Updated:April 11, 2023 / By steve

Understanding the MQTT Protocol Packet Structure

In this tutorial we will take a more detailed look at the MQTT protocol, and how MQTT messages or packets are formatted.

We will be looking at:

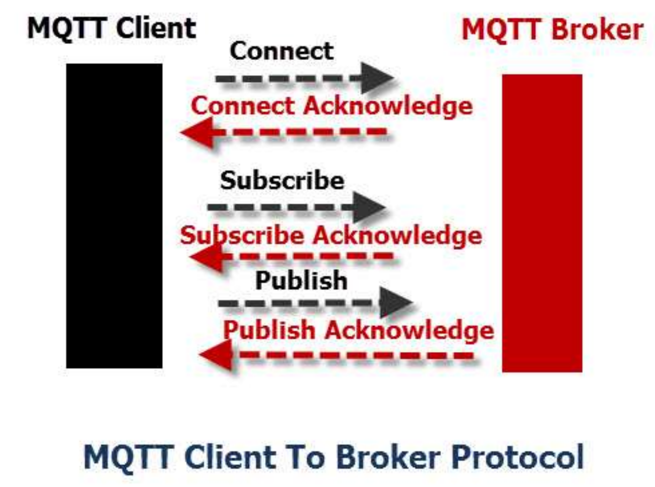
- The MQTT message format.
- The MQTT message header
- Message fields and coding
- Control Message coding example

Introduction

MQTT is a binary based protocol were the control elements are **binary bytes** and not **text strings.**

MQTT uses a **command** and **command acknowledgement** format.

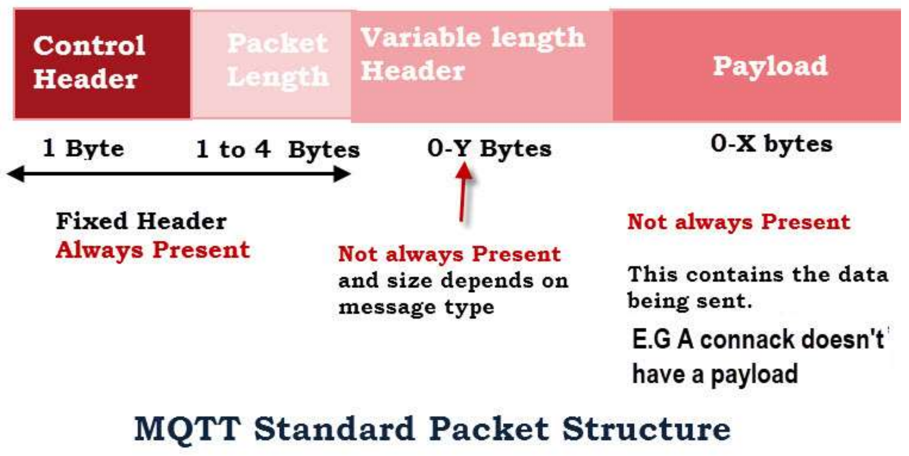
That means each command has an associated acknowledgement.



Topic names, Client ID, User names and Passwords are encoded as **UTF-8 strings**.

The **Payload** excluding **MQTT protocol information** like **Client ID** etc is binary data and the content and format is application specific.

The MQTT packet or message format consists of a **2 byte fixed header** (always present) **+** Variable-header **(not always present)+** payload **(not always present).**


Control Header
Packet Length
Variable length Header
Payload
1 Byte
1 to 4 Bytes
0-Y Bytes
0-X bytes
Fixed Header Always Present
Not always Present and size depends on message type
Not always Present
This contains the data being sent.
E.G A connack doesn't have a payload
MQTT Standard Packet Structure

Possible Packet formats are:

- Fixed Header (Control field + Length) – Example CONNACK
- Fixed Header (Control field + Length) + Variable Header -Example PUBACK
- Fixed Header (Control field + Length) + Variable Header + payload -Example CONNECT

The **fixed header field** consists of the **control field** and the variable length **packet length field.**

Hi - I'm Steve and welcome to my website where you can learn how to build IOT systems using MQTT.

Search

Search ...

- About Me
- MQTT Tools
- Networking

My Youtube Channel

- node-red
- MQTT Brokers
- mqtt and python
- Internet

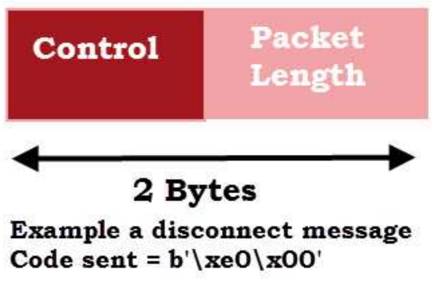
The **minimum size** of the **packet length field** is **1 byte** which is for messages with a **total length** less than 127 bytes.(not including control and length fields).

The maximum packet size is 256MB. Small packets less than **127 bytes** have a **1 byte** packet length field.

