Abstract/概览（MQTT V3.1）

MQ遥测传输（MQTT）是轻量级基于代理的发布/订阅的消息传输协议，设计思想是开放、简单、轻量、易于实现。这些特点使它适用于受限环境。例如，但不仅限于此：

- 网络代价昂贵，带宽低、不可靠。
- 在嵌入设备中运行，处理器和内存资源有限。

该协议的特点有：

- 使用发布/订阅消息模式，提供一对多的消息发布，解除应用程序耦合。
- 对负载内容屏蔽的消息传输。
- 使用TCP/IP提供网络连接。
- 有三种消息发布服务质量：
- "至多一次"，消息发布完全依赖于底层TCP/IP网络。会发生消息丢失或重复。这一级别可用于如下情况，环境传感器数据，丢失一次读记录无所谓，因为不久后还会有第二次发送。
- "至少一次"，确保消息到达，但消息重复可能会发生。
- "只有一次"，确保消息到达一次。这一级别可用于如下情况，在计费系统中，消息重复或丢失会导致不正确的结果。
- 小型传输，开销很小（固定长度的头部是2字节），协议交换最小化，以降低网络流量。
- 使用LastWill和Testament特性通知有关各方客户端异常中断的机制。

1. Introduction/介绍（MQTT V3.1）

本规范被分成三个主要部分：

- 所有的数据包类型的消息格式
- 每个数据包类型的具体细节
- 数据包如何在服务器及客户端传输

附录里提供了如何使用主题通配符（topic wildcard）的信息

1.1 v3.0与v3.1的一些变化

以下是MQTT V3和MQTT V3.1之间的变化：

- 用户名及密码能在"CONNECT"包里一并发出
- 由于安全问题，"CONNACK"包里定义了新的返回码
- 未认证"PUBLIC"或"SUBSCRIBE"的命令不会通知客户端，即使该命令尚未执行完成MQTT流程
- MQTT现在支持完整的UTF-8字符集，而不是仅仅是US-ASCII subset。

"CONNECT"数据包里协议号仍保持不变，仍然为"3"，现有的MQTT V3服务器实现应能够接受来自客户端的连接，支持本次修订，只要他们正确考虑了的"Remaining Length"字段，只是忽略了额外的安全信息。

2. Message format/消息格式（MQTT V3.1）

每个MQTT命令消息的消息头都包含一个固定的报头。一些消息也要求一个可变的报头和一个payload。下面将描述消息头的格式：

2.1 Fixed header/固定报头（MQTT V3.1）

每个MQTT命令消息的消息头都包含一个固定的报头。下表显示了固定的报头格式。

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

Byte 1

包含消息类型和FLAGS(（DUP，QoS级别，RETAIN）字段

Byte 2

（至少一个字节）包含的剩余长度字段。

这些字段会在以下各部分说明。所有数据值都是bigi-endian（大端）order：高字节在低字节之前。一个16位字先是最高有效位（MSB），其次是最低有效位（LSB）

消息类型

位置：byte 1，bits 7-4

无符号4位值，这个版本的枚举值如下表所示：

Mnemonic	Enumeration	Description
Reserved	0	Reserved
CONNECT	1	Client request to connect to Server
CONNACK	2	Connect Acknowledgment
PUBLISH	3	Publish message
PUBACK	4	Publish Acknowledgment
PUBREC	5	Publish Received (assured delivery part 1)
PUBREL	6	Publish Release (assured delivery part 2)
PUBCOMP	7	Publish Complete (assured delivery part 3)
SUBSCRIBE	8	Client Subscribe request
SUBACK	9	Subscribe Acknowledgment
UNSUBSCRIBE	10	Client Unsubscribe request
UNSUBACK	11	Unsubscribe Acknowledgment
PINGREQ	12	PING Request
PINGRESP	13	PING Response
DISCONNECT	14	Client is Disconnecting
Reserved	15	Reserved

（0->保留；1->客户端请求连接到服务器；2->连接确认；3->发布消息；4->发布确认；5->发布收稿（有保证的交付第1部分）；6->出版发行（有保证交付第2部分）；7->发布完整（有保证的交付第3部分）；8->客户端订阅请求；9->订阅确认；10->客户端退订请求；11->退订确认；12->ping请求；13->ping相应；14->客户端端口；15->保留；）

FLAGS

剩下的三位分别为DUP，QoS及RETAIN字段，bit位置如下表所示

Bit position	Name	Description
3	DUP	Duplicate delivery
2-1	QoS	Quality of Service
0	RETAIN	RETAIN flag

DUP

位置：byte 1，bit 4

这个标识是一个集合，当client或者server企图重
发“PUBLISH”、“PUBREL”、“SUBSCRIBE”、“UNSUBSCRIBE”消息时，这适用于信息的QoS值大于零（0），并且消息确认是必需的。当设置的DUP位，可变头部包含一个消息ID。

接收者应视作为一个提示消息以此标识判断此消息以前是否收到。它不应该依赖于检测重复。

QoS

位置：byte 1，bits 2-1；

此标识指示发送消息的交付质量等级。QoS级别如下表所示：

QoS value	bit 2	bit 1	Description		
0	0	0	At most once	Fire and Forget	<=1
1	0	1	At least once	Acknowledged delivery	>=1
2	1	0	Exactly once	Assured delivery	=1
3	1	1	Reserved		

RETAIN

位置：byte 1，bit 0

这个标识只用于“PUBLISH”消息。当客户端发送一个PUBLISH消息到服务器，如果保留标识位置（1），当消息发送给当前用户服务器应该保留这条消息。

当一个新的订阅者订阅某一个主题，最后保留的某主题消息应被发送到用户并带着“RETAIN”标识位置。如果没有保留消息，则不发送。

这对于发布者发错消息时非常有用，它允许一个新的订阅者立即接收到最重要的数据。

当服务器发送给客户端一个PUBLISH，发现原始的PUBLISH已经存在时，Retain标识不应该被重置，忽略掉原始PUBLISH的Retain标识。也就是说，允许客户端区分那些已经被接收的消息，正在接收的消息，以及即时接收的消息。

