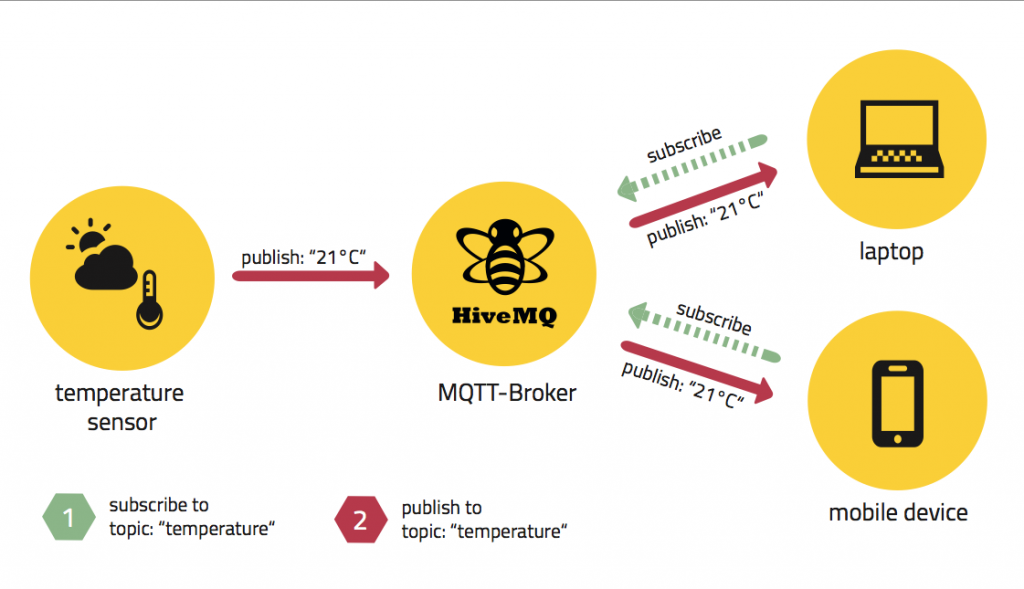
# 前言

## 诞生及优势

MQTT由Andy Stanford-Clark（IBM）和Arlen Nipper（Eurotech，现为Cirrus Link）于1999年开发，用于监测穿越沙漠的石油管道。目标是拥有一个****带宽有效且使用很少电池电量的协议****，因为这些设备是通过卫星链路连接的，当时这种设备非常昂贵。 ****与HTTP及其请求/响应范例相比，该协议使用发布/订阅体系结构。****发布/订阅是事件驱动的，可以将消息推送到客户端。中央通信点是MQTT代理，它负责调度发送者和合法接收者之间的所有消息。向代理发布消息的每个客户端都在消息中包含一个主题。****主题是代理的路由信息​。****每个想要接收消息的客户端都订阅某个主题，并且代理将具有匹配主题的所有消息传递给客户端。因此，客户不必彼此了解，他们只通过主题进行通信。该架构支持高度可扩展的解决方案，而不依赖于数据生产者和数据使用者。

## 发布/订阅架构

与HTTP的区别在于客户端不必提取所需的信息，但是在有新内容的情况下，代理会将信息推送到客户端。因此，每个MQTT客户端都与代理具有永久打开的TCP连接。如果此连接在任何情况下中断，MQTT代理可以缓冲所有消息，并在它重新联机时将它们发送到客户端。 如前所述，MQTT中用于分派消息的核心概念是主题。****主题是一个简单的字符串，可以有更多的层次结构级别，用斜杠分隔。****用于发送起居室的温度数据的示例主题可以是房屋/起居室/温度。一方面，客户端可以订阅确切的主题，或者另一方面使用通配符。对房屋/ + /温度的订阅将导致所有消息发送到先前提到的主题房屋/起居室/温度以及在起居室的地方具有任意值的任何主题，例如房屋/厨房/温度。加号是****单级通配符****，只允许一个层次结构的任意值。如果您需要订阅多个级别，例如订阅整个子树，还有一个****多级通配符****（＃）。它允许订阅所有底层层次结构级别。比如房子/＃订阅以house开头的所有主题。



## 适用人群

以下内容需要你对照着MQTT协议内容仔细推敲

推荐资源:

*MQTT协议中文版:*

*[https://mcxiaoke.gitbooks.io/mqtt-cn/content/mqtt/01-Introduction.html](https://mcxiaoke.gitbooks.io/mqtt-cn/content/mqtt/01-Introduction.html" \t "http://www.likecs.com/_blank)*

*MQTT Version 3.1.1:*

*[http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html](http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html" \t "http://www.likecs.com/_blank)*

# MQTT控制报文格式

## MQTT控制报文结构

| **结构** | **备注** |
| --- | --- |
| Fixed header | 固定报头，所有控制报文都包含 |
| Variable header | 可变报头，部分控制报文包含 |
| Payload | 有效载荷，部分控制报文包含 |

### 固定报头 Fixed header

+-----+-----+-----+-----+-----+------+------+------+-------+  
|     |     |     |     |     |     |     |     |       |  
| Bit | 7 | 6 | 5 | 4 | 3   | 2   | 1   | 0   |  
+-----------+-----+-----+------------+------+------+-------+  
|     |                       |                           |  
|byte1|MQTT ControlPacket type|         Flags             |  
+----------------------------------------------------------+  
|     |                                                   |  
|byte2|         Remaining Length                         |  
+----------------------------------------------------------+

#### 控制报文类型(MQTT Control Packet type)

| **名字** | **值** | **报文流动方向** | **描述** |
| --- | --- | --- | --- |
| Reserved | 0 | 禁止 | 保留 |
| CONNECT | 1 | 客户端到服务端 | 客户端请求连接服务端 |
| CONNACK | 2 | 服务端到客户端 | 连接报文确认 |
| PUBLISH | 3 | 两个方向都允许 | 发布消息 |
| PUBACK | 4 | 两个方向都允许 | QoS 1消息发布收到确认 |
| PUBREC | 5 | 两个方向都允许 | 发布收到（保证交付第一步） |
| PUBREL | 6 | 两个方向都允许 | 发布释放（保证交付第二步） |
| PUBCOMP | 7 | 两个方向都允许 | QoS 2消息发布完成（保证交互第三步） |
| SUBSCRIBE | 8 | 客户端到服务端 | 客户端订阅请求 |
| SUBACK | 9 | 服务端到客户端 | 订阅请求报文确认 |
| UNSUBSCRIBE | 10 | 客户端到服务端 | 客户端取消订阅请求 |
| UNSUBACK | 11 | 服务端到客户端 | 取消订阅报文确认 |
| PINGREQ | 12 | 客户端到服务端 | 心跳请求 |
| PINGRESP | 13 | 服务端到客户端 | 心跳响应 |
| DISCONNECT | 14 | 客户端到服务端 | 客户端断开连接 |
| Reserved | 15 | 禁止 | 保留 |

 标志(Flags)

| **控制报文** | **固定报头标志** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| --- | --- | --- | --- | --- | --- |
| CONNECT | Reserved | 0 | 0 | 0 | 0 |
| CONNACK | Reserved | 0 | 0 | 0 | 0 |
| PUBLISH | Used in MQTT 3.1.1 | DUP1 | QoS2 | QoS2 | RETAIN3 |
| PUBACK | Reserved | 0 | 0 | 0 | 0 |
| PUBREC | Reserved | 0 | 0 | 0 | 0 |
| PUBREL | Reserved | 0 | 0 | 1 | 0 |
| PUBCOMP | Reserved | 0 | 0 | 0 | 0 |
| SUBSCRIBE | Reserved | 0 | 0 | 1 | 0 |
| SUBACK | Reserved | 0 | 0 | 0 | 0 |
| UNSUBSCRIBE | Reserved | 0 | 0 | 1 | 0 |
| UNSUBACK | Reserved | 0 | 0 | 0 | 0 |
| PINGREQ | Reserved | 0 | 0 | 0 | 0 |
| PINGRESP | Reserved | 0 | 0 | 0 | 0 |
| DISCONNECT | Reserved | 0 | 0 | 0 | 0 |

| **Bit3** | **Bit2** | **Bit1** | **Bit0** |
| --- | --- | --- | --- |
| DUP | Qos | Qos | RETAIN |