Packets larger than 127 and less than 16383 will use 2 bytes. etc.

Note: 7 bits are used with the **8th bit** being a **continuation bit.**

The **minimum packet size** is just **2 bytes** with a **single byte control field** and a **single byte packet length field**. E.g the **disconnect message** is only 2 bytes.

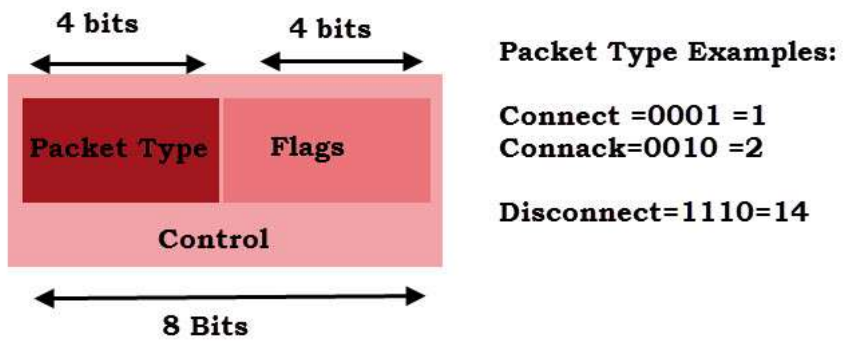


MQTT Minimum Packet

The Control Field

The 8 bit **control field** is the first byte of the **2 byte fixed header**. It is divided into two **4 bit fields**,and contains all of the protocol commands and responses.

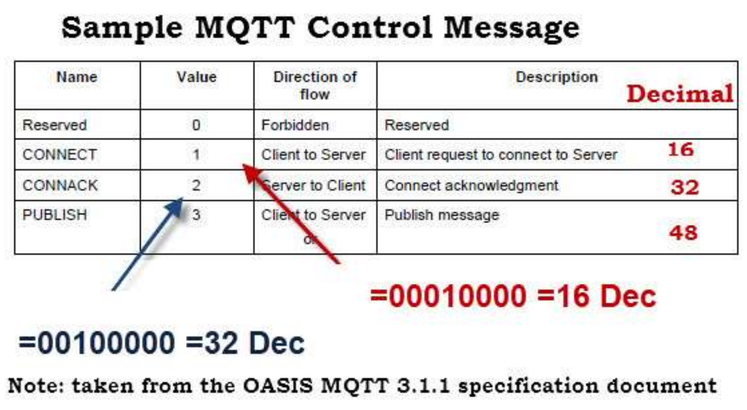
The first **4 Most significant bits** are the command or **message type field** and the other 4 bits are used as **control flags**.



MQTT Control Field Structure

The table below is taken from the MQTT 3.1.1 specification and shows a sample of MQTT commands, and their associated codes.

Because they are the most significant part of an **8 bit byte field** I have also shown their **byte values** in decimal as they would appear in the data packet.


Sample MQTT Control Message

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	Publish message

Decimal
16
32
48
=00010000 =16 Dec
=00100000 =32 Dec

Note: taken from the OASIS MQTT 3.1.1 specification document

Control Flags

Although there are **16 possible flags** very few are actually used.

The **publish message** makes the most use of these flags as shown in the table below:

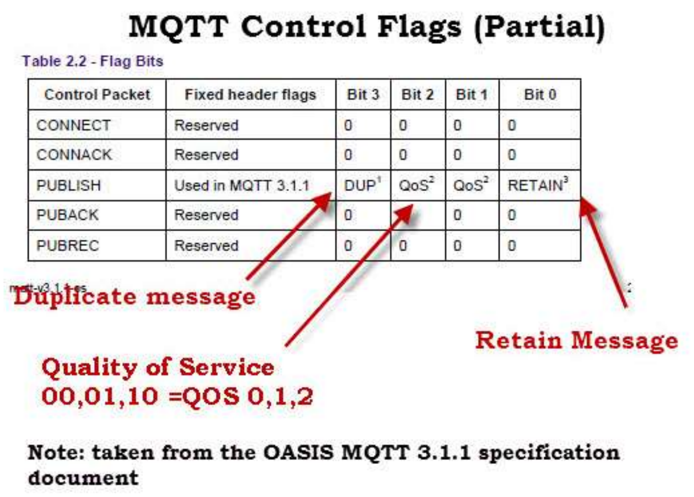

MQTT Control Flags (Partial)

Table 2.2 - Flag Bits

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

Duplicate message
Retain Message
Quality of Service
00,01,10 =QOS 0,1,2

Note: taken from the OASIS MQTT 3.1.1 specification document

The **duplicate flag** is used when re-publishing a message with QOS or 1 or 2

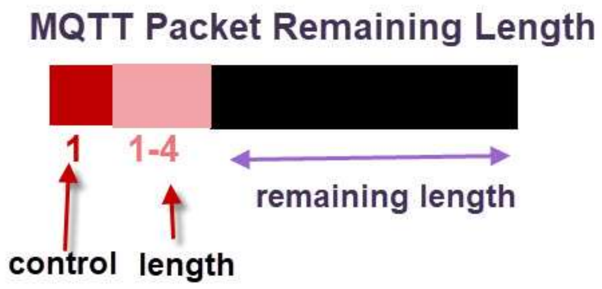
QOS Flags are used when publishing and indicate the QOS level -0,1,2

Retain Message flag is also used on publishing.