保留的消息应该在服务器重启启动后也应该保留下来。

服务器可以删除某个已保留的消息，如果接收的消息长度为0或者保留标识为设置同一个主题。

Remaining Length

位置：byte 2

表示当前的消息剩余的字节数，包括数据在可变头和有效载荷。

可变长度编码方案采用了单字节的消息，多大127个字节长。较长的消息处理方式如下，每个字节7位编码剩余长度的数据，和第八位表示在下面还有值。每个字节编码128个值和一个"延续位"。例如，数字64十进制编码为一个字节的十进制64，十六进制0×40的。

321十进制（=65+2*128）编码为两个字节的重要的最初性最低。

第一个字节65+128=193.需要注意的是最高位表明至少有下列其中一个字节。第二个字节是2。

协议限制最多四个字节表示，这允许应用程序发送消息到268435455（256M）。

这个数字表示：值为0xFF，0xFF，0xFF，0x7F。

下表显示了增加字节数所代表的剩余长度值。

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)

该算法编码成可变长度编码的十进制数（X方案如下）

```
do
  digit = X MOD 128
  X = X DIV 128
  // if there are more digits to encode, set the top bit of this digit
  if ( X > 0 )
    digit = digit OR 0x80
  endif
  ''output'' digit
while ( X> 0 )
```

MOD是模运算符（% in C），DIV是整数除法（/ in C），OR是逐位或（| in C）。剩余长度字段解码算法如下：

```
multiplier = 1
value = 0
do
  digit = ''next digit from stream''
  value += (digit AND 127) * multiplier
  multiplier *= 128
while ((digit AND 128) != 0)
```

AND是操作符（& in C）

这种算法终止时，value包含以字节为单位的剩余长度。

剩余长度编码是不可变头的一部分。字节数用于编码的剩余长度，不利于剩余长度值。可变长度的"扩展字节"是固定的头，而不是可变头的一部分。

2.2 variable header/可变的报头（MQTT V3.1）

MQTT命令消息的某些类型还包括一个variable header组件。它位于fixed header和payload之间。

Remaining Length字段的可变长度不是variable header的一部分。Remaining Length字段的字节不是Remaining Length值的字节计数。它的值只考虑variable header和playload。更多信息查看Fixed header。

Variable header的格式在下面的章节描述，它们必须顺序出现在header里。

协议名称

这个协议名称代表可变头的一个MQTT CONNECT消息。该字段是UTF8编码的字符串，代表协议名称MQIsdp，大写显示。

协议版本

协议版本代表可变头的CONNECT消息。

该字段是一个8位无符号值，描述由客服端所使用的协议的修订基本，当前版本的协议为3（0×03），如下表所示。

bit	7	6	5	4	3	2	1	0
	Protocol Version							
	0	0	0	0	0	0	1	1

Connect flags（连接标志）

The Clean session, Will, Will QoS 和 Retain flags 在可变头的CONNECT消息里是存在的。

Clean session flag

Position：bit 1 of the Connect flags bytes（连接标志字节的第1位）

如果不设置（0），那么服务器必须存储客户端的订阅后断开。这包括继续存储订阅消息QoS1和QoS2消息，以便当客户端重新连接时它们能传递。服务器还必须保持在连接丢失delivered in-flight消息的状态。此信息必须保持，直到客户端重新连接。

如果设置（1），那么服务器必须放弃任何以前客户端维护的信息和让连接保持"clean"。当客户端断开连接时，服务器还必须丢弃一切状态。

通常情况下，一个客户端将在一个模式下或者其他情况下操作，并且不会改变。该选择将取决于application。一个clean的会话的客户端将不会接收到过时的信息，每次连接时它必须重新订阅。一个non-clean会话的客户端不会错过任何QoS1或者QoS2的信息，当它失去连接时会被发布。当它们被发布出去时QoS 0不会被存储。

这个标志以前被称为"Clean start"，它已经被重命名，它适用于整个会话，不仅仅是初始连接。

当这个客户端永远不会重新连接时，一个服务器可以为客户端提供一种清除存储信息的管理机制。

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

在当前版本的协议中这个字节的0位不使用。它是为将来使用保留。

Will flag

Position：bit 2 of the Connect flags byte（连接标志字节的第2位）

Will message定义了当服务器与客户端通信遇到I/O错误或客户端没有在一定时间计划内保持连接时，会发布一个message。服务器从客户端接收到DISCONNECT消息时，并不会触发服务器发送一条Will message。

如果设置Will flag，Will QoS和Will Retain字段必须存在于Connect flags字节中，Will Topic和Will message字段必须在payload里。

Will flag的格式如下表所示：

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

在当前版本的协议中这个字节的0位不使用。它是为将来使用保留。

Will QoS

Position：bits 4 and 3 of the Connect flags byte.（连接标志字节的第4位和3位。）

当一个连接中的客户端被偶然的断开时为一个Will message定义在Will QoS字段中的QoS级别。Will message被定义在CONNECT message的payload中。（A connecting client specifies the QoS level in the Will QoS field for a Will message that is sent in the event that the client is disconnected involuntarily. The Will message is defined in the payload of a CONNECT message.）

如果设置Will flag，Will QoS字段是强制性的，否则他的值将被忽略。

Will QoS的值是0（0×00），1（0×01），or 2（0×02）。Will QoS标志显示在下面的表格中。

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

在当前版本的协议中这个字节的0位不使用。它是为将来使用保留的。

Will Retain flag

Position：bit 5 of the connect flags byte（连接标志字节的第5位）

当客户端意外的断开连接时，Will Retain标志说明服务器是否应该保留被服务器发布的Will message。

如果设置Will Retain标志，Will Retain标志是强制性的，否则它将被忽略。Will Retain标志的格式如下表所示。

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

在当前版本的协议中这个字节的0位不适用。它是为将来使用保留。

User name and password flags（用户名和密码标志）

Position：bits 6 and 7 of the Connect flags byte.（连接标志字节的第6位和第7位）

一个连接中的客户端可以指定一个用户名和密码，设置标志位表示一个用户和密码（可选），包含在一个CONNECT message的payload里。

如果设置User Name标志，User Name字段是强制性的，否则User Name的值是无效的，如果设置Password标志，Password字段是强制性的，否则Password的值是无效的。如果没有设置User Name，而提供Password是无效的。