DUP =控制报文的重复分发标志

QoS = PUBLISH报文的服务质量等级

RETAIN = PUBLISH报文的保留标志

****备注:****

****服务质量等级Qos****:****位置：****第1个字节，第2-1位。这个字段表示应用消息分发的服务质量等级保证。

| **QoS值** | **Bit 2** | **Bit 1** | **描述** |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 最多分发一次 |
| 1 | 0 | 1 | 至少分发一次 |
| 2 | 1 | 0 | 只分发一次 |
| - | 1 | 1 | 保留位 |

#### 剩余长度(Remaining Length)

****位置：****从第2个字节开始。

剩余长度（Remaining Length）表示当前报文剩余部分的字节数，包括可变报头和负载的数据。剩余长度不包括用于编码剩余长度字段本身的字节数。

### 可变报头(Variable header)

某些MQTT控制报文包含一个可变报头部分。它在固定报头和负载之间。可变报头的内容根据报文类型的不同而不同。可变报头的报文标识符（Packet Identifier）字段存在于在多个类型的报文里。这个在后续的MQTT各个控制报文中进行****手撕****。

### 有效载荷(Payload)

某些MQTT控制报文在报文的最后部分包含一个有效载荷，对于PUBLISH来说有效载荷就是应用消息。

****包含有效载荷的控制报文Control Packets that contain a Payload****

| **控制报文** | **有效载荷** |
| --- | --- |
| CONNECT | 需要 |
| CONNACK | 不需要 |
| PUBLISH | 可选 |
| PUBACK | 不需要 |
| PUBREC | 不需要 |
| PUBREL | 不需要 |
| PUBCOMP | 不需要 |
| SUBSCRIBE | 需要 |
| SUBACK | 需要 |
| UNSUBSCRIBE | 需要 |
| UNSUBACK | 不需要 |
| PINGREQ | 不需要 |
| PINGRESP | 不需要 |
| DISCONNECT | 不需要 |

# MQTT控制报文

## CONNECT – 连接服务端

客户端到服务端的网络连接建立后，客户端发送给服务端的第一个报文****必须****是CONNECT报文 。

在一个网络连接上，客户端只能发送一次CONNECT报文。服务端****必须****将客户端发送的第二个CONNECT报文当作协议违规处理并断开客户端的连接。

有效载荷包含一个或多个编码的字段。包括客户端的唯一标识符，Will主题，Will消息，用户名和密码。除了客户端标识之外，其它的字段都是可选的，基于标志位来决定可变报头中是否需要包含这些字段。

### 可变报头

CONNECT报文的可变报头按下列次序包含四个字段：协议名（Protocol Name），协议级别（Protocol Level），连接标志（Connect Flags）和保持连接（Keep Alive）。

#### 协议名（Protocol Name）

协议名是表示协议名 MQTT 的UTF-8编码的字符串。MQTT规范的后续版本不会改变这个字符串的偏移和长度。

如果协议名不正确服务端****可以****断开客户端的连接，也****可以****按照某些其它规范继续处理CONNECT报文。对于后一种情况，按照本规范，服务端****不能****继续处理CONNECT报文

#### 协议级别（Protocol Level）

客户端用8位的无符号值表示协议的修订版本。对于3.1.1版协议，协议级别字段的值是4(0x04)。

如果发现不支持的协议级别，服务端****必须****给发送一个返回码为0x01（不支持的协议级别）的CONNACK报文响应CONNECT报文，然后断开客户端的连接 。

#### 连接标志（Connect Flags）

连接标志字节包含一些用于指定MQTT连接行为的参数。它还指出有效载荷中的字段是否存在

+-------+---------+----------+--------+------+-------+-------+--------+--------+  
|       |         |         |       |     |       |       |       |       |  
| Bit |   7   |   6     |   5   | 4   | 3   |   2   |   1   |   0   |  
+--------------------------------------------+---------------------------------+  
|       |User Name|Password | Will   |             | Will | Clean |       |  
|       | Flag   | Flag     | Retain | Will Qos   | Flag |Session |Reser^ed|  
+--------------------------------------------+---------------------------------+  
|       |         |         |       |     |       |       |       |       |  
| byte8 |   X   |   X     |   X   |   X |   X   |   X   |   X   |   0   |  
+-------+---------+----------+--------+------+-------+-------+--------+--------+