Remaining Length

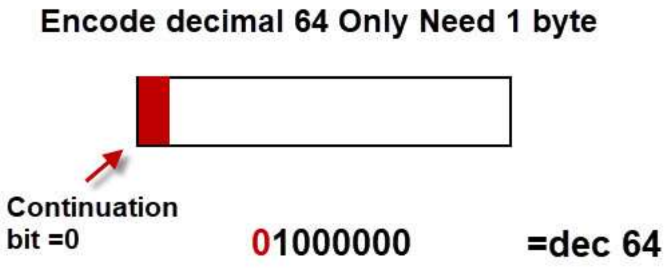
This is of variable length between 1 and 4 bytes. Each byte uses **7 bits** for the length with the **MSB** used as a continuation flag.

The remaining length is the number of bytes following the length field, includes variable length header and payload as illustrated below:



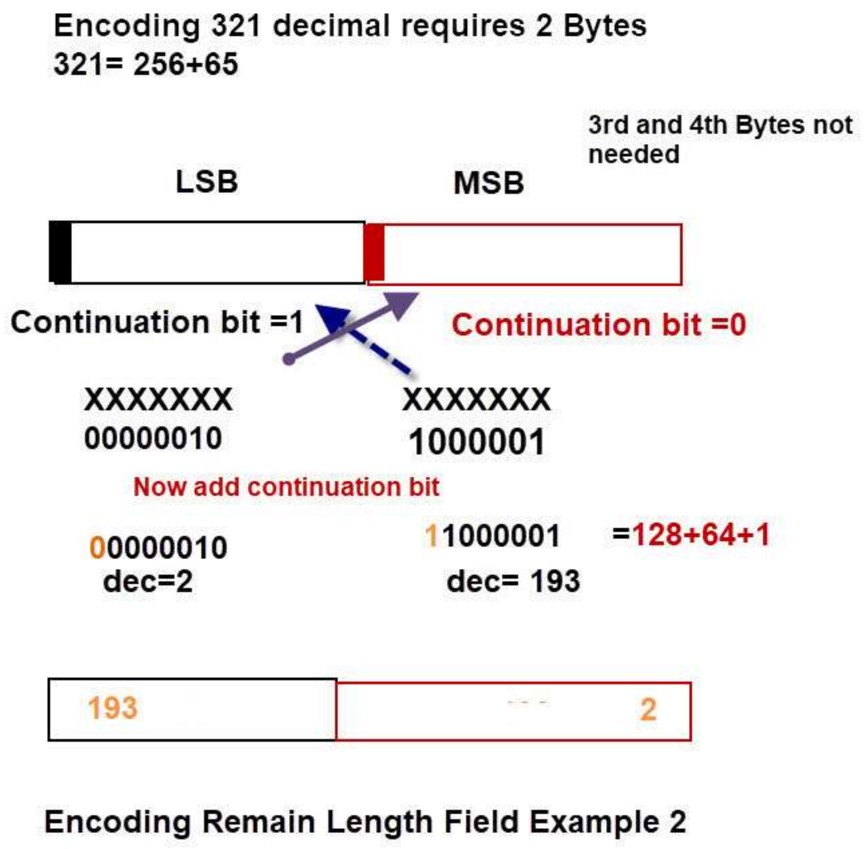
The following illustrates the length field for a packet size of 64 and 321 bytes

Remaining Packet length 64 bytes of requires only 1 byte:



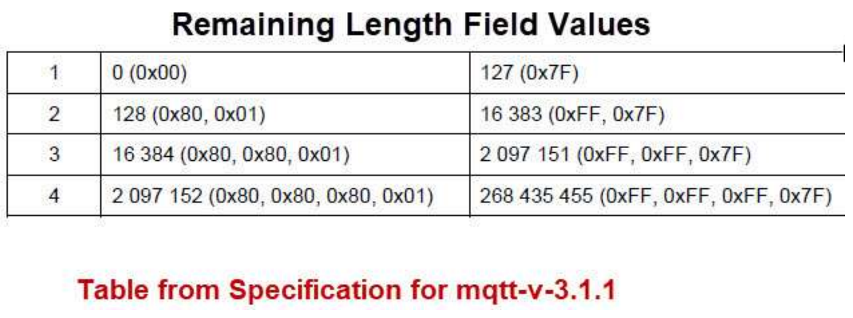
Encoding Remain Length Field Example 1

Packet length of 321 bytes requires a 2 byte remaining length field:



Note error in above should be 0xC1 0x01 .

