MQTT Packet Format

(*https://openlabpro.com/guide/mqtt-packet-format/*)

In this article, we are looking into the MQTT packet format with relevant example packets.

MQTT is a binary-based protocol and has command and command acknowledgement format. Therefore, every time a client sends a command to the broker, the broker sends an acknowledgement. This communication protocol is actually based on the TCP/IP protocol. So first, there will be a TCP connection establishment and then there will be MQTT connection establishment and then the data transfer will occur. After that, TCP connection will be terminated.

As this is a command and command acknowledgement based protocol, for each function the client needs to send commands to the broker. And they are sent as packets.

When a client has to publish a message, following steps are made:

- The client has to establish a connection to the broker by sending a connect packet (with username and password if needed)
- The client has to wait for the acknowledgement to see if the connection is accomplished or if there is an error.
- The client will send a publish packet which will contain the topic name and message to be published.
- The client has to wait for the publish response packet depending on the QoS level.

  QoS 0: There won't be any response
  QoS 1: PUBACK – Publish acknowledgement response
  QoS 2:
    - Wait for PUBREC – Publish received.
    - Send back PUBREL – Publish release.
    - Wait for PUBCOMP – Publish complete.

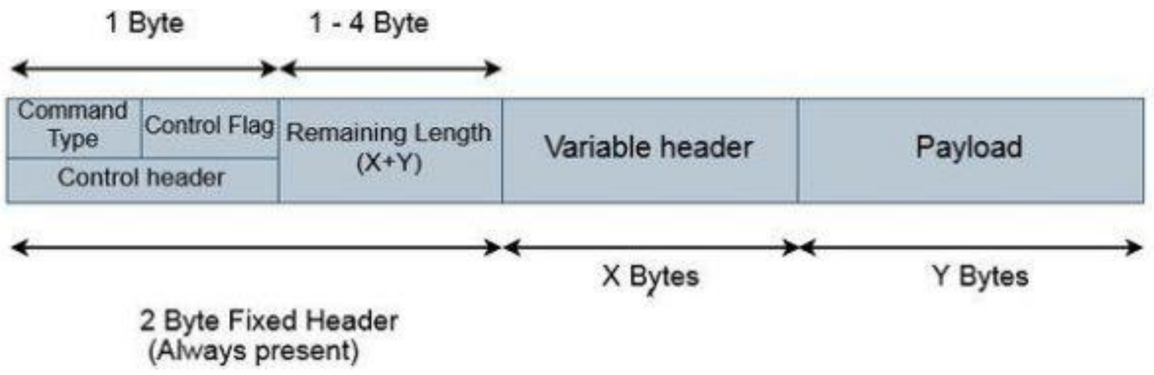
- If the communication is complete, the client can disconnect from the broker by sending a disconnect packet.

Similarly for subscribing to a topic:

- The client sends a Connect packet to the broker with username and password if needed)
- The client has to wait for the Connect acknowledgement packet from the broker to see if the connection is accomplished or if there is an error.
- When Connect acknowledgement is received, sends the subscribe packet with the appropriate topic name.
- The client has to wait for the subscribe acknowledgement packet.

