MQTT Version 3.1.1

Working Draft 13

October 5, 2013

Technical Committee:

OASIS Message Queuing Telemetry Transport (MQTT) TC

Chairs:

Raphael J Cohn (raphael.cohn@stormmq.com), Individual
Richard J Coppen (coppen@uk.ibm.com), IBM

Editors:

Andrew Banks (Andrew_Banks@uk.ibm.com), IBM
Rahul Gupta (rahul.gupta@us.ibm.com), IBM

Additional artifacts:

- None

Related work:

Declared XML namespaces:

- None

Abstract:

MQ Telemetry Transport (MQTT) is a lightweight broker-based publish/subscribe messaging protocol designed to be open, simple, lightweight and easy to implement. These characteristics make it ideal for use in constrained environments, for example, but not limited to:

- Where the network is expensive, has low bandwidth or is unreliable
- When run on an embedded device with limited processor or memory resources

Features of the protocol include:

- The publish/subscribe message pattern to provide one-to-many message distribution and decoupling of applications
- A messaging transport that is agnostic to the content of the payload
- The use of TCP/IP to provide basic network connectivity
- Three qualities of service for message delivery:
  - "At most once", where messages are delivered according to the best efforts of the operating environment. Message loss or duplication can occur. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after.
  - "At least once", where messages are assured to arrive but duplicates may occur.
  - "Exactly once", where message are assured to arrive exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied.
- A small transport overhead and protocol exchanges minimized to reduce network traffic
- A mechanism to notify interested parties to an abnormal disconnection of a client using the Last Will and Testament feature

Status:

This Working Draft (WD) has been produced by one or more TC Members; it has not yet been voted on by the TC or approved as a Committee Draft (Committee Specification Draft or a Committee Note Draft). The OASIS document Approval Process begins officially with a TC vote to approve a WD as a Committee Draft. A TC may approve a Working Draft, revise it, and re-approve it any number of times as a Committee Draft.

Deleted: 2

Deleted: September 25, 2013

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

Initial URI pattern:

http://docs.oasis-open.org/mqtt/mqtt/v4.0/csd01/mqtt-v4.0-csd01.doc

(Managed by OASIS TC Administration; please don't modify.)

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

Table of contents

Contents

Deleted: 24
Deleted: 25
Deleted: 25
Deleted: 26
Deleted: 27
Deleted: 27
Deleted: 27
Deleted: 28
Deleted: 28
Deleted: 28
Deleted: 28
Deleted: 28
Deleted: 2
Deleted: 2
Deleted: 25 September 2013

Deleted: 29
Deleted: 29
Deleted: 29
Deleted: 30
Deleted: 30
Deleted: 30
Deleted: 30
Deleted: 30
Deleted: 31
Deleted: 31
Deleted: 31
Deleted: 32
Deleted: 33
Deleted: 34
Deleted: 34
Deleted: 35
Deleted: 36
Deleted: 37
Deleted: 37
Deleted: 37
Deleted: 38
Deleted: 38
Deleted: 38
Deleted: 38
Deleted: 39
Deleted: 39
Deleted: 39
Deleted: 39
Deleted: 39
Deleted: 41
Deleted: 41
Deleted: 2
Deleted: 2
Deleted: 25 September 2013

Deleted: 41
Deleted: 41
Deleted: 42
Deleted: 42
Deleted: 43
Deleted: 44
Deleted: 45
Deleted: 45
Deleted: 46
Deleted: 47
Deleted: 48
Deleted: 50
Deleted: 50
Deleted: 50
Deleted: 50
Deleted: 51
Deleted: 51
Deleted: 51
Deleted: 52

Deleted: 2
Deleted: 2
Deleted: 25 September 2013

1 Introduction

This specification is split into six main sections:

- Introduction and concepts
- Control Packet format
- The specific details of each Control Packet type
- Operational behaviour of the Client and Server
- Security considerations
- Conformance requirements for this version of the specification

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 [RFC2119].

Network Connection:

- Connects the Client to the Server,
- Provides the means to send an ordered, lossless, stream of bytes in both directions.

For examples see section 4.2.

Client:

A program that establishes the Network Connection:

The client can:

- Publish Application Messages that Clients may be interested in.
- Subscribe to request Application Messages that it is interested in receiving.
- Unsubscribe to remove a request for Application Messages.
- Disconnect from the server.

Server:

Accepts connections from Clients. It is the intermediary between the Client publishing Application Messages and the Clients which have made Subscriptions.

Application Message:

The data carried by the MQTT protocol across the network for the application.

Application Messages are transported by MQTT they have an associated Quality of Service and a Topic Name.

Topic Name:

The label attached to an Application Message which is matched against the subscriptions known to the Server. The Application Message is sent to each client with a matching Subscription.

Deleted: TCP

Deleted: <#>Defined in **[RFC793]** .¶

Deleted: ¶
>>>>>>>>>>Needs to be Generalised to SSSL/Websockets etc.

Deleted: TCP

Deleted: application

Deleted: Payload Message

Deleted: Payload Message

Deleted: Payload Message

Deleted: Payload Message

Deleted: Payload Message

Deleted: ¶
>>>>> should be Application Message

Deleted: Payload Message

Deleted: Payload Message

Deleted: Payload Message

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

Subscription:
The request made by a client to receive Application Messages published by Clients. The Subscription contains a Topic Name which can include wild card characters so that it may match many Topic Names.

Session:
A stateful interaction between a Client and a Server**.** Some sessions only last as long as the Network Connection, others span multiple NETWORK Connections.

MQTT Control Packet:
The packet of information that MQTT flows across the network, this might contain the "Application Message".

1.2 Conventions

Bits in a byte are labeled 7 through 0. Bit number 7 is the most significant bit, the least significant bit is assigned bit number 0.

All 16-bit word presented in this specification is in big-endian order: higher order bytes precede lower order bytes over the wire. A 16-bit word is presented on the wire as Most Significant Byte (MSB), followed by Least Significant Byte (LSB). This is based on IETF RFC 1700.

1.3 Normative references

[RFC793]
Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
http://www.ietf.org/rfc/rfc793.txt

[RFC2119]
S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997.
http://www.ietf.org/rfc/rfc2119.txt

[RFC 3629]
F. Yergeau UTF-8, a transformation format of ISO 10646 IETF RFC 3629, November 2003.
http://www.ietf.org/rfc/rfc3629.txt

[Unicode 6.2] *Unicode 6.2 specification*
http://www.unicode.org/versions/Unicode6.2.0

[RFC 1700]
Joyce K. Reynolds, Assigned Numbers, Internet standard STD 2, IETF RFC 1700, October 1994
http://tools.ietf.org/html/rfc1700

[RFC 6455]
T. Dierks, The Transport Layer Security (TLS) Protocol *Version 1.2, August 2008*
http://tools.ietf.org/html/rfc6455

[RFC 5246]
I Fette, Proposed Standard STD 2, The WebSocket Protocol, IETF RFC 6455, December 2011
http://tools.ietf.org/html/rfc6455

Deleted: Payload Message
Deleted: TCP
Deleted: TCP
Deleted: Payload Message
Formatted: Font: (Default) Arial Unicode MS, Font color: Auto, English (U.S.)
Formatted: HTML Preformatted, Tabs: Not at 1.62 cm + 3.23 cm + 4.85 cm + 6.46 cm + 8.08 cm + 9.69 cm + 11.31 cm + 12.92 cm + 14.54 cm + 16.16 cm + 17.77 cm + 19.39 cm + 21 cm + 22.62 cm + 24.23 cm + 25.85 cm
Deleted: I Fette
Deleted:
Formatted: Font: (Default) Arial Unicode MS, Font color: Auto, English (U.S.)
Deleted: *Proposed Standard STD 2, The WebSocket Protocol, IETF RFC 6455, December 2011*¶
Formatted: Font: (Default) Courier New, Font color: Black
Deleted: 2
Deleted: 2
Deleted: 25 September 2013

1.4 Non normative references

[MQTTV31]

MQTT V3.1 Protocol Specification.

http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.htmlhttp://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html.

1.5 Acknowledgements

Secretary:

Geoff Brown (geoff.brown@m2mi.com), M2MI

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

2 MQTT Control Packet format

The MQTT protocol works by exchanging a series of MQTT Control Packets in a defined way. This section describes the format of these packets. An MQTT Control Packet consists of up to three parts, always in the following order:

Fixed header, present in all MQTT Control Packets
Variable header, present in some MQTT Control Packets
Payload, present in some MQTT Control Packets

Unless stated otherwise, if either the server or client receives a Control Packet which does not meet this specification, it MUST disconnect the Network Connection. If the client or server encounters a transient error while processing an inbound Control Packet it MUST disconnect Network Connection which was used to send the packet. If a server detects a transient error it SHOULD NOT affect any other client.

Deleted: TCP

Deleted:

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

2.1 Data representations

2.1.1 Integer data values

Integer data values are in big-endian order: higher order bytes precede lower order bytes. A 16-bit word is presented on the wire as Most Significant Byte (MSB), followed by Least Significant Byte (LSB).

2.1.2 UTF-8 encoded strings

Many of the fields in the Control Packets are encoded as UTF-8. UTF-8 [RFC3629] is an efficient encoding of Unicode character that optimizes the encoding of ASCII characters in support of text-based communications.

In MQTT many of the Control Packets contain components that are defined as UTF-8 encoded strings. Each of these strings is prefixed with a two byte length field that gives the number of bytes in the UTF-8 encoded string itself, as shown in table below. Consequently there is a limit on the size of a string that can be passed in one of these UTF-8 encoded string components; you cannot use a string that would encode to more than 65535 bytes.

Unless stated otherwise all UTF-8 encoded strings are 0 to 65535 bytes in length.

Bit	7	6	5	4	3	2	1	0
Byte 1	String byte length MSB							
Byte 2	String byte length LSB							
Byte 3 ….	UTF-8 Encoded Character Data, if length > 0.							

Non normative example.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

316 For example, the string A𠮷 which is LATIN CAPITAL Letter A followed by the code point
317 U+2A6D4 (which represents a CJK IDEOGRAPH EXTENSION B character).
318

Bit	7	6	5	4	3	2	1	0
byte 1	Message Length MSB (0x00)							
	0	0	0	0	0	0	0	0
byte 2	Message Length LSB (0x05)							
	0	0	0	0	0	1	0	1
byte 3	'A' (0x41)							
	0	1	0	0	0	0	0	1
byte 4	(0xF0)							
	1	1	1	1	0	0	0	0
byte 5	(0xAA)							
	1	0	1	0	1	0	1	0
byte 6	(0x9B)							
	1	0	0	1	1	0	1	1
byte 7	(0x94)							
	1	0	0	1	0	1	0	0

319
320

321 2.2 Fixed header

322 Each MQTT Control Packet contains a fixed header. The table below shows the fixed header format.
323

Bit	7	6	5	4	3	2	1	0
Byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
Byte 2…	Remaining Length							

324

325 2.2.1 MQTT Control Packet types

326

327 **Position:** byte 1, bits 7-4.