The following table taken from the specification shows packet sizes and packet length field.



Remaining Length Field Values

1	0 (0x00)	127 (0x7F)
2	128 (0x80, 0x01)	16 383 (0xFF, 0x7F)
3	16 384 (0x80, 0x80, 0x01)	2 097 151 (0xFF, 0xFF, 0x7F)
4	2 097 152 (0x80, 0x80, 0x80, 0x01)	268 435 455 (0xFF, 0xFF, 0xFF, 0x7F)

Table from Specification for mqtt-v-3.1.1

Variable Length Header

As mentioned previously the **variable length header field** is not always present in an MQTT message.

Certain MQTT message types or commands require the use of this field to carry **additional control information**.

The variable length header field is similar, but not the same for all message types.

MQTT Connect and Disconnect Message Example

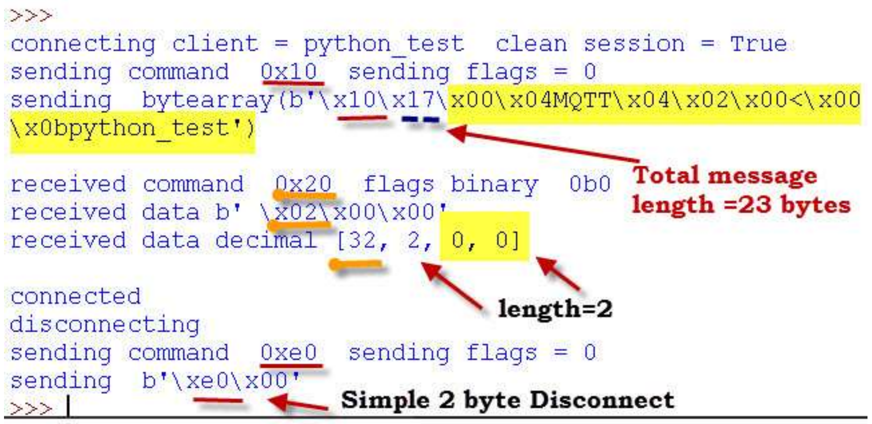
As an illustration we will now look at the packet details for a **connect message**.

Below is a real client connection and disconnect example showing the actual byte values for the sent and received data.

The CONNECT control code =**0x10**

The CONNACK control code =**0x20**

MQTT packet =control + length + protocol name + Protocol Level +Connect Flags + keep alive +**Payload**



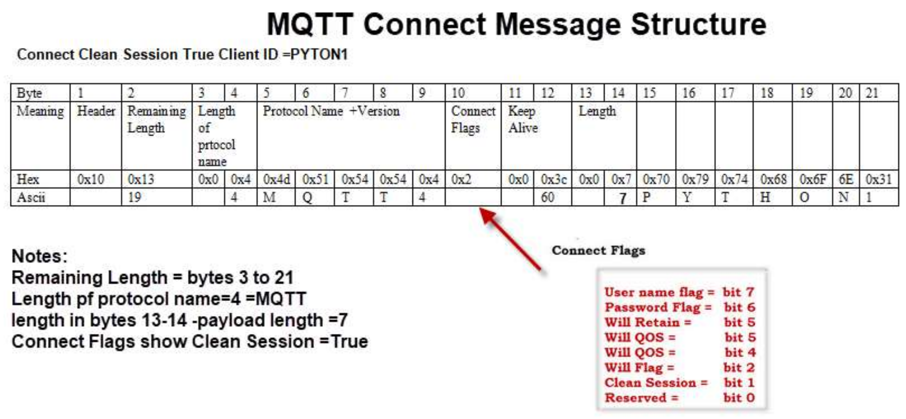
```
>>>
connecting client = python_test  clean session = True
sending command  0x10  sending flags = 0
sending  bytearray(b'\x10\x17\x00\x04MQTT\x04\x02\x00<\x00
\x0bpython_test')
received command  0x20  flags binary  0b0
received data b' \x02\x00\x00'
received data decimal [32, 2, 0, 0]

connected
disconnecting
sending command  0xe0  sending flags = 0
sending  b'\xe0\x00'
>>>
```

MQTT Message Example -1

Notes:

- Notice the connection (**0x10**) and connection acknowledge (**0x20**) control codes.
- Notice the total length of hex17 or 23 bytes not including the control and length fields. The length field is only 1 byte.
- You should also be able the see the client ID (python_test) in the sent packet.
- When looking at the actual packet bytes Python prints the hex values unless it can match a ASCII character.In the example above the keep alive field is **x00x3C** but is displayed as **x00<.** Where **Ascii < =0x3C**