MQTT Packet Format

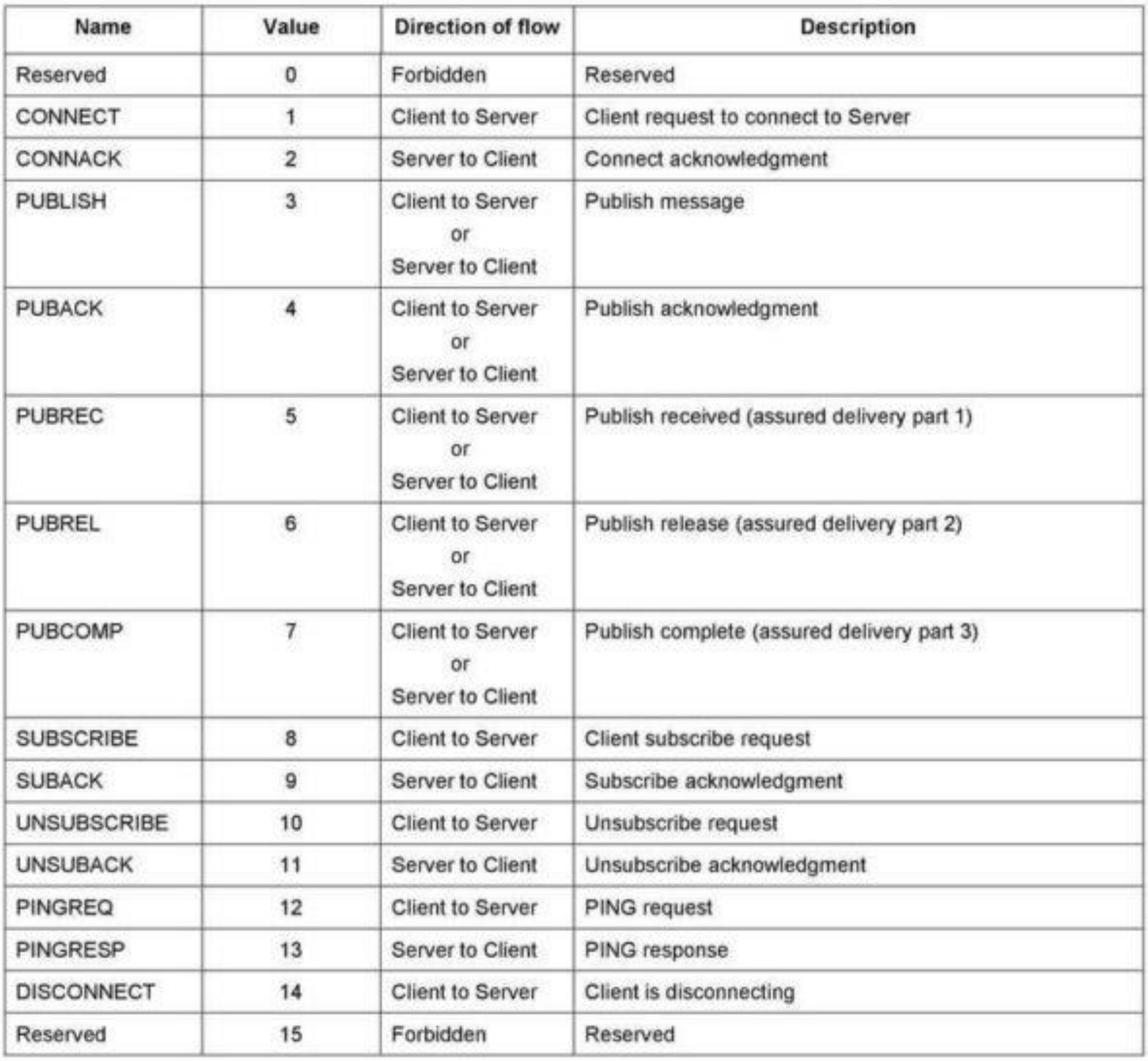
The MQTT packet consists of 2-byte fixed header plus a variable header and a payload. In this first 2-byte fixed header will be always present in all the packets and the other two, variable header and payload, are not always present.



Command type

Out of the two-byte fixed header, the first byte is the control field. This 8-bit control field is divided into two 4-bit fields. The first 4 MSB bits are the command type field.

E.g. The value of the connect command is 1. That means for connect command the connect type field should be 1 which is 0001. For publish command the value is 3, therefore, the connect type field should be 0011.