328 Represented as a 4-bit unsigned value, the values are shown in the table below.

329

Name	Value	Direction of	Description

Deleted: 2
Deleted: 2
Deleted: 25 September 2013

		flow	
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	Client 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

2.2.2 Flags

The remaining bits[3-0] of byte 1 in the fixed header contain flags specific to each MQTT Control Packet type as detailed in the table below. Where a bit is unused, it must be set to zero and is reserved for future use. If invalid flags are received, the receiver MUST disconnect the Network connection.

Deleted: TCP

Deleted:

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

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

PUBLISH	Used in MQTT	Dup	QoS	QoS	RETAIN
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Used in MQTT	Dup	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Used in MQTT	Dup	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Used in MQTT	Dup	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

Dup = Duplicate delivery of Control Packet

QoS = Quality of Service

RETAIN = Retain flag

2.2.2.1 Dup

Position: byte 1, bit 3.

This flag MUST be set to 1 by the client or server when it attempts to re-deliver a PUBLISH Packet. The Dup flag MUST be 0 for all QoS 0 messages and for PUBREL, SUBSCRIBE and UNSUBSCRIBE Packets. If Dup 0 then the flow is the first occasion that the client or server has attempted to send the MQTT PUBISH Packet. If Dup 1 then the this indicates that the flow might be re-delivery of an earlier packet. The DUP flag MUST not be propagated when the PUBLISH Packet is sent to subscribers by the server.

Non Normative comment.

The recipient of a Control Packet that contains the Dup flag set to 1 cannot assume that it has seen an earlier copy of this packet. It is important to note that the Dup flag refers to the Control Packet itself and not to to the Application Message that it contains. When using QoS 1, it is possible for a client to receive a PUBLISH Packet with DUP set to 0 that contains a repetition of an Application Message that it received earlier, but with a different MessageID.

2.2.2.2 QoS

Position: byte 1, bit 2-1.

This field indicates the level of assurance for delivery of an Application Message. The QoS levels are shown in the table below.

QoS value	bit 2	bit 1	Description
0	0	0	At most once delivery

Deleted: ¶

Formatted: Left

Deleted: ¶
See MQTT Issue-27.

Formatted: Heading 4,H4, Indent: Left: 2.16 cm

Formatted: Justified, Indent: Left: 0.63 cm

Deleted: PUBREL, SUBSCRIBE or UNSUBSCRIBE Control

Deleted: if

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Deleted: Control

Deleted:

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear

Deleted: If Dup 1 then the flow is a re-delivery of an earlier packet.

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: Not Bold

Deleted: If 1, the recipient might have previously received the Control Packet it should not treat this flag as an indication that it has previously received the MQTT Control Packet and should not use the Dup flag alone to guarantee to its application that a duplica ... [1]

Formatted: Bullets and Numbering ... [2]

Deleted: Payload Message

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

1	0	1	At least once delivery
2	1	0	Exactly once delivery
3	1	1	Reserved

2.2.2.3 RETAIN

Position: byte 1, bit 0.

See issue MQTT-10.

This flag is only used on the PUBLISH Packet.

If 1, in a PUBLISH Packet sent by a client to a server, the server MUST store the publication, so that it can be delivered to future subscribers whose subscriptions match the topic. When a new subscription is established, the last retained message, if any, on each matching topic MUST be sent to the subscriber.

If 0, in a PUBLISH Packet sent by a client to a server, the server does not store the message nor does it remove or replace any existing retained publication.

When sending a PUBLISH Packet to a client the server MUST set the RETAIN bit to 1 if a message is sent as a result of a new subscription being made by a client. It MUST set the RETAIN bit to 0 when a PUBLISH Packet is sent to a client because it matches an established subscription regardless of how the bit was set in the message it received.

A PUBLISH Packet with a retain flag set to 1 and a payload containing zero bytes will be processed as normal by the server and sent to clients with a subscription matching the topic name. Additionally any existing retained publication with the same topic name MUST be removed and any future subscribers for the topic will not receive a retained publication. As normal means that the retained bit is not set in the message received by existing clients.

Non normative comment.

Retained messages are useful where publishers send state messages on an irregular basis. A new subscriber will receive the most recent state.

2.2.3 Remaining Length

Position: byte 2.

The Remaining length is the number of bytes remaining within the current packet, including data in the variable header and the payload. The Remaining Length does not include the bytes used to encode the Remaining Length.

The Remaining Length is encoded using a variable length encoding scheme which uses a single byte for values up to 127. Larger values are handled as follows. Seven bits of each byte encode the data, and the eighth bit indicates that there are following bytes in the representation. Each byte encodes 128 values and a "continuation bit". The maximum number of bytes in the Remaining Length is four.

Deleted: Control

Deleted: p

Deleted: p

Deleted: p

Deleted: A server MAY delete a retained message if it receives a message with a zero-length payload and the Retain flag set on the same topic.¶

Deleted: will

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

Non normative comment.

For example, the number 64 decimal is encoded as a single byte, decimal value 64, hex 0x40. The number 321 decimal (= 65 + 2*128) is encoded as two bytes, least significant first. The first byte 65+128 = 193. Note that the top bit is set to indicate at least one following byte. The second byte is 2.

Non normative comment.

This allows applications to send Control Packets of size up to 268,435,455 (256 MB). The representation of this number on the wire is: 0xFF, 0xFF, 0xFF, 0x7F. The table below shows the Remaining Length values represented by increasing numbers of bytes.

Deleted: p

Digits	From	To
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)

Non normative comment.

The algorithm for encoding a non negative integer (X) into the variable length encoding scheme is as follows:

```
do
       encodedByte = X MOD 128
       X = X DIV 128
      // if there are more data to encode, set the top bit of this byte
      if ( X > 0 )
          encodedByte = encodedByte OR 128
      endif
          'output' encodedByte
 while ( X > 0 )
```

Where `MOD` is the modulo operator (`%` in C), `DIV` is integer division (`/` in C), and `OR` is bit-wise or (`|` in C).

Non normative comment.

The algorithm for decoding the Remaining Length field is as follows:

```
multiplier = 1
value = 0
do
     encodedByte = 'next byte from stream'
     value += (encodedByte AND 127) * multiplier
     multiplier *= 128
     if (multiplier > 128*128*128)
```

Deleted: 2
Deleted: 2
Deleted: 25 September 2013

```
                    throw Error(Malformed Remaining Length)
            while ((encodedByte AND 128) != 0)
```

where `AND` is the bit-wise and operator (& in C).

When this algorithm terminates, `value` contains the Length.

2.3 Variable header

Some types of MQTT Control Packets also contain a variable header component. It resides between the fixed header and the payload. The Packet Identifier is common to several packet types

2.3.1 Packet Identifier

Bit	7	6	5	4	3	2	1	0
	Packet Identifier MSB							
	Packet Identifier LSB							

The variable header component of many of the Control Packet types includes a 2 byte Packet Identifier (PacketId) field. These Control Packets are PUBLISH (where QoS > 0), PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK.

A PUBACK, PUBREC, PUBREL Packet MUST contain the same PacketId as the PUBLISH Packet that initiated the flow. Similarly SUBACK and UNSUBACK MUST contain the PacketId that was used in the corresponding SUBSCRIBE and UNSUBSCRIBE Packet respectively.

A PUBLISH Packet MUST NOT contain a PacketId if its QoS value is set to 0.

SUBSCRIBE, UNSUBSCRIBE, and PUBLISH (in cases where QoS > 0) MUST contain a non-zero 16-bit PacketId. Each time a client sends a new packet of one of these types it MUST assign it a currently unused PacketId. If a client resends a particular Control Packet, then it MUST use the same PacketId in subsequent resends of that packet. The PacketId becomes available for reuse after the client has processed the corresponding acknowledgement packet (PUBREC in the case of QoS 2). The same conditions apply to a server when it sends a PUBLISH with QoS > 0.

Control Packets that contain a Packet Identifier

Control Packet	Packet Identifier field
CONNECT	NO
CONNACK	NO
PUBLISH	YES (If Qos > 0)
PUBACK	YES
PUBREC	YES
PUBREL	YES

Deleted: i
Deleted: Control
Deleted: Control
Deleted: Control
Deleted: Control
Deleted: 2
Deleted: 2
Deleted: 25 September 2013

PUBCOMP	YES
SUBSCRIBE	YES
SUBACK	YES
UNSUBSCRIBE	YES
UNSUBACK	YES
PINGREQ	NO
PINGRESP	NO
DISCONNECT	NO

The client and server assign PacketIds independently of each other. As a result, client server pairs can participate in concurrent message exchanges using the same PacketId.

Non-Normative comment

It is possible for a client to send a PUBLISH Packet with PacketId 0x1234 and then receive a different PUBLISH with PacketId 0x1234 from its server before it receives a PUBACK for the PUBLISH that it sent.

```
Client                    Server
   PUBLISH PacketId=0x1234--->
   <--PUBLISH PacketId=0x1234
   PUBACK PacketId=0x1234--->
   <--PUBACK PacketId=0x1234
```

2.4 Payload

Some MQTT Control Packets contain a payload as the final part of the packet, as described in section 3.

Deleted: Control
Deleted: s
Deleted: ublish
Deleted: ublish
Deleted: ,
Deleted: 3
Deleted: 2
Deleted: 2
Deleted: 25 September 2013

3 MQTT Control Packets

3.1 CONNECT – Client requests a connection to a server

After a Network connection is established by a client to a server, the first flow from the client to the server MUST be a CONNECT Packet. A client can only flow the CONNECT Packet once over a Network Connection. The Server MUST process a second CONNECT Packet sent from a client as a protocol violation and disconnect the client.

The payload contains one or more encoded fields. They specify a unique client identifier for the client, a Will topic and message and the User Name and Password. All but the client identifier are optional and their presence is determined based on flags in the variable header.