MQTT Connect Message Structure

Connect Clean Session True Client ID =PYTON1

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Meaning	Header	Remaining Length	Length of prtocol name		Protocol Name +Version					Connect Flags	Keep Alive		Length								
Hex	0x10	0x13	0x0	0x4	0x4d	0x51	0x54	0x54	0x4	0x2	0x0	0x3c	0x0	0x7	0x70	0x79	0x74	0x68	0x6F	6E	0x31
Ascii		19		4	M	Q	T	T	4			60		7	P	Y	T	H	O	N	1

Notes:
Remaining Length = bytes 3 to 21
Length pf protocol name=4 =MQTT
length in bytes 13-14 -payload length =7
Connect Flags show Clean Session =True

Connect Flags

User name flag =	bit 7
Password Flag =	bit 6
Will Retain =	bit 5
Will QOS =	bit 5
Will QOS =	bit 4
Will Flag =	bit 2
Clean Session =	bit 1
Reserved =	bit 0

Notes:

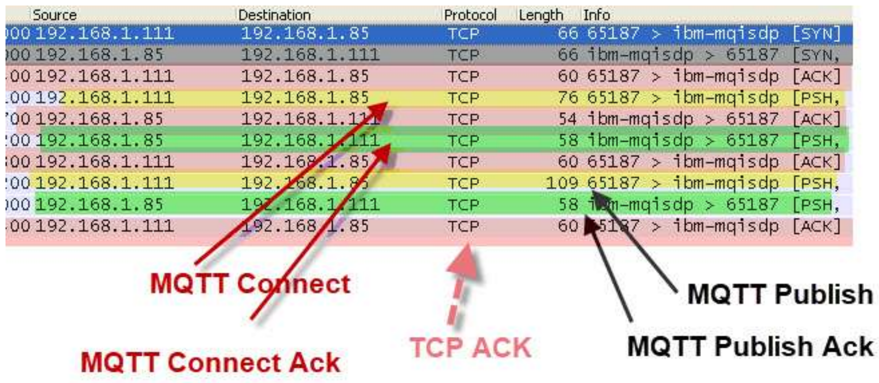
- The **client ID** field is sent as the first part of the **payload**, and not as part of the header.
- The **Client ID** is proceeded by a **length field**
- The Connect flags indicate that a **clean session** is being requested.
- Connection flags are part of the Variable Length header and used to indicate the presence or absence of username,password, and will message fields in the payload. It also contains the clean session flag and Will QOS.

Control Packet Table Summary

Control Packet	Variable Header	Payload
CONNECT	Required	Required
CONNACK	None	None
PUBLISH	Required	Optional
PUBACK	Required	None
PUBREC	Required	None
PUBREL	Required	None
PUBCOMP	Required	None
SUBSCRIBE	Required	Required
SUBACK	Required	Required
UNSUBSCRIBE	Required	Required
UNSUBACK	Required	Required
PINGREQ	None	None
PINGRESP	None	None
DISCONNECT	None	None

Wireshark Network Analysis

In response to a reader question regarding TCP protocol I created this screen shot taken from wireshark.



Source	Destination	Protocol	Length	Info
192.168.1.111	192.168.1.85	TCP	66	65187 > ibm-mqisdp [SYN]
192.168.1.85	192.168.1.111	TCP	66	ibm-mqisdp > 65187 [SYN,
192.168.1.111	192.168.1.85	TCP	60	65187 > ibm-mqisdp [ACK]
192.168.1.111	192.168.1.85	TCP	76	65187 > ibm-mqisdp [PSH,
192.168.1.85	192.168.1.111	TCP	54	ibm-mqisdp > 65187 [ACK]
192.168.1.85	192.168.1.111	TCP	58	ibm-mqisdp > 65187 [PSH,
192.168.1.111	192.168.1.85	TCP	60	65187 > ibm-mqisdp [ACK]
192.168.1.111	192.168.1.85	TCP	109	65187 > ibm-mqisdp [PSH,
192.168.1.85	192.168.1.111	TCP	58	ibm-mqisdp > 65187 [PSH,
192.168.1.111	192.168.1.85	TCP	60	65187 > ibm-mqisdp [ACK]

Wireshark Packet Capture- MQTT

It shows an MQTT client connecting and publishing (QOS 1). You can clearly see the ACK packets which have a total packet length of 58 bytes.

We know that ACK packets are 2 bytes.

Therefore the TCP packet without MQTT is around **56 bytes.**

What is also interesting to note, and something I hadn't thought of until I did the packet capture, is that each MQTT command or response will get a **TCP ACK** and maybe also an **MQTT ACK**.

If you look at the screenshot you can the MQTT connection get a TCP ACK response and an **MQTT Connect ACK** response.