服务端****必须****验证CONNECT控制报文的保留标志位（第0位）是否为0，如果不为0必须断开客户端连接 。

##### 清理会话 Clean Session

****位置：****连接标志字节的第1位

这个二进制位指定了会话状态的处理方式。

客户端和服务端可以保存会话状态，以支持跨网络连接的可靠消息传输。这个标志位用于控制会话状态的生存时间。

##### 遗嘱标志 Will Flag

****位置：****连接标志的第2位。

遗嘱标志（Will Flag）被设置为1，表示如果连接请求被接受了，遗嘱（Will Message）消息****必须****被存储在服务端并且与这个网络连接关联。之后网络连接关闭时，服务端****必须****发布这个遗嘱消息，除非服务端收到DISCONNECT报文时删除了这个遗嘱消息

##### 遗嘱QoS Will QoS

****位置：****连接标志的第4和第3位。

这两位用于指定发布遗嘱消息时使用的服务质量等级。

##### 遗嘱保留 Will Retain

****位置：****连接标志的第5位。

如果遗嘱消息被发布时需要保留，需要指定这一位的值。

##### 用户名标志 User Name Flag

****位置：****连接标志的第7位。

如果用户名（User Name）标志被设置为0，有效载荷中****不能****包含用户名字段 。

如果用户名（User Name）标志被设置为1，有效载荷中****必须****包含用户名字段 。

##### 密码标志 Password Flag

****位置：****连接标志的第6位。

如果密码（Password）标志被设置为0，有效载荷中****不能****包含密码字段 。

如果密码（Password）标志被设置为1，有效载荷中****必须****包含密码字段 。

如果用户名标志被设置为0，密码标志也****必须****设置为0 。

#### 保持连接 Keep Alive

保持连接（Keep Alive）是一个以秒为单位的时间间隔，表示为一个16位的字，它是指在客户端传输完成一个控制报文的时刻到发送下一个报文的时刻，两者之间允许空闲的最大时间间隔。

客户端负责保证控制报文发送的时间间隔不超过保持连接的值。如果没有任何其它的控制报文可以发送，客户端****必须****发送一个PINGREQ报文 。

### 有效载荷

CONNECT报文的有效载荷（payload）包含一个或多个以长度为前缀的字段，可变报头中的标志决定是否包含这些字段。如果包含的话，****必须****按这个顺序出现：客户端标识符，遗嘱主题，遗嘱消息，用户名，密码 。

### 响应Response

网络连接建立后，如果服务端在合理的时间内没有收到CONNECT报文，服务端****应该****关闭这个连接。

服务端****必须****按照3.1节的要求验证CONNECT报文，如果报文不符合规范，服务端不发送CONNACK报文直接关闭网络连接 [MQTT-3.1.4-1]。

服务端****可以****检查CONNECT报文的内容是不是满足任何进一步的限制，****可以****执行身份验证和授权检查。如果任何一项检查没通过，按照3.2节的描述，它****应该****发送一个适当的、返回码非零的CONNACK响应，并且****必须****关闭这个网络连接。

## CONNACK – 确认连接请求

服务端发送CONNACK报文响应从客户端收到的CONNECT报文。服务端发送给客户端的第一个报文****必须****是CONNACK。

如果客户端在合理的时间内没有收到服务端的CONNACK报文，客户端****应该****关闭网络连接。合理 的时间取决于应用的类型和通信基础设施。

****剩余长度字段****

表示可变报头的长度。对于CONNACK报文这个值等于2。

### 可变报头

#### 连接确认标志 Connect Acknowledge Flags

第1个字节是 连接确认标志，位7-1是保留位且****必须****设置为0。 第0 (SP)位 是当前会话（Session Present）标志。

##### 当前会话 Session Present

****位置：****连接确认标志的第0位。

#### 连接返回码 Connect Return code

****位置：****可变报头的第2个字节。

连接返回码字段使用一个字节的无符号值。如果服务端收到一个合法的CONNECT报文，但出于某些原因无法处理它，服务端应该尝试发送一个包含非零返回码（表格中的某一个）的CONNACK报文。如果服务端发送了一个包含非零返回码的CONNACK报文，那么它****必须****关闭网络连接 。