Deleted: TCP/IP [RFC793]

Deleted: c

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

3.1.1 Fixed header

The fixed header format is shown in the table below.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (1)				Reserved			
	0	0	0	1	0	0	0	0
byte 2…	Remaining Length							

Remaining Length field

Remaining Length is the length of the variable header (10 bytes) plus the length of the Payload. As described in section 2.2.3

3.1.2 Variable header

The variable header for the CONNECT Packet consists of four fields in the following order: Protocol Name, Protocol Level, Connect Flags, and Keep Alive.

Deleted: Control

Deleted: Timer

3.1.2.1 Protocol Name

	Description	7	6	5	4	3	2	1	0
Protocol Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (4)	0	0	0	0	0	1	0	0
byte 3	‘M’	0	1	0	0	1	1	0	1
byte 4	‘Q’	0	1	0	1	0	0	0	1
byte 5	‘T’	0	1	0	1	0	1	0	0

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

byte 6	'T'	0	1	0	1	0	1	0	0

The Protocol Name, is a UTF-8 encoded string that represents the protocol name "MQTT", capitalized as shown. The string, its offset and length will not be changed by future versions of the MQTT specification.

The server MAY disconnect the client if the protocol name is incorrect.

Non normative comment

Control Packet inspectors, such as firewalls, can use the Protocol Name to identify MQTT traffic

3.1.2.2 Protocol Level

	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Protocol Level									
byte 7	Level(4)	0	0	0	0	0	1	0	0

The 8 bit unsigned value that represents the revision level of the protocol used by the client. The value of the Protocol Level field for the current version of the protocol is 4 (0x04). The server MUST respond to the CONNECT Packet with a CONNACK return code 0x01 (unacceptable protocol level) and then disconnect the client if the Protocol Level is not supported by the server.

Deleted: p

3.1.2.3 Connect Flags

The Connect Flags byte contains a number of parameters specifying the behavior of the MQTT connection. It also indicates the presence or absence of fields in the payload.

	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
byte 8	X	X	X	X	X	X	X	0

The server MUST validate that reserved bits are set to zero and disconnect the client if they are not zero.

3.1.2.3.1 Clean Session

Position: bit 1 of the Connect Flags byte.

If set to 0, the Server resumes communications with the Client based on state from the current Session, otherwise it creates a new Session. The Client and Server will store the Session after the Client and Server are disconnected. After disconnection, the Server MUST accumulate further QoS 1 and QoS 2 messages that match any subscriptions that the client had at the time of disconnection. It MAY accumulate QOS 0 messages that meet the same criteria.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

If set to 1, the Client and Server MUST discard any previous Session and start a new one. This Session lasts as long as the Network Connection and MUST NOT be reused.

The Session state in the client consists of:

- QoS 1 and QoS 2 messages for which transmission to the server is incomplete.
- The client MAY store QoS 0 messages for transmission after the CONNECT Packet has flowed.

The Session state in the server consists of:

- The client subscriptions which it has acknowledged.
- All QoS 1 and QoS 2 messages for which transmission to the client is incomplete or where transmission to the client has not yet been started.
- The server MAY store QoS 0 messages for which transmission is incomplete or where transmission to the client has not yet been started.

Retained publications do not form part of the Session state in the server, they MUST NOT be deleted when the Session ends.

State could be lost by either the client or server due to the storage mechanism used, or an administrator action. This could result in the loss or duplication of messages regardless of the QoS used.

When Clean Session is set to 1 the client and server need not process the deletion of state atomically.

Non Normative comment.

Consequently in the event of a failure to connect the client should repeat its attempts to connect with Clean Session set to 1, until it connects successfully.

Non Normative comment.

Typically, a client will always connect using CleanSession 0 or CleanSession 1 and not swap between the two values. The choice will depend on the application. A client using CleanSession 1 will not receive old publications and has to subscribe afresh to any topics that it is interested in each time it connects. A client using CleanSession 0 will receive all QoS 1 or QoS 2 messages that were published whilst it was disconnected. Hence, to ensure that you do not lose messages use QoS 1 or QoS 2 with CleanSession 0.

Non Normative comment.

When a client connects with cleanSession = 0 it is requesting that the server maintain its MQTT session state after it disconnects. Clients should only connect with cleanSession = 0 if they intend to reconnect to the server at some later point in time. When a client has determined that it has no further use for the session it should do a final reconnect with cleanSession = 1

3.1.2.3.2 Will Flag

Position: bit 2 of the Connect Flags.

The Will Flag indicates that a Will Message MUST be published by the server when the server detects that the client is disconnected for any reason other than the client flowing a DISCONNECT Packet. This includes but is not limited to the flowing situations:

- An I/O error or network failure detected by the server.

Deleted: TCP

Deleted: connect

Deleted: p

Deleted: o

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Deleted: Control

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

- The client fails to communicate within the Keep Alive time
- The client disconnects the Network Connection without first sending a DISCONNECT Packet.

If the Will Flag is set to 1, the Will QoS and Will Retain fields in the Connect Flags will be used by the server, and the Will Topic and Will Message fields MUST be present in the payload.

3.1.2.3.3 Will QoS

Position: bits 4 and 3 of the Connect Flags.

A connecting client specifies the QoS level of the Will message in the Will QoS field.

If the Will Flag is set to 0, then the Will QoS MUST be set to 0 (0x00).
If the will Flag is set to 1, the value of Will QoS can be 0 (0x00), 1 (0x01), or 2 (0x02).

3.1.2.3.4 Will Retain

Position: bit 5 of the Connect Flags.

If the Will Flag is set to 0, then the Will Retain Flag MUST be set to 0

If the Will Flag is set to 1:

- If Will Retain is set to 0, the server MUST publish the will message as a non-retained publication.
- If Will Retain is set to 1, the server MUST publish the will message as a retained publication.

3.1.2.3.5 User Name Flag

Position: bit 7 of the Connect Flags.

If the User Name Flag is set to 0, a user name MUST NOT be present in the payload.
If the User Name Flag is set to 1, a user name MUST be present in the payload.

3.1.2.3.6 Password Flag

Position: bit 6 of the Connect Flags byte.

If the Password Flag is set to 0, a password MUST NOT be present in the payload.
If the Password Flag is set to 1, a password MUST be present in the payload.
If the User Name Flag is set to 0 then the Password Flag MUST be set to 0.

3.1.2.4 Keep Alive

	7	6	5	4	3	2	1	0
byte 9	Keep Alive MSB							

Deleted: Timer schedule
Deleted: TCP
Deleted: c
Deleted: Control
Deleted: 2
Deleted: 2
Deleted: 25 September 2013

byte 10	Keep Alive LSB

The Keep Alive is a time interval measured in seconds. Expressed as a 16-bit word, it is the maximum time interval that is permitted to elapse between Control Packets sent by a client.

It is the responsibility of the client to ensure that the interval between Control Packets being sent does not exceed the Keep Alive value. In the absence of sending any other Control Packets, the client MUST send a PINGREQ Packet.

The client can send PINGREQ at any time and use the PINGRESP to determine that the network and the server are working.

If the server does not receive a Control Packet from the client within one and a half times the Keep Alive time period, it MUST disconnect the client as if the network had failed.

If a client does not receive a PINGRESP Packet within a reasonable amount of time after it has sent a PINGREQ, it SHOULD close the Network Connection.

A Keep Alive value of zero (0) has the effect of turning off the keep alive mechanism.

Non normative comment.

The actual value of the Keep Alive is application-specific, typically this is a few minutes. The maximum value is approximately 18 hours.

3.1.2.5 Variable header example, Non normative

	Description	7	6	5	4	3	2	1	0
Protocol Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (4)	0	0	0	0	0	1	0	0
byte 3	‘M’	0	1	0	0	1	1	0	1
byte 4	‘Q’	0	1	0	1	0	0	0	1
byte 5	‘T’	0	1	0	1	0	1	0	0
byte 6	‘T’	0	1	0	1	0	1	0	0
Protocol Level									
	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 7	Level (4)	0	0	0	0	0	1	0	0
Connect Flags									

Deleted: Control

Deleted: p

Deleted: TCP c

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

byte 8	User Name Flag (1) Password Flag (1) Will Retain (0) Will QoS (01) Will Flag (1) Clean Session (1) *Reserved* (0)	1	1	0	0	1	1	1	0
Keep Alive									
byte 9	Keep Alive MSB (0)	0	0	0	0	0	0	0	0
byte 10	Keep Alive LSB (10)	0	0	0	0	1	0	1	0

3.1.3 Payload

The payload of the CONNECT Packet contains one or more length-prefixed fields, whose presence is determined by the flags in the variable header. These fields, if present, MUST appear in the order below.

3.1.3.1 Client Identifier

The Client Identifier (ClientId) MUST be present and is the first field in the payload.

The ClientId MUST comprise only Unicode characters, and the length of the UTF-8 encoding must be at least zero bytes and no more than 65535 bytes.

The server MAY restrict the ClientId it allows in terms of their lengths and the characters they contain, however the server MUST allow ClientIds which are between 1 and 23 UTF-8 encoded bytes in length, and contain only the characters "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ".

A zero-byte clientId is considered a special case and a server MUST support the following behaviour or not permit zero-byte clientIds.

If the client supplies a zero-byte ClientId, the client MUST also set Clean Session to 1. This combination indicates that the client has no unique identifier and that the server will process the connection as if it were a unique client.

If the client supplies a zero-byte clientId with Clean Session set to 0, the server MUST respond to the CONNECT Packet with a CONNACK return code 0x02 (Identifier rejected) and then terminate the Network Connection.

If the server rejects the ClientId it MUST respond to the CONNECT Packet with a CONNACK return code 0x02 (Identifier rejected) and then terminate the Network Connection

The ClientId identifies the client to the server. The ClientId MUST be unique for each client connecting to the server, It can be used by clients and by servers to identify state that they hold relating to this MQTT connection between the client and the server.

Deleted: Control
Deleted: ¶
Formatted: Justified
Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Font: 10 pt, Font color: Auto
Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Font: 10 pt, Font color: Auto
Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Font: 10 pt, Font color: Auto
Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Font: 10 pt, Font color: Auto
Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Font: 10 pt, Font color: Auto
Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear
Deleted: ¶ The ClientId MUST comp ... [3]
Deleted: 2
Deleted: 2
Deleted: 25 September 2013

Non Normative comment.

A client implementation may provide a convenience method to generate a random ClientId. Use of such a method should be actively discouraged when the Clean Session flag is set to 0.

Deleted: ¶

3.1.3.2 Will Topic

If the Will Flag is set to 1, the Will Topic is the next field in the payload. The Will Topic is a UTF-8 encoded string.