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

在当前版本的协议中这个字节的0位不使用，它是为将来使用保留。

Keep Alive timer

Keep Alive timer目前在一个MQTT CONNECT message的可变头中。

Keep Alive timer，以秒为单位，定义从客户端收到的消息的最大时间间隔。它使服务器可以检测客户端的网络连接是否已经断开，而不需要等待long TCP/IP timeout。客户端在每个Keep Alive time的间隔内发送一个消息给服务器。在缺乏相关数据消息时，客户端会发送一个PINGREQ消息给服务器，该服务器得到ping请求后悔发送一个PINGRESP消息。

如果在Keep Alive的1.5倍时间间隔内，服务器没有从客户端收到消息，这是客户端如发送了DISCONNECT消息，则断开客户端。此操作不会影响任何客户端的订阅。请参阅有关详细信息DISCONNECT。

发送了一个PINGREQ后，如在一个Keep Alive时间内，客户端没有收到一个PINGRESP消息，就会关闭掉TCP/IP端口连接。

Keep Alive timer是一个16位的数值，表示一段时间的秒数。实际值是特定于应用程序的，但一个典型的值是几分钟。最大值是大约18小时。值为零（0）是指客户端没有断开。

The format of the Keep Alive timer is shown in the table below. The ordering of the 2 bytes of the Keep Alive Timer is MSB, then LSB(big-endian).

bit	7	6	5	4	3	2	1	0
	Keep Alive MSB							
	Keep Alive LSB							

Connect return code（连接返回码）

连接返回码被设置在一个变量头的CONNACK消息中。

该字段定义了一个无符号位的返回码。下面的表格显示的是具体的消息类型的值。返回码0通常表示成功。

Enumeration	HEX	Meaning
0	0x00	Connection Accepted
1	0x01	Connection Refused: unacceptable protocol version
2	0x02	Connection Refused: identifier rejected
3	0x03	Connection Refused: server unavailable
4	0x04	Connection Refused: bad user name or password
5	0x05	Connection Refused: not authorized
6-255		Reserved for future use

bit	7	6	5	4	3	2	1	0
	Return Code							

Topic name（主题名称）

主题名称是在变量头的一个MQTT PUBLISH消息

主题名称是关键，标志payload数据被发布的信息管道。订阅者接收发布的信息通过信息管道。

主题名称是UTF8编码的字符串。参见上节MQTT和UTF-8的更多信息。主题名称有一个较长的字符长度（32,,767字符）。

2.3 Payload(MQTT V3.1)

以下几种类型的MQTT命令消息有一个payload（有效载荷）：

CONNECT:

Payload包含一个或多个UTF-8编码的字符串。它们为客户端指定了一个唯一的标识符，Will Topic，message和User Name和密码使用。但是第一个是可选的，它们的存在是基于标量头的标志。

SUBSCRIBE（订阅）：

Payload包含一个主题名称，该客户端可以订阅的类别和QoS级别。这些字符串是UTF-8编码的。

SUBACK（订阅确认）：

Payload包含一些授权的QoS级别，这些QoS级别允许服务器管理员授权客户端订阅特殊的主题。授权的QoS级别已相同顺序列出作为在相应SUBSCRIBE消息的主题名称。

Payload作为一个PUBLISH消息只包含application-specific数据。任何假设的性质和内容的数据，消息的一部分被看作一个BLOB。

如果你想一个application申请压缩payload数据，你需要在application中定义适当的payload标志字段来处理压缩的详细信息。你不能再固定或者可变的头里定义应用程序特定的标志。

2.4 Message identifiers

消息标识符出现在下面的MQTT消息的可变头中：PUBLISH，PUBACK，PUBREC，PUBREL，PUBCOMP，SUBSCRIBE，SUBACK，UNSUBSCRIBE，UNSUBACK。

消息标识符（message id）字段仅在固定头中的QoS位中出现，表示QoS级别1或2的消息。更多信息，参见Quality of Service levels and flows。

消息标识符是一个16位无符号位的整数，在一个特定方向的通信的"in flight"消息中必须唯一。它精确的增长通过一个消息到下一个消息，但是不要求这样做。

客户端将维护自己的连接到服务器使用的IDs的消息列表。客户端很有可能同时发送和接收一个Message ID为1的PUBLISH。

消息标识符的两个字节的顺序是MSB，LSB（大端）。

不要使用消息ID 0。它是保留作为无效的消息ID。

bit	7	6	5	4	3	2	1	0
	Message Identifier MSB							
	Message Identifier LSB							

2.5 MQTT and UTF-8(MQTT V3.1)

UTF-8是一个有效Unicode字符的字符串编码，它优化支持基于文本通信的ASCII字符编码。

在MQTT中，前面两个字节的字符串表示长度，如下表所示：

bit	7	6	5	4	3	2	1	0
byte 1	String Length MSB							
byte 2	String Length LSB							
bytes 3 ...	Encoded Character Data							

字符串长度是编码后的字符串的字节数而不是字符的个数，例如，字符串OTWP是UTF-8编码，如下表所示：

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 (0x04)							
	0	0	0	0	0	1	0	0
byte 3	'O' (0x4F)							
	0	1	0	0	1	1	1	1
byte 4	'T' (0x54)							
	0	1	0	1	0	1	0	0
byte 5	'W' (0x57)							
	0	1	0	1	0	1	1	1
byte 6	'P' (0x50)							
	0	1	0	1	0	0	0	0

Java中的writeUTF()和readUTF()数据流的方法使用了这种格式。

2.6 Unused bits(MQTT V3.1)

任何位被标记为未使用应该设置为零（0）

Any bits marked as unused should be set to zero(0)。

3. Command messages(MQTT V3.1)

3.1 CONNECT-Client requests a connection to a server(MQTT V3.1)

当客户端向服务器建议一个TCP/IP端口连接，协议基本会话必须使用一个CONNECT flow建立。

Fixed header（固定头）

Fixed header的格式如下表所示。

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (1)				DUP flag	QoS level		RETAIN
	0	0	0	1	x	x	x	x
byte 2	Remaining Length							

