Cheatography

# MQTT - Lightweight Messaging Protocol Cheat Sheet

by otavio021 via cheatography.com/35559/cs/11177/

## Basic details and Terminoly

| | |
|---|---|
| MQTT Control Packet | A packet of information sent across the network |
| Applic-ation Message | The data carried by the protocol |
| Topic Name | A label attached to an Applic-ation Message that can be subscribed by clients |
| Topic Filter | An expression used to express interest in one or more topics (can use wildcards) |
| Session | Stateful interaction between a client and a server |

**Note**: TCP ports 1883 (default) and 8883 (TLS) are commonly used for MQTT.

## Data Representations

| | |
|---|---|
| Bits | Bits in a byte, from 7 (MSB) to 0 (LSB). |
| Integers | Big-endian ordered, 16-bits |
| Strings | UTF-8 strings, prefixed by its length |

## Quality of Service

| | |
|---|---|
| At Most Once | 0 |
| At Least Once | 1 |
| Exactly Once | 2 |

## Topic Names

Must be at least 1 character long

Case sensitive

Can include spaces

Name structured divided by slashes

**Example**: `/news/ spo rts /europe`

## Wildcards

| | | |
|---|---|---|
| Multi level | # | Used to match any number of levels within a topic tree, including the parent level itself. |
| Single level | + | Used to match a single level within a topic tree. |
| Reserved | $ | Topics starting with the dollar sign ('$') are reserved for server purposes and should not be used by clients. |

**Notes**:
Multi-level wildcard must always be the last symbol on the filter. Either on its own or preceded by the topic level separator.

Single-level: can be used in conjunction with the multi-level one.

**Examples**:
`+/sports/#` - valid
`sports+` - not valid

## MQTT Control Packet Structure

The structure is formed by the aggregation of 3 sub-structures: fixed header, variable length header and payload.

## MQTT Control Packet Structure: Fixed Header

| Field | Description |
|---|---|
| Control Packet Type | 4 bit representation of the packet type. |
| Flags | 4 bit flags specific to each packet type. |

## MQTT Control Packet Structure: Fixed Header (cont)

| | |
|---|---|
| Remaining Length | Number of bytes remaining in the packet. |

**Note #1**: Control Packet Type and the Flags are stored in a single byte.
**Note #2**: The Remaining Length does not include the bytes used to encode itself

## MQTT Control Packet Structure: Variable Length

| Field | Description |
|---|---|
| Packet Identifier | Used to establish a relationship between different MQTT Control Packets |
| Payload | A payload associated with the MQTT Control Packet |

**Note**: for the PUBLISH control packet, the payload is the application message.

## MQTT Control Packets

| Packet | Name |
|---|---|
| CONNECT | Client request to connect |
| CONNACK | Connect ACK |
| PUBLISH | Publish message |
| PUBACK | Publish ACK |
| PUBREC | Publish received |
| PUBREL | Publish release |
| PUBCOMP | Publish complete |
| SUBSCRIBE | Client subscribe request |
| SUBACK | Subscribe ACK |
| UNSUBS-CRIBE | Unsubscribe request |
| UNSUBACK | Unsubscribe ACK |
| PINGREQ | Ping request |
| PINGRESP | Ping response |
| DISCONNECT | Client disconnecting |

By **otavio021**
cheatography.com/otavio021/

Published 15th March, 2017.
Last updated 15th March, 2017.


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

## CONNECT

**Important elements**:
- Connect Flags: to specify the behavior of the connection.
- Keep Alive: maximum time interval, in seconds, that can elapse between client transmission of control packets.

**Connect Flags**:
- Clean Session: controls the lifetime of the session state (0 to resume state, 1 to discard previous state).
- Will Flag: indicates that a will message is to be sent upon dirty client disconnection.
- Will QoS: indicates QoS level for the will message.
- Will Retain: indicates the retain policy for the will message.
- Password: indicates whether (1) or not (0) a password must be present in the payload.
- User Name: indicates whether (1) or not (0) a user name must be present in the payload.

**Payload**:
Length prefixed fields whose presence is determined according to the value of flags in the variable header. The fields are, in order: client identifier, will topic, will message, user name, password.

## CONNACK

Sent by the server in response to a connection request sent by a client. The important elements on the packet structure are: connect acknowledgement flags and the connect return code.

**Note**: if a session is already present and the connection request does not have the clean session, the server must set the session present flag to 1.

## CONNACK: Error Codes

| Code | Description |
|---|---|
| 0x00 | Connection accepted |
| 0x01 | Connection refused (protocol version) |
| 0x02 | Connection refused (identifier rejected) |
| 0x03 | Connection refused (server unavailable) |
| 0x04 | Connection refused (bad user/password) |
| 0x05 | Connection refused (Unauthorized) |
| 6-255 | Reserved |

## PUBLISH

This packet is used to transport application messages for a client to a server or from a server to a client.
The important elements on the packet structure are:
- DUP Flag: indicates whether this is the first time the message is being sent (0) or whether it might be a re-delivery attempt (1) of a previous message.
- QoS Level: quality of service for an application message.
- Retain Flag: indicates that the application message and its QoS must be stored in the server and delivered to future subscribers of that topic.

**Note**: The QoS flag affect how many messages can be stored on the server and sent to the client.

## Protocol Exchange: QoS 1

Client/Server protocol interaction:
1. Client → PUBLISH → Server
2. Client ← PUBACK ← Server

## Protocol Exchange: QoS 2

Client/Server protocol interaction:
1. Client → PUBLISH → Server
2. Client ← PUBREC ← Server
3. Client → PUBREL → Server
4. Client ← PUBCOMP ← Server

## Security

TLS is the recommended cryptographic protocol to be used with MQTT. Implementations should use port 8883.



Sponsored by **Readable.com**
Measure your website readability!
https://readable.com