3.1.3.3 Will Message

See Issue MQTT-2

If the Will Flag is set to 1 the Will Message is the next field in the payload. The Will Message defines the payload content of the message that is published to the Will Topic if the client is disconnected for any reason other than the client sending a DISCONNECT Packet. This field consists of a 2-byte length followed by the payload for the Will Message expressed as a sequence of zero or more bytes. The length gives the number of bytes in the payload that follows and does not include the 2 bytes taken up by the length itself.

Deleted: p

When the Will Message is published to the Will Topic its payload consists only of the payload portion of this field, not the first two length bytes.

3.1.3.4 User Name

If the User Name Flag is set to 1, this is the next field in the payload. User Name is a UTF-8 encoded string and can be used by the server for authentication, and authorization.

3.1.3.5 Password

If the Password Flag is set to 1, this is the next field in the payload. Password is a UTF-8 encoded string and can be used by the server for authentication of the client.

3.1.4 Response

Note that a server MAY support multiple protocols (including earlier versions of this protocol) on the same TCP port or other endpoint. If the server determines that the protocol is MQTT 3.1.1 then it MUST validate the connection attempt as follows.

1. If the server does not receive a CONNECT Packet within a reasonable amount of time after the Network Connection is established, the server SHOULD close the connection.

   Deleted: p

   Deleted: TCP c

2. The server MUST validate that the CONNECT Packet conforms to section 3.1 and close the Network Connection without sending a CONNACK if it does not conform.

   Deleted: TCP c

3. The server MAY check that the contents of the CONNECT Packet meet any further restrictions and MAY perform authentication and authorization checks. If any of these checks fail, it SHOULD send an appropriate CONNACK response with a non zero return code as described in section 3.2 and it MUST close the Network Connection.

   Deleted: TCP connection

If validation is successful the server MUST perform the following steps.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

1. If the ClientId represents a client already connected to the server then the server MUST disconnect the existing client.

2. Processing of Clean Session is performed as described in section 3.1.2.3.1.

3. The server acknowledges the CONNECT Packet with a CONNACK Packet containing a zero return code.

4. Start message delivery and keep alive monitoring.

Clients are allowed to send further Control Packets immediately after sending a CONNECT Packet; clients need not wait for a CONNACK Packet to arrive from the server. If the server rejects the CONNECT, it MUST NOT process any data sent by the client after the CONNECT Packet.

Non Normative comment.

Clients typically wait for a CONNACK Packet, However, if the client exploits its freedom to send Control Packets before it receives a CONNACK, it might simplify the client implementation as it does not have to police the connected state. The client accepts that any data that it sends before it receives a CONNACK packet from the server will not be processed if the server rejects the connection.

3.2 CONNACK – Acknowledge connection request

The CONNACK Packet is the packet sent by the server in response to a CONNECT Packet received from a client. The first packet sent from the server to the client MUST be a CONNACK Packet.

If the client does not receive a CONNACK Packet from the server within a reasonable amount of time, the client SHOULD close the Network Connection. A "reasonable" amount of time depends on the type of application and the communications infrastructure.

3.2.1 Fixed header

The fixed header format is shown in the table below.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet Type (2)				Reserved			
	0	0	1	0	0	0	0	0
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

Deleted: ¶
Formatted: Indent: Left: 0 cm
Deleted: p
Deleted: p
Deleted: p
Deleted: Control
Deleted: Control
Deleted: p
Deleted: TCP c
Deleted: 2
Deleted: 2
Deleted: 25 September 2013

Remaining Length field

This is the length of the variable header. For the CONNACK Packet this has the value 2.

3.2.2 Variable header

The variable header format is shown in the table below.-

	Description	7	6	5	4	3	2	1	0
Reserved for future use									
byte 1		0	0	0	0	0	0	0	0
CONNECT Return code									
byte 2									

If a correctly formed CONNECT Packet is received by the server, but the server is unable to process it (for some reason), then the server SHOULD attempt to flow one of the following CONNACK return codes before disconnecting the Network Connection. The values for the one byte unsigned CONNECT Return code field are shown in the table below.

Deleted: p

Deleted: TCP Connection

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 in 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
6-255		Reserved for future use

Deleted: TCP connection

If none of these return codes are deemed applicable, then the server MUST disconnect the Network Connection without flowing a CONNACK.

Deleted: TCP Connection

3.2.3 Payload

There is no payload in the CONNACK Packet.

3.3 PUBLISH – Publish message

A PUBLISH Control Packet is sent from a client to a server or from server to a client to transport a Application Message.

Deleted: Payload Message

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

3.3.1 Fixed header

The table below shows the fixed header format:

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (3)				Dup flag	QoS level		RETAIN
	0	0	1	1	X	X	X	X
byte 2	Remaining Length							

Dup flag

See Dup section 1.1.1.1 for details.

QoS level

See QoS section 2.2.2.2 for details.

RETAIN flag

See Retain section 2.2.2.3 for details.

Remaining Length field

This is the length of variable header plus the length of the payload.

3.3.2 Variable header

The variable header contains the following fields in the order below:

3.3.2.1 Topic name

The topic name is always present as the first field in the variable header. The topic name identifies the information channel to which payload data is published.

The Topic name MUST be a UTF-8 encoded string as defined in section 2.1.2.

The topic name in the PUBLISH Packet received by a subscribing client MUST NOT contain wild card characters. The topic name received will match the subscriber's topic filter. However, since the server is permitted to override the topic name, it might not be the same as the topic name in the original PUBLISH Packet.

3.3.2.2 Packet Identifier

The Packet Identifier (PacketId) field is only present in PUBLISH Packets where the QoS level is 1 or 2. See Packet Identifiers section 2.3.1 for more details.

3.3.2.3 Variable header example Non Normative

The table below illustrates an example of variable header for a PUBLISH Packet.

Deleted: 2.2.2.1

Deleted: p

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

Field	Value
Topic Name	a/b
PacketId	10

The format of the variable header in this case is shown in the table below.

	Description	7	6	5	4	3	2	1	0
Topic Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	'a' (0x61)	0	1	1	0	0	0	0	1
byte 4	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 5	'b' (0x62)	0	1	1	0	0	0	1	0
Packet Identifier									
byte 6	PacketId MSB (0)	0	0	0	0	0	0	0	0
byte 7	PacketId LSB (10)	0	0	0	0	1	0	1	0

3.3.3 Payload

The Payload contains the data for publishing. The content and format of the data is application specific. The length of the payload can be calculated by subtracting the length of the variable header from the Remaining Length field that is in the Fixed Header. It is valid for a PUBLISH Packet to contain a zero length payload.

Deleted: p

3.3.4 Response

The response to a PUBLISH Packet depends on the QoS level. The table below shows the expected responses.

QoS Level	Expected Response
QoS 0	None
QoS 1	PUBACK Packet
QoS 2	PUBREC Packet

The Packet Identifier in the PUBACK Packet or PUBREC Packet MUST be the same as that in the PUBLISH Packet.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

3.3.5 Actions

The client uses a PUBLISH Packet to send an Application Message to the server for distribution to clients with matching subscriptions.

The server uses a PUBLISH Packet to send an Application Message to clients with matching subscriptions.

The action of the recipient when it receives a PUBLISH Packet depends on the QoS level as described in section 4.3

See MQTT issue-52

Note that if a server implementation does not authorize a PUBLISH to be performed by a client; it has no way of informing that client. It must therefore make a positive acknowledgement, according to the normal QoS rules, and the client will not be informed that it was not authorized to publish the message.

3.4 PUBACK – Publish acknowledgement

A PUBACK Packet is the response to a PUBLISH Packet with QoS level 1.

3.4.1 Fixed header

The table below shows the format of the fixed header.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (4)				Reserved			
	0	1	0	0	0	0	0	0
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

Remaining Length field

This is the length of the variable header. For the PUBACK Packet this has the value 2.

3.4.2 Variable header

Contains the Packet Identifier (PacketId) from the PUBLISH Packet that is being acknowledged. The table below shows the format of the variable header.

Bit	7	6	5	4	3	2	1	0
byte 1	PacketId MSB							
byte 2	PacketId LSB							

Deleted: application message

Deleted: application message

Deleted: 4.2

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

3.4.3 Payload

There is no payload in the PUBACK Packet.

3.4.4 Actions

When the sender of a PUBLISH Packet receives a PUBACK Packet it discards the original message.

The action of the recipient is fully described in section 4.3

Deleted: 4.2

3.5 PUBREC – Publish received (QoS 2 publish received, part 1)

A PUBREC Packet is the response to a PUBLISH Packet with QoS 2. It is the second packet of the QoS 2 protocol flow.

Deleted: Control

3.5.1 Fixed header

The table below shows the format of the fixed header.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (5)				Reserved			
	0	1	0	1	0	0	0	0
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

Remaining Length field

This is the length of the variable header. For the PUBREC Packet this has the value 2.

3.5.2 Variable header

The variable header contains the Packet Identifier for the acknowledged PUBLISH Packet. The table below shows the format of the variable header.

bit	7	6	5	4	3	2	1	0
byte 1	PacketId MSB							
byte 2	PacketId LSB							

3.5.3 Payload

There is no payload in the PUBREC Packet.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

3.5.4 Actions

When the sender of a PUBLISH Packet receives a PUBREC Packet, it MUST reply with a PUBREL Packet.

The action of the recipient is fully described in section 4.3

Deleted: 4.2

3.6 PUBREL – Publish release (QoS 2 publish received, part 2)

A PUBREL Packet is the response to a PUBREC Packet. It is the third packet of the QoS 2 protocol flow.

Deleted: Control

3.6.1 Fixed header

The table below shows the format of the fixed header.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (6)				Dup flag	Reserved		
	0	1	1	0	0	0	1	0
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

Dup flag

See Dup section 1.1.1.1 for details.

Deleted: 2.2.2.1

Bits 2,1 and 0 of the fixed header are reserved and must be set to 0,1 and 0 respectively. The server MUST treat any other value as malformed and close the Network Connection.

Deleted: TCP Connection

Remaining Length field

This is the length of the variable header. For the PUBREL Packet this has the value 2.

3.6.2 Variable header

The variable header contains the same Packet Identifier as the PUBREC Packet that is being acknowledged. The table below shows the format of the variable header.

Bit	7	6	5	4	3	2	1	0
byte 1	PacketId MSB							
byte 2	PacketId LSB							

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

3.6.3 Payload

There is no payload in the PUBREL packet.

3.6.4 Actions

When the sender of a PUBREC Packet receives a PUBREL Packet it MUST reply with a PUBCOMP Packet.