DUP，QoS和RETAIN标志不能被使用在CONNECT消息中。

Remaining Length是可变头（12字节）的长度和payload的长度。这可以是一个多字节的字段。

Variable header（可变头）

可变头的格式如下表所示：

	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 (6)	0	0	0	0	0	1	1	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	'I'	0	1	0	0	1	0	0	1
byte 6	's'	0	1	1	1	0	0	1	1
byte 7	'd'	0	1	1	0	0	1	0	0
byte 8	'p'	0	1	1	1	0	0	0	0
Protocol Version Number									
	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 9	Version (3)	0	0	0	0	0	0	1	1
Connect Flags									
byte 10	User name flag (1) Password flag (1) Will RETAIN (0) Will QoS (01) Will flag (1) Clean Session (1)	1	1	0	0	1	1	1	x
Keep Alive timer									
byte 11	Keep Alive MSB (0)	0	0	0	0	0	0	0	0
byte 12	Keep Alive LSB (10)	0	0	0	0	1	0	1	0

User name flag

Set(1)

PasswSord flag

Set(1)

Clean Session flag

Set(1)

Keep Alive timer

设置为10秒（0x000A）

Will message

- Will flag is set(1)
- Will QoS field is 1
- Will RETAIN flag is clear(0)

Payload

CONNECT消息的payload包含一个或多个的UTF-8编码的字符串，基于可变头的基础。如果字符串存在，必须按下列顺序出现。

Client Identifier

第一个UTF编码的字符串。客户端标识符（Client ID）是介于1和23个字符长度，客户端到服务器的唯一标识。它必须在搜有客户端连接到一台服务器是唯一的，是在处理QoS级别1和2的消息ID中的关键。如果客户端ID包含23个字符，服务器响应CONNECT消息，通过一个CONNACK，返回码2：标识符被拒绝。

Will Topic

如果设置Will Flag，这是下一个UTF编码字符串。Will Message发布Will Topic。QoS级别通过Will QoS字段定义和RETAIN状态通过Will RETAIN标识定义在可变头里。

Will Message

如果设置Will Flag，这是下一个UTF编码字符串。Will Message定义消息的内容，发布Will Topic如果客户端意外的断开。这可能是一个长度为0的消息。

虽然在CONNECT消息中的Will Message是UTF编码的，当它发布Will Topic仅仅是消息的字节数被发送，而不是前两个字节的长度。因此，该消息必须只包含7位ASCII字符。

User Name

如果设置User Name标识，他是下一个UTF编码的字符串。User name标识连接的用户的名字，可用于身份验证。建议用户名为12个字符或者更少，但它不是必须的。

请注意，为了兼容原来的MQTT V3规范，固定头的Remaining Length字段优先于User Name标识。服务器必须允许实现设置User Name标识的可能性，但是User Name字符串正在缺少。这是有效的，应允许继续连接。

Password

如果设置Password标识，它是下一个UTF编码的字符串。Password标识连接的用户的名字，可用于身份验证。建议密码为12个字符或者更少，但它不是必须的。

请注意，为了与兼容原来的MQTT V3规范，固定头的Remaining Length字段优先于Password标识。服务器必须允许实现设置Password标识的可能性，但是Password字符串是缺少的（but the Pass）。This is valid, and connections should be allowed to continue.

Response

服务器发送一个CONNACK消息作为客户端发送的CONNECT消息的响应。

如果服务器建立TCP/IP连接后，在合理的时间内没有收到一个CONNECT消息，服务器应关闭连接。

如果客户端在一个合理的时间内没有收到从服务器的一个CONNACK消息，客户端应该关闭TCP/IP端口连接，并重新打开一个新的端口服务器并发出一个CONNECT消息会话。

在这两种情况，一个“合理”的时间量取决于应用和通信基础设施的类型。

如果一个具有相同Client ID的客户端已经连接到服务器，先前的客户端必须断开连接后，服务器才能完成新的客户端的CONNECT连接。

如果客户端发送一个无效的CONNECT消息，服务器应当关闭连接。这包括提供无效的协议名称或协议版本号的CONNECT消息。如果服务器可以充分的解析CONNECT消息，以确定一个无效的协议要求，服务器可能会在断开连接前尝试发送CONNACK包含“连接被拒绝，不能接受的协议版本”的代码。

3.2 CONNACK – Acknowledge connection request

CONNACK消息是由服务器发送给一个来自客户端的CONNECT的响应。

Fixed header（固定头）

固定头的格式如下表所示：

bit	7	6	5	4	3	2	1	0
byte 1	Message type (2)				DUP flag	QoS flags		RETAIN
	0	0	1	0	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

CONNACK消息中未使用的DUP，QoS和RETAIN标志。

Variable header（可变头）

可变头的格式如下表所示。

	Description	7	6	5	4	3	2	1	0
Topic Name Compression Response									
byte 1	Reserved values. Not used.	x	x	x	x	x	x	x	x
Connect Return Code									
	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 2	Return Code								

一个无符号字节连接返回代码字段的值如下表所示：

Enumeration	HEX	Meaning
0	0x00	Connection Accepted
1	0x01	Connection Refused: unacceptable protocol version
2	0x02	Connection Refused: identifier rejected
3	0x03	Connection Refused: server unavailable
4	0x04	Connection Refused: bad user name or password
5	0x05	Connection Refused: not authorized
6-255		Reserved for future use

如果一个唯一的客户端标识符不是在1和23个字符长度之间发送，返回码为2（identifier rejected）

Payload（有效载荷）

There is no payload。

3.3 PUBLISH – Publish message

服务器为订阅了此消息的客户端发布一个消息，每个PUBLISH消息与一个主题相关（又称主题或频道）。这是一个分层的名称空间，定义了一个信息源分类，用户可以有兴趣的注册。一个消息发布到一个特定的主题名称被传递到该主题的连接订阅者。