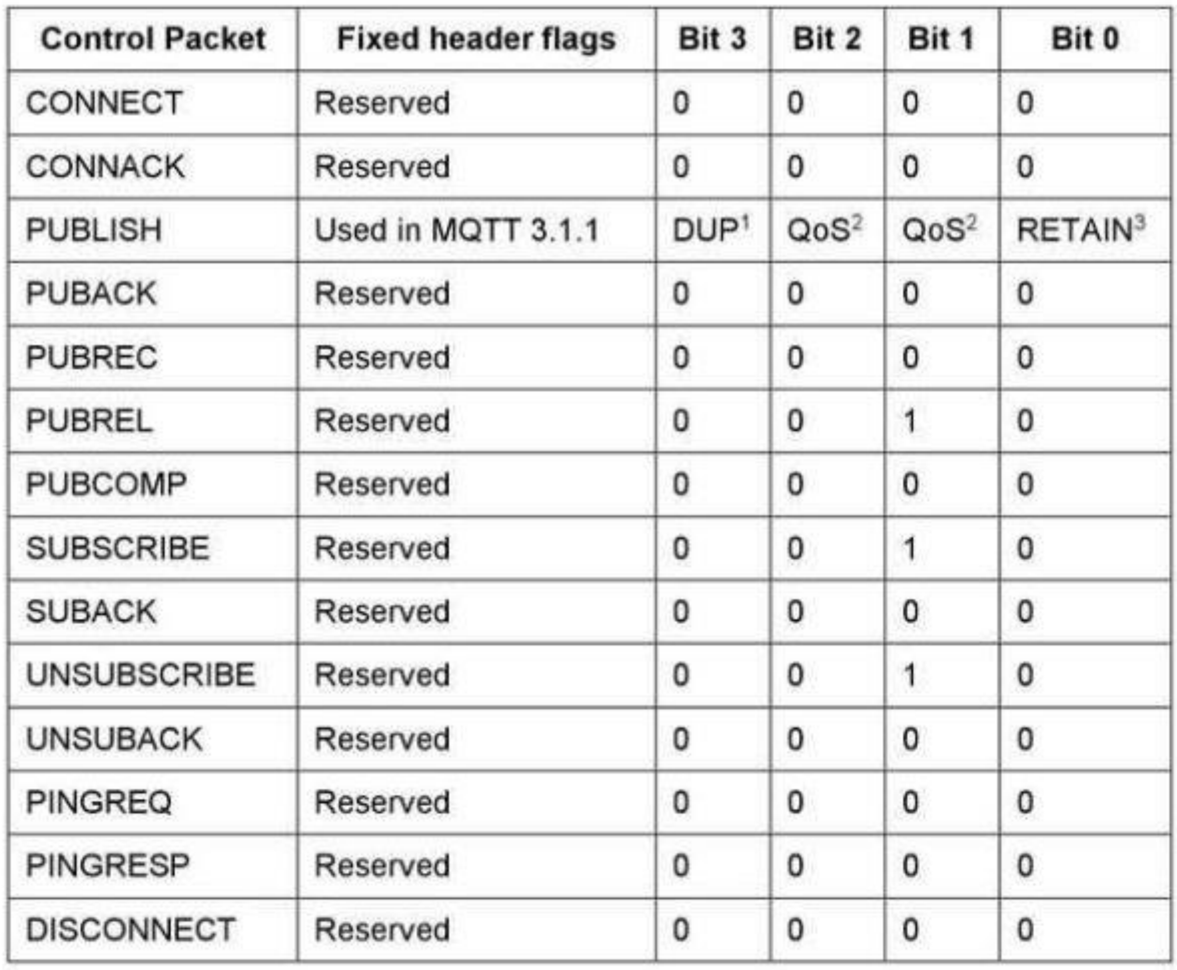
Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

Control Flag bits

The next 4 bits are the control flag bits and they are used by the publish command; for the rest of the commands they are reserved and the value will be 0.

For PUBLISH command:

- 0th bit denotes if the message that is published is to be retained.
- 1st and 2nd bits are used to select the quality of service if it is 0 or 1 or 2.
- 3rd bit denotes if it is a duplicate message.



Control Packet	Fixed header flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT 3.1.1	DUP[1]	QoS[2]	QoS[2]	RETAIN[3]
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0

Remaining Length

The second byte of the fixed header contains the remaining length, which is length of variable header plus the length of the payload. Remaining length can use up to 4 bytes in which each byte uses 7 bits for the length and the MSB bit being a continuation flag.
If the continuation flag bit of a byte is 1, it means the next byte is also part of the remaining length. Moreover, if the continuation flag bit is 0, it means that byte is the last one of the remaining length.