The action of the recipient is fully described in section 4.3

Deleted: 4.2

3.7 PUBCOMP – Publish complete (QoS 2 publish received, part 3)

The PUBCOMP Packet is the response to a PUBREL Packet. It is the fourth and final packet of the QoS 2 protocol flow.

Deleted: Control

3.7.1 Fixed header

The table below shows the format of the fixed header.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (7)				Reserved			
	0	1	1	1	0	0	0	0
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

Remaining Length field

This is the length of the variable header. For the PUBCOMP Packet this has the value 2.

3.7.2 Variable header

The variable header contains the same Packet Identifier as the acknowledged PUBREL Packet.

Bit	7	6	5	4	3	2	1	0
byte 1	PacketId MSB							
byte 2	PacketId LSB							

3.7.3 Payload

There is no payload in the PUBCOMP Packet.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

3.7.4 Actions

When the sender of a PUBREL receives a PUBCOMP Packet it removes any remaining state associated with the original PUBLISH Packet.

The action of the recipient is fully described in section 4.3

Deleted: 4.2

3.8 SUBSCRIBE - Subscribe to topics

The SUBSCRIBE Packet is sent from the Client to the Server to register an interest in one or more Topics. Application Messages that match the topic filter are delivered to the Client using a PUBLISH Packet. The SUBSCRIBE Packet also specifies the maximum QoS with which the server sends publications to the Client.

Deleted: Control

Deleted: Payload Message

Deleted: Control

Deleted: p

3.8.1 Fixed header

The table below shows the format of the fixed header.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (8)				Dup flag	Reserved		
	1	0	0	0	0	0	1	0
byte 2	Remaining Length							

See MQTT Issue-27

Dup flag

Set to zero (0). This means that the packet is being sent for the first time. See DUP, section 1.1.1.1 for more details.

Deleted: 2.2.2.1

Bits 2,1 and 0 of the fixed header are reserved and must be set to 0,1 and 0 respectively. The server MUST treat any other value as malformed and close the Network Connection.

Deleted: TCP Connection

Remaining Length field

This is the length of variable header (2 bytes) plus the length of the payload.

3.8.2 Variable header

The variable header contains a Packet Identifier. See section 2.3.1 for more details.

3.8.2.1 Variable Header Non Normative example

The table below shows an example of the variable header with a Packet Identifier of 10.

	Description	7	6	5	4	3	2	1	0
Packet Identifier									
byte 1	Packet Identifier MSB (0)	0	0	0	0	0	0	0	0

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

byte 2	Packet Identifier LSB (10)	0	0	0	0	1	0	1	0

3.8.3 Payload

The payload of a SUBSCRIBE Packet contains a list of topic filters to which the Client wants to subscribe. The topic filters are UTF-8 encoded strings. Each filter is followed by a byte giving the maximum QoS level at which the Server sends publications to the Client

Topic filters MAY contain special wildcard characters to represent a set of topics, see section 4.6.1 These topic filter / QoS pairs are packed contiguously as shown in the example payload in the table below.

Deleted: 4.5.1

The requested maximum QoS field is encoded in the byte following each UTF-8 encoded topic name as shown in the table below.

Description	7	6	5	4	3	2	1	0
Topic filter								
byte 1	Length MSB							
byte 2	Length LSB							
byte 3-N	Topic filter							
Requested QoS								
	Reserved						QoS	
byte N+1	0	0	0	0	0	0	X	X

The upper 6 bits of this byte are not used in the current version of the protocol. They are reserved for future use.

3.8.3.1 Payload Non Normative Example

Topic name	“a/b”
Requested QoS	0x01
Topic name	“c/d”
Requested QoS	0x02

The format of the example payload is shown in the table below.

	Description	7	6	5	4	3	2	1	0
Topic filter									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

byte 2	Length MSB (3)	0	0	0	0	0	0	1	1
byte 3	‘a’ (0x61)	0	1	1	0	0	0	0	1
byte 4	‘/’ (0x2F)	0	0	1	0	1	1	1	1
byte 5	‘b’ (0x62)	0	1	1	0	0	0	1	0
Requested QoS									
byte 6	Requested QoS(1)	0	0	0	0	0	0	0	1
Topic filter									
byte 7	Length MSB (0)	0	0	0	0	0	0	0	0
byte 8	Length MSB (3)	0	0	0	0	0	0	1	1
byte 9	‘c’ (0x63)	0	1	1	0	0	0	1	1
byte 10	‘/’ (0x2F)	0	0	1	0	1	1	1	1
byte 11	‘d’ (0x64)	0	1	1	0	0	1	0	0
Requested QoS									
byte 12	Requested QoS(2)	0	0	0	0	0	0	1	0

3.8.4 Response

See MQTT Issues 52,58,59,60. Processing of subscribe requests, sending retained publications, storing subscriptions, starting the flow of publications.

When the Server receives a SUBSCRIBE Packet from a Client, the Server MUST respond with a SUBACK Packet. The SUBACK Packet MUST have the same Packet Identifier as the SUBSCRIBE Packet.

The Server MAY start sending PUBLISH packets matching the subscription before the Server sends the SUBACK Packet.

See MQTT Issue-52

Note that if a server implementation does not authorize a SUBSCRIBE request to be made by a subscriber, it has no way of informing that subscriber. It MUST therefore make a positive acknowledgement with a SUBACK, and the subscriber will not be informed that it was not authorized to subscribe.

The SUBACK Packet sent to the Client by the Server contains the maximum QoS for each Topic Filter that the Server will use. The Server might grant a lower level of QoS than the subscriber requested.

The QoS of Application Messages sent in response to a subscription MUST be the minimum of the published message and the granted QoS returned by the Server to the Client.

Non normative comment.

Deleted: Payload Message

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

For example, if a subscriber has requested QoS 1 for a particular topic filter, then a QoS 0 Message matching the filter is delivered to the Client at QoS 0. A QoS 2 Message published to the same topic is downgraded by the Server to QoS 1 for delivery to the Client.

Non normative comment.

A corollary to this is that subscribing to a topic filter at QoS 2 is equivalent to saying "I would like to receive Messages matching this filter at the QoS with which they were published". This means a publisher is responsible for determining the maximum QoS a Message can be delivered at, but a subscriber is able to require that the Server downgrades the QoS to one more suitable for its usage.

3.9 SUBACK – Subscription acknowledgement

A SUBACK Packet is sent by the Server to the Client, to confirm receipt and processing of a SUBSCRIBE Packet.

Deleted: Control

Deleted: p

A SUBACK Packet contains a list of granted QoS levels.

3.9.1 Fixed header

The table below shows the fixed header format.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (9)				Reserved			
	1	0	0	1	0	0	0	0
byte 2	Remaining Length							

Remaining Length field

This is the length of variable header (2 bytes) plus the length of the payload.

3.9.2 Variable header

The variable header contains the Packet Identifier from the SUBSCRIBE Packet that is being acknowledged. The table below shows the format of the variable header.

	7	6	5	4	3	2	1	0
byte 1	Packet Identifier MSB							
byte 2	Packet Identifier LSB							

3.9.3 Payload

The payload contains a list of granted QoS levels. Each QoS level corresponds to a topic filter in the SUBSCRIBE Packet being acknowledged. The order of granted QoS levels in the SUBACK Packet MUST match the order of topic filters in the SUBSCRIBE Packet.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

The table below shows the Granted QoS field encoded in a byte.

Description	7	6	5	4	3	2	1	0
	Reserved						Granted QoS	
byte 1	0	0	0	0	0	0	X	X

The upper 6 bits of this byte are not used in the current version of the protocol. They are reserved for future use.

3.9.3.1 Payload Non Normative Example

Granted QoS	0
Granted QoS	2

The payload for this example is shown in the table below.

	Description	7	6	5	4	3	2	1	0
byte 1	Granted QoS (0)	0	0	0	0	0	0	0	0
byte 2	Granted QoS (2)	0	0	0	0	0	0	1	0

3.10 UNSUBSCRIBE – Unsubscribe from topics

See MQTT Issue-60,64

An UNSUBSCRIBE Packet is sent by the Client to the Server, to unsubscribe from topics.

3.10.1 Fixed header

The table below shows an example fixed header format.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (10)				Dup flag	Reserved		
	1	0	1	0	0	0	1	0
byte 2	Remaining Length							

See MQTT Issue-27

Dup flag

Set to zero (0). This means that the packet is being sent for the first time. See Dup section 2.2.2.1 for more details.

Bits 2,1 and 0 of the fixed header are reserved and must be set to 0,1 and 0 respectively. The server MUST treat any other value as malformed and close the Network Connection.

Deleted: TCP Connection
Deleted: 2
Deleted: 2
Deleted: 25 September 2013

Remaining Length field

This is the length of variable header (2 bytes) plus the length of the payload.

3.10.2 Variable header

The variable header contains a Packet Identifier. See section 2.3.1 for more details.

The table below shows the format of the variable header.

	7	6	5	4	3	2	1	0
byte 1	Packet Identifier MSB							
byte 2	Packet Identifier LSB							

3.10.2.1 Payload

The payload for the UNSUBSCRIBE Packet contains the list of topic filters that the Client wishes to unsubscribe from. The topic filters are UTF-8 string, packed contiguously.

3.10.2.2 Payload Non Normative example

The table below shows an example payload.

Topic filter	“a/b”
Topic filter	“c/d”

The table below shows the format of this payload.

	Description	7	6	5	4	3	2	1	0
Topic Filter									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length MSB (3)	0	0	0	0	0	0	1	1
byte 3	‘a’ (0x61)	0	1	1	0	0	0	0	1
byte 4	‘/’ (0x2F)	0	0	1	0	1	1	1	1
byte 5	‘b’ (0x62)	0	1	1	0	0	0	1	0
Topic Filter									
byte 6	Length MSB (0)	0	0	0	0	0	0	0	0
byte 7	Length MSB (3)	0	0	0	0	0	0	1	1

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

byte 8	'c' (0x63)	0	1	1	0	0	0	1	1
byte 9	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 10	'd' (0x64)	0	1	1	0	0	1	0	0

3.10.3 Response

The Server sends an UNSUBACK Packet to the Client in response to an UNSUBSCRIBE Packet. The UNSUBACK Packet MUST have the same Packet Identifier as the UNSUBSCRIBE Packet.

The Topic Filter (whether containing a wild-card or not) supplied in an unsubscribe MUST be compared byte-for-byte with the current set of Topic Filters held by the Server for the Client. If any filter matches exactly then it is deleted, otherwise no additional processing occurs.

Even where no Topic Filters are deleted, the server MUST respond with an UNSUBACK.