如果客户端订阅了一个或多个主题，服务器发布任何主题的消息给客户端作为一个PUBLISH消息。

Fixed header（固定头）

下表显示了固定头的格式：

bit	7	6	5	4	3	2	1	0
byte 1	Message type (3)				DUP flag	QoS level		RETAIN
	0	0	1	1	0	0	1	0
bit	7	6	5	4	3	2	1	0
byte 2	Remaining Length							

QoS level

设置为1。有关详细信息，请参阅QoS

DUP flag

设置为零（0）。这意味着消息顶一次被发送。请参阅DUP获得更多详情。

RETAIN flag

设置为零。这意味不保留。有关详细信息，请参阅Retain。

Remaining Length field

可变头的长度加上payload的长度。它可以是一种多字节字段。

Variable header

可变头包含以下字段：

Topic name

一个UTF-8编码的字符串。

不能含有Topic通配字符。

当接收到一个由客户端使用通配字符订阅的Topic，这个字符串将被原始的发布者指定特定的绝对主题，并且不会被客户端订阅字符使用。

Message ID

代表QoS level 1 和QoS level 2消息。更多详情请查看Message identifiers。

下表显示的是一个PUBLISH消息的可变头的例子。

Field	Value
Topic Name:	"a/b"
QoS level	1
Message ID:	10

在这种情况下，可变头的格式如下表：

	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
Message Identifier									
byte 6	Message ID MSB (0)	0	0	0	0	0	0	0	0
byte 7	Message ID LSB (10)	0	0	0	0	1	0	1	0

Payload（有效载荷）

包含发布的数据。数据格式的内容和具体应用。固定头的剩余长度字段包括可变头长度和有效载荷的长度。因此，发布含有0长度的有效载荷是有效的。

Response

发布消息的响应取决于QoS基本。下表显示了预期的反应。

QoS Level	Expected response
QoS 0	None
QoS 1	PUBACK
QoS 2	PUBREC

Actions

可以发布消息从publisher发送到服务器，或从服务器到subscriber。当它接收到一条消息时接收者的行动取决于消息的QoS level：

QoS 0

提供给任何有兴趣的各方。

QoS 1

记录消息到持久存储，提供给任何有兴趣的各方，并返回给发件人一个PUBACK消息。

QoS 2

记录消息到持久存储，不提供给有关方面，并返回给发件人一个PUBREC消息。

如果服务器收到一个消息，有兴趣的部分是指订阅者发布消息的主题。如果订阅者收到一个消息，有兴趣的部分，是指客户端的应用程序正在等待一个或多个来自服务器中已被订阅的一个或多个主题的消息。

更多细节请参看Quality of Service levels and flows

请注意，如果一台服务器的实现并不授权发布到客户端，它没有通知该客户端的方式。因此，它必须做出积极的确认，按照正常的QoS规则，客户端将不会被告知，这是未经授权的发布消息。

3.4 PUBACK-publish acknowledgment

一个PUBACK消息时一个PUBLISH消息的QoS级别1的响应。一个PUBACK消息通过服务器发布，作为一个来自从发布客户端的响应，和作为一个订阅者的来自服务器的PUBLISH消息的响应。

Fixed header

下表显示了固定头的格式

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (4)				DUP flag	QoS level		RETAIN
	0	1	0	0	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

QoS level

没有被使用。

DUP flag

没有被使用。

RETAIN flag

没有被使用。

Remaining Length field

这是可变头的长度（2字节）。它可以是一个多字节字段。

Variable header

包含一个被确认的PUBLISH消息的消息标识符（Message ID）。下表显示的是可变头的格式。

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

Payload

There is no payload.

Actions

当客户端收到一个PUBACK消息时，他摒弃掉原来的消息，因为它也是通过服务器接收（和记录）。

3.5 PUBREC–Assured publish received(part 1)

一个PUBREC消息是一个QoS级别2的PUBLISH消息的响应。这是第二个消息的QoS级别2协议流。一个PUBREC消息被发送作为一个正在发布的客户端的一个PUBLISH消息响应，或者是作为一个来自服务器的一个PUBLISH消息响应。

Fixed header

下表显示了固定头的格式：

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (5)				DUP flag	QoS level		RETAIN
	0	1	0	1	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

QoS level

没有被使用。

DUP flag

没有被使用。

RETAIN flag

没有被使用。

Remaining Length field

这是可变头的长度（2字节）。它可以是一个多字节字段。

Variable header

包含一个被确认的PUBLISH消息的消息标识符（Message ID）。下表显示的是可变头的格式。

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

Payload

There is no payload.

Actions

当它接收到一个PUBREC的消息，收件人发送一个PUBREL的消息给具有相同PUBREC消息的Message ID的发件人。

3.6 PUBREL-Assured Publish Release(part 2)

一个PUBREL消息，是发布者发布PUBREC消息从服务器，或用户订阅一个PUBREC消息从服务器的响应。它是在QoS 2协议流的第三个消息。

Fixed header

该表显示的是固定头的格式：

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (6)				DUP flag	QoS level		RETAIN
	0	1	1	0	0	0	1	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

QoS level

PUBREL消息使用QoS级别1，确认为预计PUBCOMP形式。以同样的方式作为发布消息的重试处理。

DUP flag

设置零（0）。这意味着该消息被第一次发送。更多细节参看DUP。

RETAIN flag

未被使用。

Remaining Length field

这是可变头的长度（2字节）。它可以是一个多字节字段。

Variable header

可变头包含相同的Message ID的PUBREC消息将被确认。该表显示了可变头的格式。

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

Payload

There is no payload.