E.g. if the variable header length is 10 and the length of the payload is 20, the remaining length should be 30.

Variable header

A variable header is not present in all MQTT packets. Some MQTT commands or messages use this field to provide additional information or flags and they vary depending on the packet type. A packet identifier is common in most of the packets types. We will discuss in detail the variable header for the CONNECT packet below.

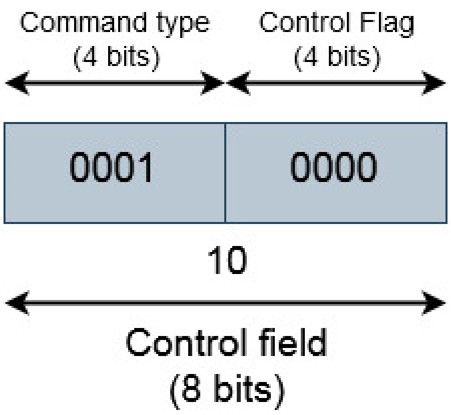
Payload

In the end, the packet may contain a payload. Even the payload is optional and varies with the type of packet. This field usually contains the data that is being sent.

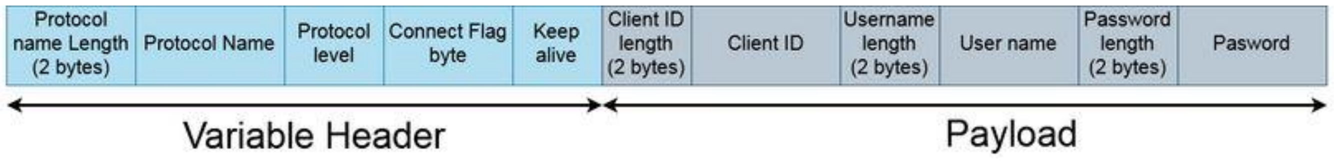
E.g. for CONNECT packet the payload is client ID and username and password, if they are present. And for PUBLISH packet, it is the message to be published.

CONNECT packet

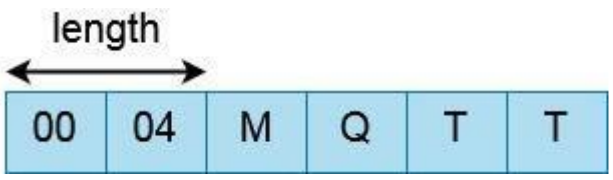
The first byte of the connect packet will be 10. Because the value of CONNECT command is 1, the first 4 MSB will be 1 and there are no flags so the next 4 bits will be 0.



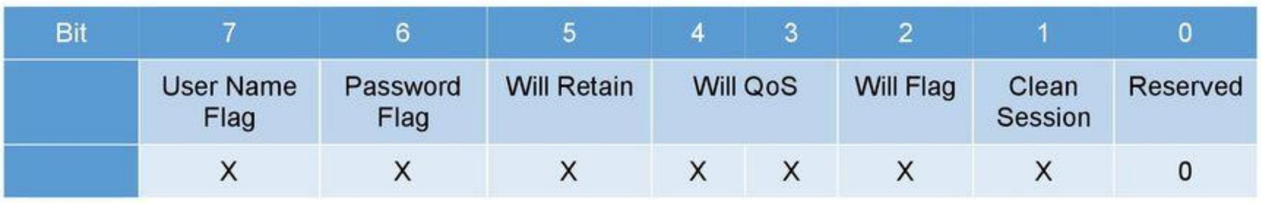
The second byte should be the remaining length, which is the length of the variable header and length of the payload. Let us decide this length after completing the variable header and payload. You can see below the format of the variable header and payload for CONNECT packet.


