MQTT Client Connector

Devices compatible with the MQTT protocol

- Name: **MQTT**
- Version 3.0.3.5
- Protocol: MQTT
- Interface: TCP/IP
- Runtime: Multiplatform
- Configuration:
    - Devices / Protocols

Channel Configuration
Node Configuration
Station Examples
Points Configuration
Driver Revision History

Module Configuration

This document has only information about the device connection settings and its address syntax.

For further Information, see:

- Devices Module
- Devices Module Reference

Channel Configuration

Protocol Options

- `DrvParams[0]` = **TimePublishRate** (integer, milliseconds): Minimum publish rate for MQTT messages. Minimum value: 50ms. Default: 500ms.
- `DrvParams[1]` = **StoreAndForward** (boolean): Enables store-and-forward mode for reliable message delivery. Default: false.

Node Configuration

Station Configuration

Station syntax: **<BrokerURL>; <Port>; <ClientID>; [Username]; [Password]; [NetworkSecurity]; [X509Certificate]; <Qos>; <KeepAlive>**

Where:

- ***<BrokerURL>***: MQTT Broker (Server) name. It must be the same name is configured in the ***bind_address*** parameter in the Broker.
- ***<Port>***: MQTT Broker port. It must be the same port is configured in the Broker to be listening. Default value is **1883**.
- ***<ClientID>***: Indicates the client identification used to connect to the MQTT Broker.
- ***[Username]***: Optional. Username defined on MQTT Broker. This will be requested if the Broker has the *allow_anonymous* parameter as false. Whether this parameter is set as true on Broker, keep this field empty. Default value is empty.
- ***[Password]***: Optional. Password defined on MTQQ Broker. This will be requested if the Broker has the *allow_anonymous* parameter as false. Whether this parameter is set as true on Broker, keep this field empty. Default value is empty.
- ***[NetworkSecurity]***: Optional. If the Broker works using certificates, this field is mandatory and must match the same option defined in the Broker. Options are: None, TLSv1.0, TLSv1.1 and TLSv1.2.
- ***[X509Certificate]***: Optional. Path of X509 Certificate using TLS v1.0. This must be the complete path of X509 certificate in the client computer. The certificate must be installed in the computer.
- ***<QoS>***: Quality of Service (QoS) is an agreement between sender and receiver of a message regarding the guarantees of delivering a message. There are three QoS levels:
    - ***AtMostOnce****: This service level guarantees a best-effort delivery. However, there is no guarantee of delivery, due to the recipient not acknowledging the receipt of the message.*
    - ***AtLeastOnce****: This service guarantees that a message is delivered at least one time to the receiver. Default Value.*
    - ***ExactlyOnce****: Highest level of service. It is safer, but slower when compared to the others. When the message flow is complete, all parties involved in the communication are sure the message was sent and delivered with success.*

Choosing the Right QoS Level

Use QoS 0 when:

- The connection between sender and receiver is stable (e.g., a test client or front-end application connected via a wired connection).
- Occasional message loss is acceptable (e.g., when sending non-critical data at short intervals).
- Message queuing is **not** required.

Use QoS 1 when:

- Every message must be received, but duplicates are acceptable.
- The application **does not require** the overhead of QoS 2 but still needs message delivery guarantees.
- Faster delivery is preferred compared to QoS 2.

Use QoS 2 when:

- Receiving **every message exactly once** is critical (e.g., when duplicate messages could cause issues).
- Applications need the highest level of reliability, despite the additional overhead and latency.

- ***<KeepAlive>***: Ensures that the connection between the Broker and Client is open and both parties are aware of the connection. Default value is ***10 seconds***.
- <***Report Out of Range By Exception***>*: Not used by this connector.*
- <***Disable Rebirth Request***>*: Prevents sending NCMD and DCMD commands to the MQTT broker.*

When creating solutions that the PrimaryStation IP will be set by script, using the Node properties, you can set the Broker URL to 1.1.1.1 to force it to try establish a connection, avoiding timeouts.

NOTE

One of the easiest ways to install the certificate on client computer is to use the wizard to import the certificates is through the "Internet Options" to import the certificate in the "Trusted Root Certification Authorities". You need to import the certificate in DER format.

Station Examples

1. Broker configuration with no Username and Password and no SSL/TLS

Station Configuration

BrokerURL: test.mosquitto.org

Port: 1883

ClientID: MyClient1

Username: <*empty*>

Password: <*empty*>

NetworkSecurity: None

X509Certificate: <*empty*>

QoS: AtLeastOnce

KeepAlive: 10

2. Broker configuration with Username and Password and no SSL/TLS

Station Configuration

BrokerURL: MSI

Port: 1883

ClientID: MyClient1

Username: Guest

Password: 1234

NetworkSecurity: None

X509Certificate: *<empty>*

QoS: AtLeastOnce

KeepAlive: 10

3. Broker configuration with SSL/TLS and no Username and Password

Station Configuration

BrokerURL: MSI

Port: 8883

ClientID: MyClient1

Username: *<empty>*

Password: *<empty>*

NetworkSecurity: TLSv1.0

X509Certificate: C:/Certificates/mqtt_ca.der

QoS: AtLeastOnce

KeepAlive: 10

4. Broker configuration with SSL/TLS and Username and Password

Station Configuration

BrokerURL: MSI

Port: 8883

ClientID: MyClient1

Username: Guest

Password: 1234

NetworkSecurity: TLSv1.2

X509Certificate: C:/Certificates/mqtt_ca.der

QoS: AtLeastOnce

KeepAlive: 10

Points Configuration

Address

The syntax for the MQTT communication points is called ***Topic***:

- ***Topic*** is a string that is used by the broker to filter messages for each connected client. A topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator). Wildcards are **not** supported.

E.g.: ***USA/Houston/Station1/Level***

Driver Revision History

MQTT Revision History

Version	Notes
3.0.3.5	TLS 1.1 and 1.2 connections now work and connection failures report a plain-language cause; invalid broker credentials no longer cause high CPU or leave the device process running after stop; the broker user name is read from the correct field; new Tag Provider option Disable Rebirth Request; client certificates can be picked from the station editor and are found in MachineSettings/Certificates on the runtime host; password fields support Secrets and the Tag Provider password is stored encrypted; Tag Provider branches without a payload no longer log read errors; Test Connection in remote engineering states it ran on the Designer machine.
3.0.0.3	Initial release on new documentation standards.

In this section...