data, there are dedicated platforms where you can register the device.