Protocol name Length (2 bytes)
Protocol Name
Protocol level
Connect Flag byte
Keep alive
Client ID length (2 bytes)
Client ID
Username length (2 bytes)
User name
Password length (2 bytes)
Pasword
Variable Header
Payload

In the variable header, there should be the protocol name. And for this, the first 2 bytes should mention the length of the protocol name followed by the protocol name. In our case, the protocol name is MQTT that is of length 4. So it becomes:



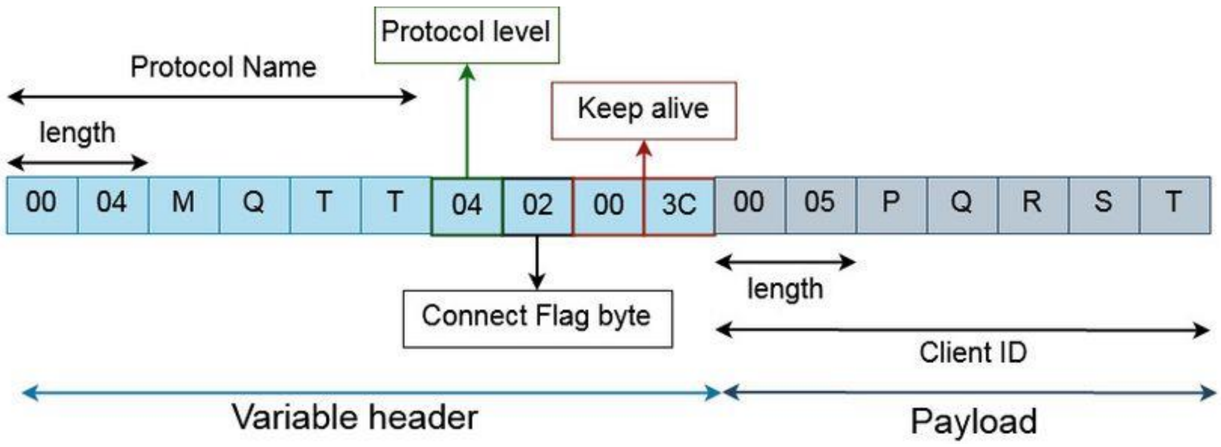
You cannot give whatever name you want here. As this name is used by the server to identify the MQTT traffic. If an invalid protocol is found, the server may reject the connection.
After the protocol name, there is the protocol level. This determines which version of MQTT it supports. For the version 3.1.1, the protocol is of level 4. And if the same protocol is not supported by the server, it disconnects by sending an acknowledgement with return code 01.
After that, there is the connect flag byte. We can access the broker ***test.mosquitto.org*** , which does not require any username and password as it, is open to the public. With a clean session and without a last-will message our CONNECT flag byte becomes 02. (0000 0010)



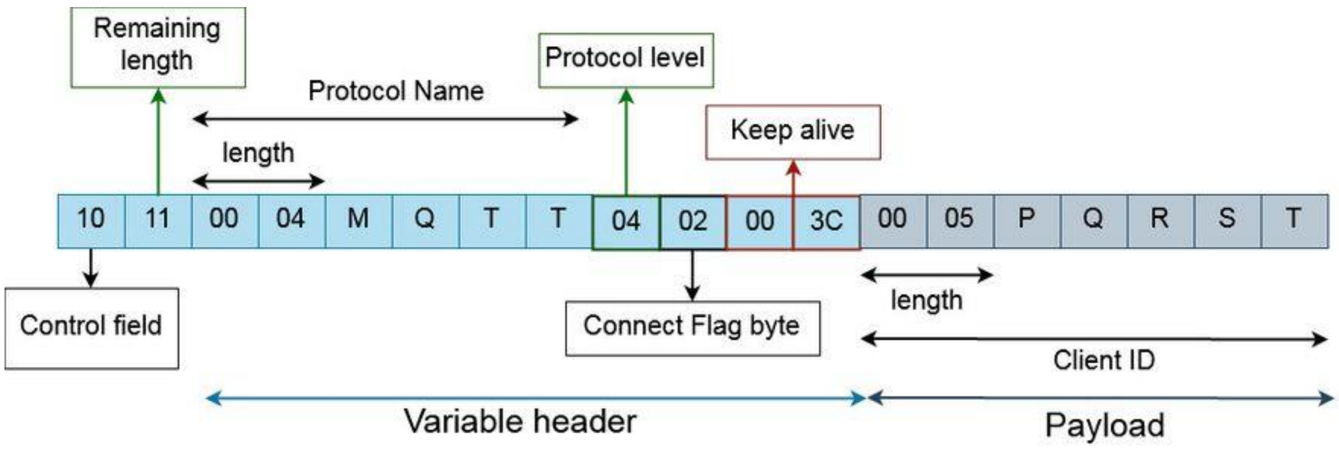
Bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
	X	X	X	X	X	X	X	0

Next two bytes are used to mention the keep alive duration in seconds. For 60 seconds, the value will be 003C in hex.