3.11 UNSUBACK – Unsubscribe acknowledgement

The UNSUBACK Packet is sent by the Server to the Client to confirm receipt and processing of an UNSUBSCRIBE Packet.

3.11.1 Fixed header

The table below shows the fixed header format.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (11)				Reserved			
	1	0	1	1	0	0	0	0
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

Remaining Length field

This is the length of the variable header. For the UNSUBACK Packet this has the value 2.

3.11.2 Variable header

The variable header contains the Packet Identifier of the UNSUBSCRIBE Packet that is being acknowledged. The table below shows the format of the variable header.

	7	6	5	4	3	2	1	0
byte 1	Packet Identifier MSB							
byte 2	Packet Identifier LSB							

3.11.3 Payload

There is no payload.

Deleted: ¶
See MQTT Issue-64.

Formatted: Heading 3,H3, Indent: Hanging: 2.38 cm

Deleted: ¶

Deleted: p

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

3.12 PINGREQ – PING request

The PINGREQ Packet is sent from a Client to the Server. It can be used to:

1. Indicate to the Server that the Client is alive in the absence of any other Control Packets flowing from the Client to the Server.
2. Request that the Server responds to confirm that it is alive.
3. Exercise the network to indicate that the Network Connection is active.

Deleted: TCP Connection

This Packet is used in Keep Alive processing, see section 3.1.2.4 for more details.

3.12.1 Fixed header

The table below shows the fixed header format.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (12)				Reserved			
	1	1	0	0	0	0	0	0
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

3.12.2 Variable header

There is no variable header.

3.12.3 Payload

There is no payload.

3.12.4 Response

See MQTT Ussue-54.

The Server MUST send a PINGRESP Packet in response to a PINGREQ packet.

3.13 PINGRESP – PING response

A PINGRESP Packet is sent by the Server to the Client in response to a PINGREQ Packet, it indicates that the server is alive.

This Packet is used in Keep Alive processing, see 3.1.2.4 for more details.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

3.13.1 Fixed header

The table below shows the fixed header format.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (13)				Reserved			
	1	1	0	1	0	0	0	0
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

3.13.2 Variable header

There is no variable header.

3.13.3 Payload

There is no payload.

3.14 DISCONNECT – Disconnect notification

The DISCONNECT Packet is the final Control Packet sent from the Client to the Server. It indicates that the Client is disconnecting cleanly.

3.14.1 Fixed header

The table below shows the fixed header format.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (14)				Reserved			
	1	1	1	0	0	0	0	0
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

The server MUST validate that reserved bits are set to zero and disconnect the client if they are not zero.

3.14.2 Variable header

There is no variable header.

3.14.3 Payload

There is no payload.

3.14.4 Response

After sending a DISCONNECT Packet the Client:

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

- MUST close the Network Connection.
- MUST NOT send any more Control Packets on that Network Connection.

On receipt of DISCONNECT the Server:

- MUST discard the Will Message without publishing it, see 3.1.2.3.2.
- SHOULD close the Network Connection if the client has not already done so.

Deleted: TCP Connection

Deleted: TCP Connection

Deleted: TCP Connection

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

4 Operational behaviour

4.1 Storing state

The client and server implement data storage independently and the duration for which data persists can be different in each. The Client and Server MUST store data for at least as long as the Network Connection lasts. Quality of Service guarantees are only valid so long as both client and server store data. Durable subscriptions and retained publications only survive as long as the Server stores them.

Non normative comment.

An MQTT user should evaluate the storage capabilities of the MQTT Client and Server implementations to ensure that they are sufficient for their needs.

For example, a user wishing to gather electricity meter readings may decide that they need to use QoS 1 messages because they need to protect the readings against loss over the network, however they may decide that the power supply is sufficiently reliable that the data in the Client and Server can be stored in volatile memory without too much risk of its loss.

Conversely a parking meter payment application provider might decide that there are no circumstances where a payment message can be lost so they require that all data are force written to non-volatile memory before it is transmitted across the network.

4.2 Network Connections

The network connection used to transport the MQTT protocol MUST be an ordered, lossless, stream of bytes from the client to server and server to client.

Non normative comment.

The initial transport protocol used to carry MQTT was TCP/IP as defined in The following are also suitable:

- TLS [RFC6455]
- WebSockets [RFC5246]

Connectionless network transports such as User Datagram Protocol (UDP) are not suitable on their own because they might loose or reorder data.

4.3 Quality of Service levels and flows

MQTT delivers Application Messages according to the Quality of Service (QoS) levels defined here. The delivery protocol is symmetric, in the diagrams below the Client and Server can each take the role of either Sender or Receiver. In the case of the Client, “Deliver Application Message” means give the message to the application. In the case of the Server it means send a copy of the Message to each Client with a matching subscription.

Deleted: TCP Connection

Formatted: Default Paragraph Font

Formatted: Font: Not Bold

Formatted: Heading 2,H2

Formatted: Font: Bold

Formatted: Bullets and Numbering

Formatted: Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Bullets and Numbering

Deleted: Payload Message

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

4.3.1 QoS 0: At most once delivery

Formatted: Bullets and Numbering

The message is delivered according to the capabilities of the underlying network. No response is sent by the receiver and no retry is performed by the sender. The message arrives at the receiver either once or not at all.

The diagram below shows the QoS 0 protocol flow.

Sender Action	Control Packet	Receiver Action
PUBLISH QoS 0		
	---------->	
		Deliver Application Message

4.3.2 QoS 1: At least once delivery

Formatted: Bullets and Numbering

The receiver of a QoS 1 PUBLISH Packet acknowledges receipt with a PUBACK Packet. If the Client reconnects and the Session is resumed, the sender MUST resend any in flight Qos 1 messages with the Dup flag set to 1.

See MQTT Issue-68.

or the acknowledgement message is not received after a specified period of time,

The message arrives at the receiver at least once.

A QoS 1 message has a Packet Identifier in its variable header see section 2.3.1.

The diagram below shows the QoS 1 protocol flow.

Sender Action	Control Packet	Receiver action
Store message		
Send PUBLISH QoS 1, Dup 0, <Packet Identifier>	---------->	
		Initiate onward delivery of the Application Message[1]
	<----------	Send PUBACK <Packet Identifier>
Discard message		

[1] The receiver is not required to complete delivery of the Application Message before sending the PUBACK. On receipt, ownership of the Application Message is transferred to the Server. The Server MUST store the message in accordance to its QoS properties and ensure onward delivery to applicable subscribers.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

See MQTT Issue-27.

When it receives a PUBLISH Packet with Dup set to 1 the receiver MUST perform the same actions as above which might result in a redelivery of the Application Message.

4.3.3 QoS 2: Exactly once delivery

This is the highest quality of service, for use when neither loss nor duplication of messages are acceptable. There is an increased overhead associated with this quality of service.

A QoS 2 message has a Packet Identifier in its variable header see section 2.3.1.

The diagram below shows the QoS 2 protocol flow. There are two semantics available for how this flow should be handled by the receiver. They affect the point within the flow at which the message is made available for onward delivery. The choice of semantic is implementation specific and does not affect the guarantees of a QoS 2 flow.

Sender Action	Control Packet	Receiver Action
Store message		
PUBLISH QoS 2 <Packet Identifier> Dup 0		
	---------->	
		Store message or Store <Packet Identifier> then Initiate onward delivery of the Application Message[1]
		PUBREC <Packet Identifier>
	<----------	
Discard message, Store PUBREC received <Packet Identifier>		
PUBREL <Packet Identifier>		
	---------->	
		Initiate onward delivery of the Application Message[1] then discard message or Discard <Packet Identifier>
		Send PUBCOMP <Packet Identifier>
	<----------	
Discard stored state		

Formatted: Bullets and Numbering

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

[1] The receiver is not required to complete delivery of the Application Message before sending the PUBREC or PUBCOMP. On receipt, ownership of the Application Message is transferred to the Server. The Server MUST store the message in accordance to its QoS properties and ensure onward delivery to applicable subscribers.

When a Client reconnects with CleanSession = 0, both the client and server MUST redeliver any previous in-flight messages. This means re-sending any unacknowledged PUBLISH Packets (where QoS > 0) and PUBREL Packets. This is the only circumstance where a Client or Server is REQUIRED to redeliver messages. Clients MAY resend SUBSCRIBE and UNSUBSCRIBE Packets on reconnect but are not REQUIRED to do this.

While a modern TCP network is unlikely to lose packets, a Client or Server is permitted to attempt redelivery at other times. However, redelivery is not encouraged unless a network failure has been detected.

Non-normative comment

Historically retransmission of Control Packets was required to overcome data loss on some older TCP networks. This might remain a concern where MQTT 3.1.1 implementations are to be deployed in such environments.

See Issue MQTT-70

4.4 Message delivery retry

In any network, it is possible for devices or communication links to fail. If this happens, one end of the link might not know what is happening at the other end; these are known as in doubt windows. In these scenarios assumptions have to be made about the reliability of the devices and networks involved in message delivery.

When a Client reconnects, if it is not marked clean session, both the client and server MUST redeliver any previous in-flight messages.

See MQTT Issue-68.

Although TCP normally guarantees delivery of packets, there are certain scenarios where an MQTT message may not be received. In the case of MQTT messages that expect a response (QoS >0 PUBLISH, PUBREL, SUBSCRIBE, UNSUBSCRIBE), if the response is not received within a certain time period, the sender may retry delivery. The sender should set the Dup flag on the message.

The retry timeout should be a configurable option. However care must be taken to ensure message delivery does not timeout while it is still being sent. For example, sending a large message over a slow network will naturally take longer than a small message over a fast network. Repeatedly retrying a timed-out message could often make matters worse so a strategy of increasing the timeout value across multiple retries should be used.

Other than this "on reconnect" retry behavior, clients are not required to retry message delivery. Brokers, however, should retry any unacknowledged message.

Deleted: ¶
See MQTT Issue-68¶
If a failure is detected, or after a defined time period, the protocol flow is retried from the last unacknowledged protocol message; either the PUBLISH or PUBREL. See Message delivery retry for more details. The additional protocol flows ensure that the message is delivered to subscribers once only.

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Formatted ... [4]

Formatted ... [5]

Formatted ... [6]

Formatted ... [7]

Formatted ... [8]

Formatted ... [9]

Formatted ... [10]

Formatted ... [11]

Formatted ... [12]

Formatted ... [13]

Formatted ... [14]

Formatted ... [15]

Formatted ... [16]

Formatted ... [17]

Formatted ... [18]

Formatted ... [19]

Formatted: Bullets and Numbering ... [20]

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

4.5 Message ordering

Formatted: Bullets and Numbering

See MQTT Issue-71

Application Messages from a single Client or Server are delivered in order as long as they have the same QoS and the same Topic Name. The Client or Server MUST respond in the same order that it receives the inbound Control Packets for Application Messages with identical Topic Names and Qos.