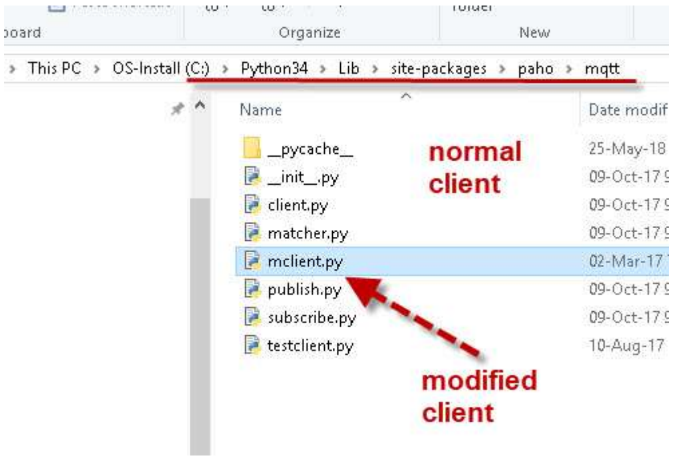
The **MQTT Connect ACK** response gets a **TCP ACK** response.

References:

MQTT V3.1.1 Specification pdf

Modified Client

To produce the screen shots showing the bytes I modified the original client and placed it in the same folder as the normal client.



To use it I changed the import from

```
import paho.mqtt.client as mqtt
```

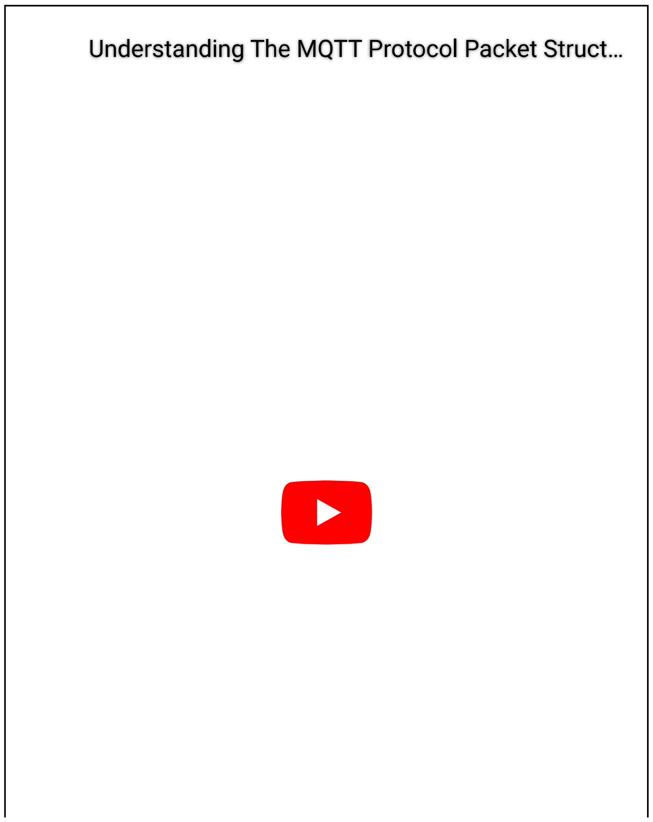
to

```
import paho.mqtt.mclient as mqtt
```

Here is the download

Modified Python MQTT Client
1 file(s) 20.62 KB
Download

Video -MQTT Protocol Packet Structure



Resources and related Tutorials

- MQTT for Beginners
- Using The Paho Python MQTT client

Please rate? And use Comments to let me know more

42 comments

Aristide Vanwanzeele says:

April 10, 2023 at 6:31 pm

Example with 321, should that not be 0xC1 0x01 (LSB, MSB) instead of 0xC1 0x02?
0xC1 comes from 0x80 | 0x41 (continuation bit + 65 hex), LSB.
0x01 comes from 0x00 | 0x01 (no continuation bit + 256), MSB.
256 + 65 = 321

Reply

steve says:

April 11, 2023 at 2:45 pm

I believe you are correct.0xC1 0x01
rgds
steve

Reply

Martin says:

January 29, 2022 at 12:03 pm

Can we check packet loss for mqtt on wireshark ?

Reply

steve says:

January 29, 2022 at 6:11 pm

Don't know as I am not a big wireshark user. I have a couple of python scripts that will do it.
Rgds
Steve

Reply

Martin says:

January 30, 2022 at 7:22 pm

Hi, can you shear the python codes, that can check the mqtt packet loss ?

Reply

steve says:

January 30, 2022 at 8:39 pm

They are here
http://www.steves-internet-guide.com/mqtt-broker-testing/

they are the stress test tools
rgds
steve

Reply

Martin says:

October 8, 2021 at 9:52 am

Hi, is there some test performance between MQTT vs MQTT-SN ?

Reply

steve says:

October 8, 2021 at 7:00 pm

Hi

I haven't seen an comparison data.

Rgds

Steve

Reply

Deepak kumar swami says:

July 9, 2021 at 2:50 am

Hey steve,

I'm trying to establish MQTT connection over two arduinos (mkr series).