After the variable header, there will be the payload and it will contain client id, username and password. In our case there is no username and password, so only client Id will be present. Same as we did for protocol name, the first 2 bytes will denote the client id length. Let's assume our client id is PQRST.

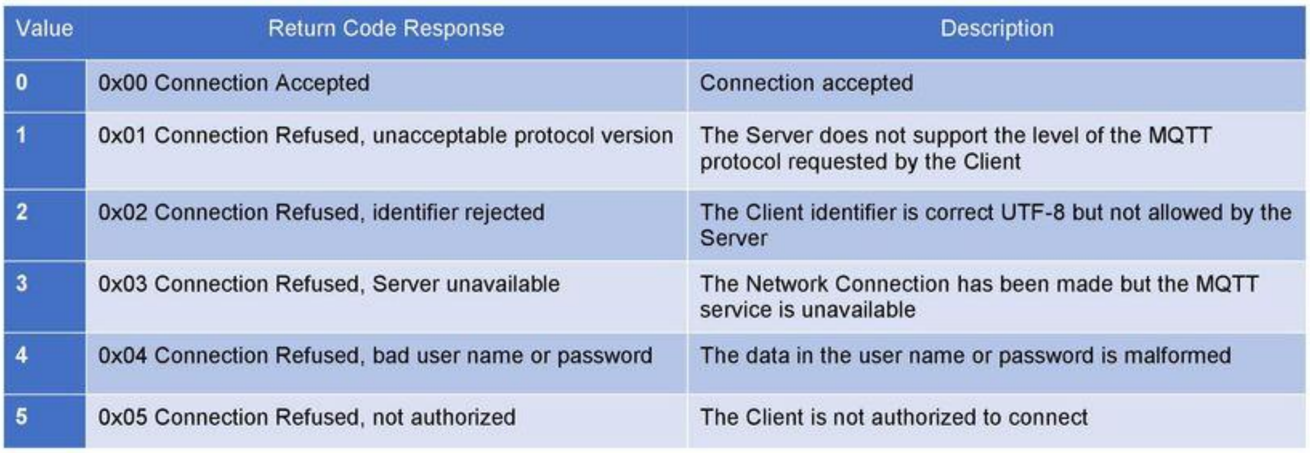


Now it's easy to determine the 'remaining length' which is supposed to be here. If you count the total bytes used for variable header and payload. It is 17. Therefore the 'remaining length will be 17 (hex 11).



CONNACK packet

Once the connect packet is sent, and if the broker receives the connection it will send back the acknowledgement – CONNACK. In the variable header of CONNACK there will be the connect return code. By reading that we can understand if the connection is established and if not, the reason behind rejection.