Deleted: application message

Non normative comment.

Message ordering can be affected by a number of factors, including how many in-flight PUBLISH Packet flows a client allows and whether the client is single or multi-threaded. For purposes of discussion, clients are assumed to be single-threaded at the point Control Packets are written to and read from the network.

Deleted: -

Deleted: p

For an implementation to provide any guarantees regarding the ordering of messages it must ensure each stage of the message delivery flows are completed in the order they were started. For example, in a series of QoS level 2 flows, the PUBREL Packet flows must be sent in the same order as the original PUBLISH Packet flows:

Client	Direction	Server
PUBLISH 1	---------->	
PUBLISH 2	---------->	
PUBLISH 3	---------->	
	<----------	PUBREC 1
	<----------	PUBREC 2
PUBREL 1	---------->	
	<----------	PUBREC 3
PUBREL 2	---------->	
	<----------	PUBCOMP 1
PUBREL 3	---------->	
	<----------	PUBCOMP 2
	<----------	PUBCOMP 3

Deleted: Message and d

The number of in-flight messages permitted also has an effect on the type of guarantees that can be made:

- Where a single message is in-flight and each delivery flow is completed before the next one starts. This means that messages are carried between the Client and Server in order.
- Where more than one message is in flight, message ordering can only be achieved within the QoS level.

Deleted: are

Deleted: -

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

4.6 Topic Names and Topic Filters

4.6.1 Topic wildcards

A subscription may contain special characters, which allow you to subscribe to multiple topics at once.

The topic level separator is used to introduce structure into the topic, and can therefore be specified within the topic for that purpose. The multi-level wildcard character and single-level wildcard character can be used for subscriptions. Wildcard characters MUST NOT be used within a topic name of a PUBLISH Packet.

4.6.1.1 Topic level separator

The forward slash ('/' 0x2F) is used to separate each level within a topic tree and provide a hierarchical structure to the topic names. The use of the topic level separator is significant when either of the two wildcard characters are encountered in topic filters specified by subscribing Clients. Topic level separators may appear anywhere in a Topic Filter or Topic Name. Adjacent Topic level separators indicate a zero length topic level. Topic names and topic filters are used in their literal form.

4.6.1.2 Multi-level wildcard

The number sign ('#' 0x23) is a wildcard character that matches any number of levels within a topic.

The multi-level wildcard represents the parent and any number of child levels. The multi-level wildcard character MUST be specified on its own or following a topic level separator, in either case it MUST be the last character specified in the topic filter.

Non normative comment.

For example, if a Client subscribes to "sport/tennis/player1/#", it receives messages published using these topic names:

"sport/tennis/player1"
"sport/tennis/player1/ranking"
"sport/tennis/player1/score/wimbledon"

Non normative comment.

- "sport/#" also matches the singular sport, since # includes the parent level.
- "#" is valid and will receive every publication
- "sport/tennis/#" is valid
- "sport/tennis#" is not valid
- "sport/tennis/#/ranking" is not valid

4.6.1.3 Single level wildcard

The plus sign ('+' 0x2B) is a wildcard character that matches only one topic level.

Formatted: Bullets and Numbering
Formatted: Bullets and Numbering
Deleted: The forward slash ('/' 0x2F) is used to separate each level within a topic tree and provide a hierarchical structure to the topic names. The use of the topic level separator is significant when either of the two wildcard characters are encountered in topic filters specified by subscribing Clients.
Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear
Formatted: Bullets and Numbering
Deleted: nadal
Deleted: nadal
Deleted: nadal
Deleted: nadal
Formatted: Bullets and Numbering
Deleted: 2
Deleted: 2
Deleted: 25 September 2013

The single-level wildcard can be used at any level in the topic filter, including first and last levels. Where it is used it MUST occupy an entire level of the filter. It can be used at more than one level in the topic filter and can be used in conjunction with the multilevel wildcard.

Non normative comment.

For example, sport/tennis/+ matches sport/tennis/player1 and sport/tennis/player2, but not sport/tennis/player1/ranking. Also, because the single-level wildcard matches only a single level, sport/+ does not match sport.

Non normative comment.

- "+" is valid
- "+/tennis/#" is valid
- "sport+" is not valid
- "sport/+/player1" is valid
- "/finance" matches "+/+" and "/+", but not "+"

4.6.2 Topics beginning with $

MQTT server implementations MAY define topic names that start with a leading $ character

Non normative comment.

- $SYS/ has been widely adopted as a prefix to topics that contain server-specific information or control APIs
- Applications cannot use a topic with a leading $ character for their own purpose

4.6.2.1 Subscription handling

A topic filter that starts with a wildcard character (# or +) does not match topic names that begin with a $ character

Non normative comment.

- A subscription to # will not receive any messages published to a topic beginning with a $
- A subscription to +/monitor/clients will not receive any messages published to $SYS/monitor/clients
- A subscription to $SYS/# will receive messages published to topics beginning with $SYS/
- A subscription to $SYS/monitor/+ will receive messages published to $SYS/monitor/clients
- For a client to receive messages from topics that begin with $SYS/ and from topics that don't begin with a $, it must subscribe to both # and $SYS/#

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Deleted: The single-level wildcard can be used at any level in the topic filter. It can be used in conjunction with the multilevel wildcard. ¶
It MUST be the only character between two topic level separators, or the last level, or on its own. ¶

Deleted: nadal

Deleted: murray

Deleted: nadal

Deleted: sport/+

Formatted: Font: 10 pt, Font color: Auto, Pattern: Clear

Deleted: nadal

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

4.6.3 Topic semantic and usage

Formatted: Bullets and Numbering

The following rules apply to topic names and topic filters

See MQTT Issue-44

- A topic names and topic filters MUST be at least one character long
- Topic names and topic filters are case sensitive
- Topic names and topic filters can include the space character
- A leading "/" creates a distinct topic name or topic filter
- Topic names and topic filters MUST NOT include the null character (Unicode \x0000)
- Topic names and topic filters are UTF-8 encoded strings, they MUST NOT encode to more than 65535 bytes. Refer section 2.1.2
- There is no limit to the number of levels in a topic name or topic filter

Non normative comment.

- “ACCOUNTS” and “Accounts” are two different topic names
- “Accounts payable” is a valid topic name
- “/finance” is different from “finance”

Non Normative comment.

A publication is sent to each client subscription that matches the topic name in the publication. The topic resource may be either predefined in the server by an administrator or it may be dynamically created by the server when it receives the first subscription or publication with that topic name. The server may also use a security component to selectively authorize actions on the topic resource for a given client.

4.7 Handling protocol violations

Formatted: Bullets and Numbering

See MQTT Issue-8,6,50 etc

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

5 Security

The recommendations contained in this section are provided for guidance only and are not intended to serve as a complete reference on the subject.

There are a number of threats that solution providers should consider. For example:

- Devices may be compromised
- Data at rest in Clients and Servers may be accessible
- Protocol behaviours may have side effects (e.g., 'timing attacks')
- Denial of Service (DoS) attacks
- Communications may be intercepted, altered, re-routed or disclosed
- Injection of spoofed Control Packets

MQTT solutions are often deployed in hostile communication environments. In such cases, implementations will often need to provide mechanisms for:

- Authentication of users and devices
- Authorisation of access to server resources
- Integrity of MQTT Control Packets and application data contained therein
- Privacy of MQTT Control Packets and application data contained therein

As a transport protocol, MQTT is concerned only with message transmission and it is the implementors' responsibility to provide appropriate security features. This is commonly achieved by using TLS.

Server implementations that offer TLS SHOULD use TCP port 8883 [IANA service name: secure-mqtt].

In addition to technical security issues there may also be geographic (e.g., European SafeHarbour), industry specific (e.g., PCI DSS) and regulatory considerations (e.g., Sarbannes-Oxley).

The remainder of this section is non-normative.

5.1 MQTT solutions: security and certification

An implementation may want to provide conformance with specific industry security standards such as NIST Cyber Security Framework, PCI-DSS, FIPS-140-2 and NSA Suite B.

The use of industry proven, independently verified and certified technologies will help meet compliance requirements.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

5.2 Lightweight cryptography and constrained devices

Advanced Encryption Standard (AES) and Data Encryption Standard (DES) are widely adopted

ISO 29192 makes recommendations for cryptographic primitives specifically tuned to perform on constrained 'low end' devices.

5.3 Implementation notes

There are many security concerns to consider when implementing or using MQTT. The following section should not be considered a “check list”.

An implementation might want to achieve some, or all, of the following:

5.3.1 Authentication of Clients by the Server

The CONNECT Packet contains Username and Password fields. Implementations can choose how to make use of the content of these fields. They may provide their own authentication mechanism, use an external authentication system such as LDAP or Oauth tokens, or leverage operating system authentication mechanisms.

Implementations passing authentication data in clear text, obfuscating such data elements or requiring no authentication data should be aware this may give rise to Man-in-the-Middle and replay attacks. Section 5.4.5 introduces approaches to ensure data privacy.

A Virtual Private Network (VPN) between the Clients and Servers can provide confidence that data is only being received from authorised Clients.

Where TLS is used, SSL Certificates flowed from the Client can be used by the Server to authenticate the Client.

An implementation might allow for authentication where the credentials are flowed in an Application Message from the Client to the Server.

5.3.2 Authorisation of Clients by the Server

An implementation may restrict access to Server resources based on information provided by the Client such as Username, ClientID, the hostname/IP address of the Client, or the outcome of authentication mechanisms.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

5.3.3 Authentication of the Server by the Client

The MQTT protocol is not trust symmetrical: it provides no mechanism for the client to authenticate the server.

Where TLS is used, SSL Certificates flowed from the Server can be used by the Client to authenticate the Server. Implementations providing MQTT service for multiple hostnames from a single IP address should be aware of the Server Name Indication extension to TLS [https://tools.ietf.org/html/rfc3546#section-3.1]. This allows a Client to tell the Server the hostname of the Server it is trying to connect to.

An implementation may allow for authentication where the credentials are flowed in an Application Message from the Server to the Client.

A VPN between Clients and Servers can provide confidence that Clients are connecting to the intended Server.

5.3.4 Integrity of Application Messages and Control Packets

Deleted: application message

Applications can independently include hash values in their Application Messages. This can provide integrity of the contents of Publish Control Packets across the network and at rest.