The connection is successful but i'm not able to store the published message into a variable that i can use to turn on a buzzer.

```
while (mqttClient.available()) {
Serial.println((char)mqttClient.read());
qualButton = mqttClient.read();
if(){
Serial.println(qualButton);
tone(buzzerPin, 1000);
}
else {
Serial.println(qualButton);
noTone(buzzerPin);
}
}
```

Reply

---

steve says:

July 9, 2021 at 1:52 pm

Can you use the ask-steve page and send me the entire script.

rgds

steve

Reply

Nick says:

January 28, 2021 at 10:39 am

On the second figure, it says "E.g.. a connect message doesn't have a payload", then in the next section, it says an example that has a "fixed header + variable length header + payload" is a connect message. I think figure 2 is meant to say e.g. a CONNACK is an example that doesn't have a payload? unless I'm missing something

Reply

steve says:
January 28, 2021 at 4:35 pm

Hi
It was meant to say connect ack doesn't have a payload. Tks fo pointing it out I will amend the figure
rgds
steve

Reply

Efrain says:
January 11, 2021 at 5:44 am

Hi Steve. I've been following your tutorials, and have been trying to use Wireshark to determine the size of the data sent (including headers and such), but the packets don't show to be from the MQTT protocol nor from TCP port 1883. I am using the HiveMQ broker if that helps

Reply

steve says:
January 11, 2021 at 1:50 pm

Are you running wireshark on the same machine ans the client?

Reply

Efrain says:
January 11, 2021 at 6:52 pm

Yes, I am using one laptop to do everything: As subscriber, publisher, running the broker and capturing with wireshark

Reply

steve says:
January 12, 2021 at 2:44 pm

Then you should see the packets, You need to check your filter

Reply

Chop says:
November 19, 2020 at 3:18 pm

The controller that I am using to publish messages to the broker has a limitation to the number of bytes I can send at one time. If my data exceeds that limitation(but is within the mqtt broker limitations) how do I set my message up to let the broker know the data does not end at this particular message ie packet 1 of # ?

Reply

steve says:

November 19, 2020 at 3:21 pm

Hi
You need to split your data into smaller blocks. There is a python example in sending files over MQTT that should help.
http://www.steves-internet-guide.com/send-file-mqtt/
Rgds
Steve

Reply

Chop says:

December 7, 2020 at 6:15 pm

I can send it in smaller blocks but it removes some of the meaning of the data. The data I am sending is better if it stays together. So there isn't a way to send the data in smaller chunks but to put it back together on the broker side?

Reply

steve says:

December 7, 2020 at 6:51 pm

How large is the data? Is it your broker that you are sending to?

Reply

Chop says:

December 7, 2020 at 7:50 pm

My limitations are around 4k bytes in one message but my data can go up to 10k (maybe more). The end-user wants to see the data altogether and not in chunks. This is the data that I am sending to the broker.

steve says:

December 7, 2020 at 7:53 pm

Then you need to split it an put it back together at the other end. This is a common operation. I did a tutorial using python that does this and includes a

script.
http://www.steves-internet-guide.com/send-file-mqtt/
rgds
steve

Ehsan says:
February 18, 2020 at 11:15 am

Hi, I am Ehsan who is a beginner in MQTT.
I am trying to set reserved message types for doing specific tasks in MQTT but I cannot find the message types in source code of MQTT.

I would appreciate it if you guide me.

Thanks.

Reply

steve says:
February 18, 2020 at 6:10 pm

They are in the client.py file under lib/site-packages/paho/ (windows)
Here is what they look like
# Message types
CONNECT = 0x10
CONNACK = 0x20
PUBLISH = 0x30
PUBACK = 0x40
PUBREC = 0x50
PUBREL = 0x60
PUBCOMP = 0x70
SUBSCRIBE = 0x80
SUBACK = 0x90
UNSUBSCRIBE = 0xA0
UNSUBACK = 0xB0
PINGREQ = 0xC0
PINGRESP = 0xD0
DISCONNECT = 0xE0

Reply

KC says:
December 16, 2019 at 2:02 am

Hi Steve,
Just to clarify for the remaining length, each byte uses 7 bits of length and MSB as the continuation bit.

I noticed in Encoding Remain Length Field Example 1 has 9 bits (1 continuation bit, 8 length bits) instead of 8 bits (1 continuation bit, 7 length bits).
In Encoding Remain Length Field Example 2, the MSB has 9 bits too.

Can you please clarify and clear my doubts.
Thank you.
Reply

steve says:
December 16, 2019 at 2:12 pm

Hi
Tks for pointing that out there should only be 8 bits not 9 I will redo the images.
Rgds
Steve

Reply

Eddie says:
May 24, 2019 at 12:40 pm

Hello, can you send the codes of example 1. How do I capture the message that the message reaches the recipient? Is there sample code?