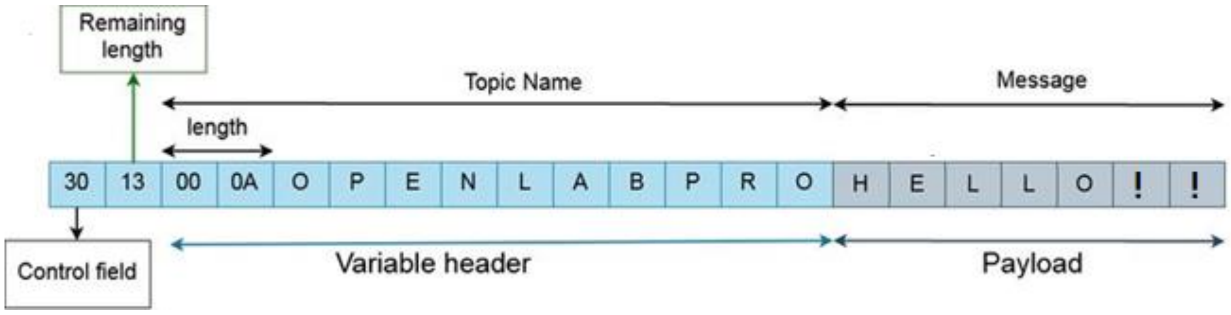


Value	Return Code Response	Description
0	0x00 Connection Accepted	Connection accepted
1	0x01 Connection Refused, unacceptable protocol version	The Server does not support the level of the MQTT protocol requested by the Client
2	0x02 Connection Refused, identifier rejected	The Client identifier is correct UTF-8 but not allowed by the Server
3	0x03 Connection Refused, Server unavailable	The Network Connection has been made but the MQTT service is unavailable
4	0x04 Connection Refused, bad user name or password	The data in the user name or password is malformed
5	0x05 Connection Refused, not authorized	The Client is not authorized to connect

PUBLISH packet

Now let’s publish the message “HELLO!!” to the topic OPENLABPRO.

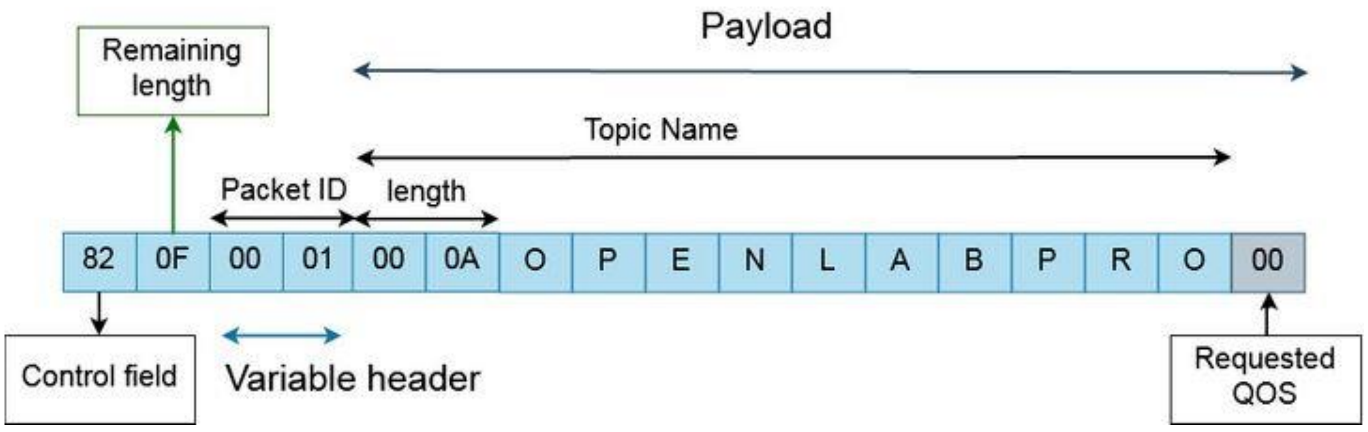
For the publish packet command value is 3. With QoS level 0 and without retaining the message control flag will be 0. In the variable header section, first 2 bytes will denote the length of the topic and then followed by topic.



*All numbers are in hex

SUBSCRIBE packet

Now the message is published and if there are any subscribers for that topic, they will receive the message. To subscribe to a topic the client has to send the SUBSCRIBE packet. Command value of Subscribe packet is 8 and the Control flag is reserved and should be 2. The variable header will contain a non-zero 16-bit packet ID. And as payload, there will be the topic to subscribe followed by requested QOS level.
To subscribe to the topic OPENLABPRO with QOS 0:



Now you have a clear picture of what data are sent in an MQTT protocol. And as you can see, all the commands and instructions are sent and received as bits. And that is why it is a binary-based protocol. Text fields such as username, password, topic etc. are encoded as UTF-8 strings. It uses one to four, 8-bit blocks to represent a character. Here, as the username and password are sent as raw data, the security is less. So SSL certification is preferred to be a better option.

You can find the complete packets and description at https://docs.oasis-open.org/mqtt/mqtt/