Actions

当服务器接收到一个发布者的PUBREL消息，服务器提供原来的消息给有兴趣的订阅者，并发送一个PUBCOMP消息给具有相同Message ID的发布者。当订阅者接收来自服务器的PUBREL的消息，用户订阅应用程序提供的消息，并发送一个PUBCOMP消息给服务器。

3.7 PUBCOMP-Assured publish complete(part 3)

此消息是一个发布者从服务器到一个PUBREL的消息响应，或一个订阅者从服务器到一个PUBREL消息的响应。它是在QoS2协议流的第四个和最后的消息。

Fixed header

该表显示了固定头的格式：

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (7)				DUP flag	QoS level		RETAIN
bit	7	6	5	4	3	2	1	0
	0	1	1	1	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

QoS level

未使用。

DUP flag

未使用

RETAIN flag

未使用

Remaining Length field

这是可变头的长度（2字节）。它可以是一个多字节字段。

Variable header

可变头包含相同的Message ID的PUBREC消息被确认。该表显示了可变头的格式。

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

Payload

There is no payload

Actions

当客户端接收一个PUBCOMP消息时，客户端摒弃原来的消息，因为它已经发送一次到服务器。

3.8 SUBSCRIBE-Subscribe to named topics

订阅消息，允许客户端注册一个或多个对服务器有兴趣主题名称。从服务器发布这些主题的消息传送到客户端作为一个PUBLISH消息。订阅消息还制定了QoS级别，用户要接收发布的消息。

Fixed header

下表显示了固定头的格式：

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (8)				DUP flag	QoS level		RETAIN
	1	0	0	0	0	0	1	x
byte 2	Remaining Length							

QoS level

订阅消息使用QoS级别1来确认多个订阅请求。通信中的SUBACK消息通过匹配Message ID来识别。以同样的方式作为PUBLISH消息的重试处理。

DUP flag

设置为零（0）。这意味着消息正在被第一次发送。更多详细情况请查看DUP。

RETAIN flag

未使用

Remaining Length field

Payload的长度。它可以是一个多字节字段。

Variable header

可变头包含一个Message ID，因为一个SUBSCRIBE消息有一个位1的QoS级别。更多详细情况参考Message identifiers。

下表显示了一个可变头的Message ID为10的格式的例子。

	Description	7	6	5	4	3	2	1	0
Message Identifier									
byte 1	Message ID MSB (0)	0	0	0	0	0	0	0	0
byte 2	Message ID LSB (10)	0	0	0	0	1	0	1	0

Payload

订阅消息的有效载荷包含一个客户端希望订阅的主题名称列表，并在客户端要接收消息的QoS级别。字符串是UTF编码，QoS级别占用一个字节的高2位。主题串还可能包含指定通配符，代表一套主题。这些topic/QoS成对连续包装，如下表的有效载荷的一个例子。

Topic name	"a/b"
Requested QoS	1
Topic name	"c/d"
Requested QoS	2

在一个SUBSCRIBE消息的主题名称是不会被压缩的。

Payload例子的格式如下表所示：

	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
Requested QoS									
byte 6	Requested QoS (1)	x	x	x	x	x	x	0	1
Topic Name									
byte 7	Length MSB (0)	0	0	0	0	0	0	0	0
byte 8	Length LSB (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)	x	x	x	x	x	x	1	0

假设被请求的QoS级别被授予，客户端接收小于或等于这个级别的PUBLISH消息，根据发布者的原始消息的QoS级别。例如，如果客户端的QoS级别为1订阅特定主题，然后一个QoS级别0发布消息，该主题将被传递到客户端QoS级别0。一个QoS级别2消息发布同一个主题，传递给客户端QoS降级，QoS级别1.

这样做的一个结果是订阅一个QoS级别2的主题，就等于说："我想收到这个被它们发布的QoS主题的消息"。

这意味着发布者负责确定传递一个最大的QoS消息，但一个订阅者能降级QoS到一个更适合它使用的QoS。一个QoS消息时永远不会升级的。

被请求的QoS字段被编码的字节，每个UTF编码的主题名称如下表所示：

bit	7	6	5	4	3	2	1	0
	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	QoS level	
	x	x	x	x	x	x		

高六位的字节在当前协议中未被使用。被保留以供将来使用。

一个请求包含所有QoS级别位应被视为无效的数据包，并关闭连接。

Response

当从客户端接收一个SUBSCRIBE消息时，服务器返回一个SUBACK消息。

由于在订阅之前，客户端收到一个SUBACK消息，服务器可能会发送一个PUBLISH消息。

请注意，如果服务器实现没有授权一个客户端的SUBSCRIBE请求，它没有通知该客户端的方式。因此，它必须与SUBACK的积极确认，客户端将不会被告知，该订阅是不授权的。

一台服务器可以选择给予比客户端请求较低的QoS级别。如果服务器不能提供较高的QoS级别，该情况也是有可能发生的。例如，如果服务器不能提供一个可靠的持久性机制，它可能会选择只授予QoS 0级别的订阅。

3.9 SUBACK – Subscription acknowledgement

一个SUBACK消息是由服务器发送到客户端确认收到了SUBSCRIBE消息。

一个SUBACK消息包含授予QoS级别的列表。SUBACK消息中授予的QoS级别的顺序与在相应的SUBSCRIBE消息的主题名称的顺序相匹配。

Fixed Header

下表显示了固定头的格式。

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (9)				DUP flag	QoS level		RETAIN
	1	0	0	1	x	x	x	x
byte 2	Remaining Length							

QoS level

未被使用

DUP flag

未被使用。

RETAIN flag

未被使用。

Remaining Length field

Payload的长度，它可以是一个多字节的字段。

Variable header

可变头包含了SUBSCRIBE消息正在被确认的Message ID。下表显示了可变头的格式。

	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

Payload

有效载荷包含一个授予QoS级别矢量。每个级别都对应一个相应的SUBSCRIBE消息的主题名称。SUBACK消息的QoS级别的顺序匹配SUBSCRIBE消息topic name和Requested QoS成对顺序出现。变量头的Message ID能匹配相应的SUBSCRIBE消息和SUBACK消息。

下表显示了Granted QoS字段在一个字节中的编码。

bit	7	6	5	4	3	2	1	0
	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	QoS level	
	x	x	x	x	x	x		