Reply

steve says:
May 24, 2019 at 4:01 pm

Hi
The modified client script is on the web page. I assume you are looking for a script that displays the header information in detail.
I did think of creating one but not really sure of the demand.
Can you tell Why do you need such a script.
Rgds
steve

Reply

Earle says:
January 15, 2019 at 7:24 pm

Hi Steve,

We are using IBM MQ 8 and the MQXR service and have seen instances where messages reside in the MQTT transmit queue and are flagged as in-doubt out for a client. From reading what the docs say, it sounds like the server has sent the message to the client but the client has not acknowledged the receipt. Am I correct here? The client then seems to get hung up trying to read messages and requires us to 'clear' the stuck messages. The client seems to get a nullPointerException. Is there a good tutorial I can point them to so they can connect and clear or acknowledge the previously delivered messages?

Reply

steve says:

January 16, 2019 at 9:01 pm

Sorry but I'm not familiar with the client or server. Is the the transmit queue on the client?I would check that the messages do get delivered.Does it happen on all QOS settings.
I would try a google search on
BM MQ 8 nullPointerException
if you haven't done so already
rgds
steve

Reply

Earle Ake says:

July 23, 2019 at 11:52 am

I was able to capture a tcpdump. Looks like what is happening is this:

MQTT client connects (already has a durable subscription in place)
MQTT client sends 'Connect Command'
MQTT Broker sends a message that is waiting in the MQTT transmit queue and marks message as 'in doubt out'
MQTT client sends the subscription request
MQTT Broker ACKs it
MQTT Broker sends a message to MQTT client
MQTT client ACKs it

So it looks like the MQTT client is just dropping messages until it has sent the subscribe packet. If the Broker has messages waiting, is it correct to send them once the client connects or should it be waiting until the subscribe comes in? Granted this is a durable subscription so messages are already waiting on it. My guess is the client must be ready to accept messages once the 'Connect Command' has been sent.

Reply

steve says:

July 25, 2019 at 1:21 pm

I would agree as there is actually no need for the client to subscribe.
Rgds
Steve

Reply

Cathy says:
January 12, 2019 at 6:40 pm

Hi Steve,

Can you share the code for the output shown in "MQTT Message Example – 1" fig? I am trying to write python code to work on the packet data but I do not know how to do it. Any examples or guidance you can provide on that would be helpful.

thank you.

Reply

steve says:
January 12, 2019 at 7:02 pm

Answered by email and tutorial amended to include download

Reply

Chris says:
December 3, 2018 at 7:26 am

Hi Steve,

Am using MQTT to develop a chat system, I need to send some additional information during publish and I dont want to include it in Payload(for some reason). Is it right way to go with Variable header field for the same??

Reply

steve says:
December 3, 2018 at 8:51 am

Not in version 3 as you can only use the payload. However v5 supports extensible emtadata which you could use see
http://modelbasedtesting.co.uk/2018/04/09/a-story-of-mqtt-5-0/
I think if you want that functionality you will need to develop for mqtt v5.
I would imagine most brokers will support v5 by early next year.
rgds
steve

Reply

Girija says:
November 13, 2018 at 5:13 pm

I enjoyed reading through, it is very crisp and detailed article.

Could you also please add a section how the data gets transitioned through MQTT over Google Protocol Buffer and significance of various QoS ?

Reply

steve says:

November 14, 2018 at 9:17 am

Hi
What is the Google protocol buffer?
rgds
steve

Reply

mbahrejo says:

October 3, 2018 at 10:57 am

Thank you for your explanation, .. I can now connect and publish message to the brooker. But i still fail to publish with retain message. My first byte is 0x31 which is 0x30 for publish and 0001 for retain flag. The message succesfully sent to brooker but not retained.

Reply

Shun says:

September 24, 2018 at 3:41 pm

Wow, Thank you. Nice work!!
Helped me a lot.

Reply

Sumit kulhadia says:

July 6, 2018 at 6:52 am

Hello,
I am using "mosquitto" implementation for MQTT installed on OpenWrt and Ubuntu.
I am sending the content of a file using mosquitto_pub and receiving it via mosquitto_sub. I did packet analysis of this transmission and found that,
1. If the file size 1434 Bytes then the content of the file gets split into TCP packet and MQTT packet.

Though the received content is intact, I am curious to know that why is this happening.

Reply

steve says:

July 6, 2018 at 2:00 pm

That is the way TCP/IP works. An MQTT packet needs to be inserted into a TCP/IP packet.
THe maximum size of the IP Packet is limited by the transmission media. See wiki MTU
https://en.wikipedia.org/wiki/Maximum_transmission_unit
This means that a large MQTT packet will need to be split over several TCP/IP packets.
AT the other end of the link the packet is reassembled by the MQTT client and appears as a single MQTT packet. Take a look at this tutorial it may help
https://stevessmarthomeguide.com/basic-networking-course/
section on frames

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *	
Email *	
Website	

Post Comment