TLS provides hash algorithms to verify the integrity of data sent over the network.

The use of VPNs to connect Clients and Servers can provide integrity of data across the section of the network covered by a VPN.

5.3.5 Privacy of Application Messages and Control Packets

TLS can provide encryption of data sent over the network. There are valid TLS cipher suites that include a NULL encryption algorithm that does not encrypt data. To ensure privacy Clients and Servers should avoid these cipher suites.

An application may independently encrypt the contents of its Application Messages. This could provide privacy of the Application Message both over the network and at rest. This would not provide privacy for other properties of the Application Message such as Topic Name.

Client and Server implementations may provide encrypted storage for data at rest such as Application Messages stored as part of a Session.

The use of VPNs to connect Clients and Servers can provide privacy of data across the section of the network covered by a VPN.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

5.3.6 Non-repudiation of message transmission

Application designers might need to consider appropriate strategies to achieve end to end non-repudiation.

5.3.7 Detecting compromise of Clients and Servers

Client and Server implementations using TLS should provide capabilities to ensure that any SSL certificates provided when initiating a TLS connection are associated with the hostname of the Client connecting or Server being connected to.

Client and Server implementations using TLS may choose to provide capabilities to check Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol (OSCP) to prevent revoked certificates from being used.

Physical deployments might combine tamper-proof hardware with the transmission of specific data in Application Messages. For example a meter might have an embedded GPS to ensure it is not used in an unauthorised location. IEEE 802.1AR is a standard for implementing mechanisms to authenticate a device's identity using a cryptographically bound identifier.

5.3.8 Detecting abnormal behaviours

Server implementations might monitor client behaviour to detect potential security incidents. For example:

- Repeated connection attempts
- Repeated authentication attempts
- Abnormal termination of connections
- Topic scanning (attempts to send or subscribe to many topics)
- Sending undeliverable messages (no subscribers to the topics)
- Clients that connect but do not send data

Server implementations might disconnect clients that breach its security rules.

Server implementations detecting unwelcome behaviour might implement a dynamic block list based on identifiers such as IP address or Client ID.

Deployments might use network level controls (where available) to implement rate limiting or blocking based on IP address or other information.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

5.3.9 Other security considerations

If Client or Server SSL certificates are lost or it is considered that they might be compromised they should be revoked (utilising CRLs and/or OSCP).

Client or Server authentication credentials, such as Username and Password, that are lost or considered compromised should be revoked and/or reissued.

In the case of long lasting connections (such as meters):

- Client and Server implementations using TLS should allow for session renegotiation to establish new cryptographic parameters (replace session keys, change cipher suites, change authentication credentials).
- Servers may disconnect clients and require them to re-authenticate with new credentials.

Constrained devices and clients on constrained networks can make use of TLS session resumption to reduce the costs of reconnecting TLS sessions.

Clients connected to a Server have an transitive trust relationship with other Clients connected to the same Server and who have authority to publish data on the same topics.

5.3.10 Use of SOCKS

Implementations of clients should be aware that some environments will require the use of SOCKSv5 [http://www.ietf.org/rfc/rfc1928.txt] proxies to make outbound Network Connections. Some MQTT implementations may make use of alternative secured tunnels (e.g. SSH) through the use of SOCKS. Where implementations choose to use SOCKS, they should support both anonymous and user-name password authenticating SOCKS proxies. In the latter case, implementations should be aware that SOCKS authentication may occur in plain-text and so should avoid using the same credentials for connection to a MQTT server.

Deleted: TCP connection

5.3.11 Security profiles

Implementors and solution designers may wish to consider security as a set of profiles which can be applied to the MQTT protocol. An example of a layered security hierarchy is presented below.

5.3.11.1 Clear communication profile

MQTT protocol running over an open network with no additional secure communication mechanisms in place.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

5.3.11.2 Secured network communication profile

MQTT protocol running over a physical or virtual network which has security controls e.g., VPNs or physically secure network.

5.3.11.3 Secured transport profile

MQTT protocol running over a physical or virtual network and using TLS which provides authentication, integrity and privacy.

TLS Client authentication may be used in addition to – or in place of – MQTT Client authentication as provided by the Username and Password fields.

5.3.11.4 Industry specific security profile

It is anticipated that the MQTT protocol will be designed into industry specific application profiles, each defining a threat model and the specific security mechanisms to be used to address these threats. Recommendations for specific security mechanisms will often be taken from existing works including:

NIST Cyber Security Framework

NISTIR 7628 Guidelines for Smart Grid Cyber Security

Federal Information Processing Standards (FIPS-140-2)

PCI-DSS

NSA Suite B

An MQTT supplemental publication: MQTT security standards will provide further information related to the usage of various industry security frameworks and standards.

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

6 Conformance

The MQTT specification defines conformance for MQTT Client implementations and MQTT Server implementations.

A single entity MAY conform as both an MQTT Client and MQTT Server implementation. For example, a server that both accepts inbound connections and establishes outbound connections to other servers MUST conform as both an MQTT Client and MQTT Server.

Conformant implementations SHALL NOT require the use of any extensions defined outside of this specification in order to interoperate with any other conformant implementation.

6.1 Conformance Targets

6.1.1 MQTT Server

An MQTT Server accepts Network Connections from MQTT Clients.

Deleted: TCP connection

An MQTT Server conforms to this specification if it satisfies all of the MUST level requirements in the following sections that are identified for the Server:

- [MQTT0001] Section 2 - MQTT Control Packet format
- [MQTT0002] Section 3 - MQTT Control Packets
- [MQTT0003] Section 4 - Operational behaviour
- [MQTT0004] Section 5 - Security

6.1.2 MQTT Client

An MQTT Client creates a Network Connection to an MQTT Server.

Deleted: TCP connection

An MQTT Client conforms to this specification if it satisfies all of the MUST level requirements in the following sections that are identified for the Client:

- [MQTT0005] Section 2 - MQTT Control Packet format
- [MQTT0006] Section 3 - MQTT Control Packets
- [MQTT0007] Section 4 - Operational behaviour
- [MQTT0008] Section 5 - Security

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

Appendix A. Using WebSockets as a network transport.

MQTT can be transported over a WebSocket [RFC 6455] connection using the following conventions:

- WebSocket binary frames are used. A single frame may contain multiple or partial MQTT Control Packets; they are not required to be aligned.
- The WebSocket Protocol Name consists of the MQTT Protocol Name concatenated with the ASCII representation of the MQTT Protocol Version number. For MQTT v3.1.1, this will be "MQTT4".
- No restriction is placed on the path portion of the WebSocket url.

Deleted: 2
Deleted: 2
Deleted: 25 September 2013

Appendix B. Revision history

Revision	Date	Editor	Changes Made
[02]	[29 April 2013]	[A Banks]	[Tighten up language for Connect packet]
[03]	[09 May 2013]	[A Banks]	[Tighten up language in Section 02 Command Message Format]
[04]	[20 May 2013]	[Rahul Gupta]	Tighten up language for PUBLISH message
[05]	[5th June 2013]	[A Banks] [Rahul Gupta]	[Issues -5,9,13] [Formatting and language tighten up in PUBACK, PUBREC, PUBREL, PUBCOMP message]
[06]	[20th June 2013]	[Rahul Gupta]	[Issue – 17, 2, 28, 33] [Formatting and language tighten up in SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK, PINGREQ, PINGRESP, DISCONNECT Control Packets] Terms Command message change to Control Packet Term “message” is generically used, replaced this word accordingly with packet, publication, subscription.
[06]	[21 June 2013]	[A Banks] [Rahul Gupta]	Resolved Issues – 12,20,15, 3, 35, 34, 23, 5, 21 Resolved Issues – 32,39, 41
[07]	[03 July 2013]	[A Banks] [Rahul Gupta]	Resolved Issues – 18,11,4 Resolved Issues – 26,31,36,37
[08]	[19 July 2013]	[A Banks] [Rahul Gupta]	Resolved Issues – 6, 29, 45 Resolved Issues – 36, 25, 24 Added table for fixed header and payload
[09]	[01 August 2013]	[A Banks]	Resolved Issues – 49, 53, 46, 67, 29, 66, 62, 45, 69, 40, 61, 30
[10]	[10 August 2013]	[A Banks] [Rahul Gupta]	Resolved Issues – 19, 63, 57, 65, 72 Conformance section added
[11]	[10 September 2013]	[A Banks] [N O'Leary & Rahul Gupta]	Resolved Issues – 56 Updated Conformance section
[12]	[18 September 2013]	[Rahul Gupta] [A Banks]	Resolved Issues – 22, 42, 81, 84, 85, 7, 8, 14, 16, Security section is added Resolved Issue -1

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

[12]	[27 September 2013]	[A Banks]	Resolved Issues – 64, 68, 76, 86, 27, 60, 82, 55, 78, 51, 83, 80

1723

Deleted: 2

Deleted: 2

Deleted: 25 September 2013

Page 12: [1] Deleted **Andrew_Banks** **03/10/2013 17:29:00**

If 1, the recipient might have previously received the Control Packet it should not treat this flag as an indication that it has previously received the MQTT Control Packet and should not use the Dup flag alone to guarantee to its application that a duplicate message is being redelivered to its application interface.

Page 12: [2] Change **Andrew_Banks** **27/09/2013 14:08:00**

Formatted Bullets and Numbering

Page 22: [3] Deleted **Andrew_Banks** **03/10/2013 17:44:00**

The ClientId MUST comprise only Unicode characters, and the length of the UTF-8 encoding MUST be at least one byte and no more than 65535 bytes.

The server MAY restrict the ClientId it allows in terms of their lengths and the characters they contain, however the server MUST allow ClientIds which are 23 or fewer UTF-8 encoded bytes in length, and contain only the characters "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ".

If the server rejects the ClientId it MUST respond to the CONNECT packet with a CONNACK return code 0x02 (Identifier rejected) and then terminate the TCP connection.

Page 45: [4] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [5] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [6] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [7] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [8] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Font: 10 pt, Font color: Auto

Page 45: [9] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [10] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [11] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [12] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [13] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Font: 10 pt, Font color: Auto

Page 45: [14] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [15] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [16] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Font: 10 pt, Font color: Auto

Page 45: [17] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [18] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Default Paragraph Font, Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [19] Formatted **Andrew_Banks** **27/09/2013 14:12:00**

Font: 10 pt, Font color: Auto, Pattern: Clear

Page 45: [20] Change **Andrew_Banks** **27/09/2013 14:08:00**

Formatted Bullets and Numbering