高六位的字节在当前协议中未被使用。被保留供将来使用。

下表显示了一个payload例子。

Granted QoS	0
Granted QoS	2

下表显示了payload的格式。

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

3.10 UNSUBSCRIBE – Unsubscribe from named topics

一个UNSUBSCRIBE消息通过客户端发送给服务器，用来退订named topic。

Fixed header

下表显示了一个固定头的格式的一个例子。

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (10)				DUP flag	QoS level		RETAIN
	1	0	1	0	0	0	1	x
byte 2	Remaining Length							

QoS level

UNSUBSCRIBE消息使用QoS级别1来获得多个退订请求。相应的UNSUBSCRIBE消息是通过Message ID来标识的。以同样的方式作为PUBLISH消息的重试处理。

DUP flag

设置零（0）。这意味着这个消息正在第一次被发送。更多详细情况参考DUP。

RETAIN flag

未被使用。

Remaining Length

Payload的长度。它可能是一个多字节字段。

Variable header

可变头包含一个Message ID，因为一个UNSUBSCRIBE消息有一个为1的QoS级别。详细参看Message identifiers。

下表显示了一个可变头的Message ID为10的格式的例子。

	Description	7	6	5	4	3	2	1	0
Message Identifier									
byte 1	Message ID MSB (0)	0	0	0	0	0	0	0	0
byte 2	Message ID LSB (10)	0	0	0	0	1	0	1	0

Payload

来自有效载荷的named topic列表的客户端退订者。字符串是UTF编码的，并连续包装。在退订邮件的named topic是不会被压缩。下表显示了一个有效载荷的例子。

（The client unsubscribes from the list of topics named in the payload. The strings are UTF-encoded and are packed contiguously. Topic names in a UNSUBSCRIBE message are not compressed.）

Topic Name	"a/b"
Topic Name	"c/d"

下表显示了一个有效载荷的格式：

	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
Topic Name									
byte 6	Length MSB (0)	0	0	0	0	0	0	0	0
byte 7	Length LSB (3)	0	0	0	0	0	0	1	1
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

Response

服务器发送一个UNSUBACK到客户端来响应一个UNSUBSCRIBE消息。

3.11 UNSUBACK-Unsubscribe acknowledgment

UNSUBACK消息是服务器发送到客户端，用来确认接收一个UNSUBSCRIBE消息。

Fixed header

下表显示了固定头的格式

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (11)				DUP flag	QoS level		RETAIN
	1	0	1	1	x	x	x	x
byte 2	Remaining length (2)							
	0	0	0	0	0	0	1	0

QoS level

未被使用。

DUP flag

没有被使用。

RETAIN flag

没有被使用。

Remaining Length

可变头的长度（两个字节）。

Variable header

可变头包含一个被确认的UNSUBSCRIBE消息的Message ID，下表显示了可变头的格式。

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

Payload

There is no payload.

3.12 PINGREQ-PING request

PINGREQ消息是一个从客户端发送到服务器的“are you alive?”消息。

更多详细情况参看 Keep Alive timer。

Fixed header

下表显示固定头的格式。

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (12)				DUP flag	QoS level		RETAIN
	1	1	0	0	x	x	x	x
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

DUP，QoS，和RETAIN标志没有被使用。

Variable header

There is no variable header.

Payload

There is no payload

Response

PINGREQ消息的响应是一个PINGRESP消息。

3.13 PINGRESP – PING response

PINGRESP消息是从服务器发送给PINGREQ消息的响应，说明"yes I am alive"。

更多细节参看Keep Alive timer

Fixed header

下表显示了固定头的格式。

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (13)				DUP flag	QoS level		RETAIN
	1	1	0	1	x	x	x	x
bit	7	6	5	4	3	2	1	0
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

DUP，QoS，和RETAIN标志没有被使用。

Payload

There is no payload

Variable header

There is no variable header.

3.14 DISCONNECT-Disconnect notification

DISCONNECT消息是从客户端发送到服务器，以表明它即将关闭TCP/IP连接。这允许一个干净的断开，而不是只是删除了一条记录。

如果客户端连接使用了clean session标志，所有一切保持对客户端的信息将被摒弃。

服务器在接收一个DISCONNECT消息后，不依赖客户端关闭TCP/IP连接。

Fixed header

下表显示了固定头的格式。

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (14)				DUP flag	QoS level		RETAIN
	1	1	1	0	x	x	x	x
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

DUP，QoS，和RETAIN标志在DISCONNECT消息中没有被使用。

Payload

There is no payload.

Variable header

There is no variable header.

4. Flows(MQTT V3.1)

4.1 Quality of Service levels and flows

MQTT提供的消息，根据在服务质量（QoS）的定义水平。各级别描述如下:

QoS level 0: At most once delivery(至多一次)

消息发布完全依赖底层TCP/IP网络。在协议中一个响应是没有定义预期的和重试语义的。消息到达服务器一次或者一次没有。

下表显示了QoS level 0协议流量。

Client	Message and direction	Server
QoS = 0	PUBLISH ---------->	**Action:** Publish message to subscribers

QoS level 1: At least once delivery(至少一次)

服务器收到的消息是由PUBACK消息确认。如果在指定的时间段内没有收到通信连接或者发送设备，或者没有确认消息是一个确定的失败，发送者使用在头消息中设置DUP位重新发送消息。

消息到达服务器至少一次。SUBSCRIBE和UNSUBSCRIBE消息使用的是QoS level 1。

一个QoS level 1消息在消息头中有一个Message ID。下表显示了QoS level 1协议流。

Client	Message and direction	Server
QoS = 1 DUP = 0 Message ID = x **Action:** Store message	PUBLISH ---------->	**Actions:** • Store message • Publish message to subscribers • Delete message
Action: Discard message	PUBACK <----------	

如果客户端没有收到PUBACK消息（无论在应用程序中定义的一个时间段还是如果检测到故障，并重新启动通信会话），客户端可能会重新发送一个设置了DUP标志的PUBLISH消息。

当它从客户端接收到重复的消息，服务器重新发布消息给订阅者，并发送另一个PUBACK消息。

QoS level 2: Exactly once delivery(只有一次)

在QoS level 1以上的附加协议流，确保重复消息不会被传递给正在接收的应用程序。当重复消息不被接收的时候，这是最高级别的传送。在网络中是比较常见的，但是它通常是可以接受的，因为消息内容的重要性。

一个QoS level 2的消息在消息头中有一个Message ID。

下表显示了QoS level 2协议流。PUBLISH流应如何被接收者处理有两种语义。它们影响向订阅者提供的消息。语音的选择的具体实施并不会影响QoS level 2流。

Client	Message and direction	Server
QoS = 2 DUP = 0 Message ID = x **Action:** Store message	PUBLISH ---------->	**Action:** Store message *or* **Actions:** • Store message ID • Publish message to subscribers
	PUBREC <----------	Message ID = x
Message ID = x	PUBREL<br---------->	**Actions:** • Publish message to subscribers • Delete message *or* **Action:** Delete message ID
Action: Discard message	PUBCOMP <----------	Message ID = x

如果检测到故障，或在规定的时间内，协议流被重试从最近未被确认的协议消息；PUBLISH或者PUBREL。

有关详细信息，请参阅Message delivery retry。附加协议流量保证消息传递给订阅者，只有一次。

Assumptions for QoS levels 1 and 2

在任何网络，设备或通信链路都有失败的可能。如果发生这种情况，链接的一端可能不知道在另一端的情况；这些被知道的疑问问题。在这些情节假设有你要做出的关于可靠性的设备和网络有关的消息交付。

MQTT假设客户端和服务器一般可靠，通信通道更可能是不可靠的。如果客户端设备发生故障，它通常是一个灾难性的故障，而不是一个短暂的。从设备中恢复数据的可能性很低。有些设备具有非易失性存储，例如闪存ROM。提供更多的客户端设备上的持久性存储，保护最关键的数据在一些故障中。

除了基本的通信链路故障，故障模式的矩阵变得复杂，导致更多的场景比MQTT规范可以处理。

4.2 Message delivery retry

尽管TCP通常保证数据包传递，在某些情况下有可能无法接收MQTT消息。在MQTT消息期望一个响应情况下（QoS>0 PUBLISH, PUBREL, SUBSCRIBE, UNSUBSCRIBE），如果响应没有被接收在一定时间内，发送者可以重新传递，发送者应该在消息中设置DUP标志。

当一个客户端重新连接，如果它没有标记clean session，客户端和服务器应该重新传递任何先前in-flight的消息。

除此之外"连接上的"重试行为，客户并不需要重试消息传递。然而，broker应该重试任何未确认的消息。

4.3 Message ordering

Message ordering会受许多因素的影响，包含客户端允许多少in-flight PUBLISH流，和客户端是否是单线程或者多线程。讨论的目的，客户端被假设是单线程的在某一个数据包从网络中写入和读出。

对于实施提供任何保证消息的顺序，它必须保证消息传递流程的每个阶段，都在他们启动的顺序完成。例如，在一系列的QoS level 2 流，PUBREL流必须在原PUBLISH流相同的顺序发送：

Client	Message and 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 <----------	

in-flight消息的数量允许有一个可保证的效果：

- 在in-flight窗口1中，每个传递流在下一个开始之前完成。这保证消息以提交的顺序传递。
- 在大于in-flight窗口1，消息排序只能在QoS level内被保证。

Appendix A – Topic wildcards

订阅可能包含特殊字符，让您一次订阅多个主题。

主题级别的分隔符是用来引入主题的结构，因此，可以指定用于这一目的的主题内。多层次的通配符和单层次的通配符可以用于订阅，但他们不能再一个主题内发布消息。

Topic level serparator

斜线（/）是用来分隔一个主题树内的每个级别的层次结构，并提供主题空间。

话题水平分隔是显著的，当遇到两个通配符是由订阅者指定的主题。

Multi-level wildcard

数字符号（#）是通配符匹配一个主题内的任何级别的数量。例如，如果您订阅了finance/stock/ibm/#，您会收到有关这些主题的消息：

- finance/stock/ibm
- finance/stock/ibm/closingprice
- finance/stock/ibm/currentprice

多层次的通配符可以代表零个或多个级别。因此，finance/#也可以匹配单个的finance，其中#代表零级别。主题级别的分隔符在这种情况下毫无意义，因为没有级别来分隔。

多层次的通配符可以指定只对自己或下一个话题水平分隔字符。因此，#和finance/#都有效，但是，finance#是无效的。多层次的通配符必须在主题数的使用中作为最后一个字符。例如，finance/#是有效的，finance/#/closingprice是无效的。

Single-level wildcard

加号（+）是一个只匹配一个主题级别的通配符。例如，finance/stock/+匹配finance/stock/ibm和finance/stock/xyz，而不是finance/stock/ibm/closingprice。同时，由于单级通配符匹配只有一个层面，finance/+不匹配finance。

单级通配符可以在任何水平的主题树和在与多层次的通配符一起使用。它必须使用下一个话题水平分隔，当它自身指定的除外。因此，+和finance/+都有效，但finance+是无效的。单级通配符可以用于主题树结尾或是主题树内。例如，finance/+和finance/+/ibm都是有效的。

Topic semantics and usage

当你构建一个应用程序，主题树的设计应按一下原则考虑主题名称的语法和语义：

- 一个主题，必须至少有一个字符长。
- 主题名称是区分大小写的。例如，ACCOUNTS和Accounts是两个不同的主题。
- 主题名称可以包含空格字符，例如Accounts payable是一个合法的主题。
- A leading "/" creates a distinct topic. 例如，/finance 与finance不同。/finance匹配"+/+"和"/+"，而不是"+"。
- 不要在任何主题中包含空字符（Unicode x0000）。

以下原则适用于一个主题树的构建和内容：

- 长度被限制为64K，但是在这一个主题树内的级别数没有限制。
- 可以有任意数量的根节点，也就是说，可以有任意数量的主题树。

PS：

- 该篇文字由青樱桃团队翻译完成，可转载，但请标注青樱桃团队翻译（**www.girlcoding.com**）。
- 目录：**MQTT V3.1 Protocol Specification**目录
- 文中有翻译不对的地方，敬请赐教。

原创文章，转载请注明： 转载自**Girl is coding**