##### 表格 3.1 –连接返回码的值

| **值** | **返回码响应** | **描述** |
| --- | --- | --- |
| 0 | 0x00连接已接受 | 连接已被服务端接受 |
| 1 | 0x01连接已拒绝，不支持的协议版本 | 服务端不支持客户端请求的MQTT协议级别 |
| 2 | 0x02连接已拒绝，不合格的客户端标识符 | 客户端标识符是正确的UTF-8编码，但服务端不允许使用 |
| 3 | 0x03连接已拒绝，服务端不可用 | 网络连接已建立，但MQTT服务不可用 |
| 4 | 0x04连接已拒绝，无效的用户名或密码 | 用户名或密码的数据格式无效 |
| 5 | 0x05连接已拒绝，未授权 | 客户端未被授权连接到此服务器 |
| 6-255 |  | 保留 |

### 有效载荷

CONNACK报文没有有效载荷。

## PUBLISH – 发布消息

PUBLISH控制报文是指从客户端向服务端或者服务端向客户端传输一个应用消息。

### 固定报头

DUP =控制报文的重复分发标志

QoS = PUBLISH报文的服务质量等级

RETAIN = PUBLISH报文的保留标志

### 可变报头

可变报头按顺序包含主题名和报文标识符。

#### 主题名 Topic Name

主题名（Topic Name）用于识别有效载荷数据应该被发布到哪一个信息通道。

主题名****必须****是PUBLISH报文可变报头的第一个字段。它****必须****是 1.5.3节定义的UTF-8编码的字符串。

PUBLISH报文中的主题名****不能****包含通配符 。

服务端发送给订阅客户端的PUBLISH报文的主题名****必须****匹配该订阅的主题过滤器（根据 4.7节定义的匹配过程）。

#### 报文标识符 Packet Identifier

只有当QoS等级是1或2时，报文标识符（Packet Identifier）字段才能出现在PUBLISH报文中。2.3.1节提供了有关报文标识符的更多信息。

### 有效载荷

有效载荷包含将被发布的应用消息。

数据的内容和格式是应用特定的。有效载荷的长度这样计算：用固定报头中的剩余长度字段的值减去可变报头的长度。包含零长度有效载荷的PUBLISH报文是合法的。

### 响应

PUBLISH报文的接收者****必须****按照根据PUBLISH报文中的QoS等级发送响应。

| **服务质量等级** | **预期响应** |
| --- | --- |
| QoS 0 | 无响应 |
| QoS 1 | PUBACK报文 |
| QoS 2 | PUBREC报文 |

## PUBACK –发布确认

PUBACK报文是对QoS 1等级的PUBLISH报文的响应。

****剩余长度字段****

表示可变报头的长度。对PUBACK报文这个值等于2.

### 可变报头

包含等待确认的PUBLISH报文的报文标识符。

### 有效载荷

PUBACK报文没有有效载荷。

## PUBREC – 发布收到（QoS 2，第一步）

PUBREC报文是对QoS等级2的PUBLISH报文的响应。它是QoS 2等级协议交换的第二个报文。

****剩余长度字段****

表示可变报头的长度。对PUBREC报文它的值等于2。

### 可变报头

可变报头包含等待确认的PUBLISH报文的报文标识符。

### 有效载荷

PUBREC报文没有有效载荷。

### Netty实现

MqttMessage pubRecMessage = MqttMessageFactory.newMessage(

new MqttFixedHeader(MqttMessageType.PUBREC, false, MqttQoS.AT\_MOST\_ONCE, false, 2),

MqttMessageIdVariableHeader.from(variableHeader.messageId()),

null);

## PUBREL – 发布释放（QoS 2，第二步）

PUBREL报文是对PUBREC报文的响应。它是QoS 2等级协议交换的第三个报文。

PUBREL控制报文固定报头的第3,2,1,0位是保留位，****必须****被设置为0,0,1,0。服务端****必须****将其它的任何值都当做是不合法的并关闭网络连接 。

****剩余长度字段****

表示可变报头的长度。对PUBREL报文这个值等于2.

### 可变报头

可变报头包含与等待确认的PUBREC报文相同的报文标识符。

### 有效载荷

PUBREL报文没有有效载荷。

### Netty实现

MqttMessage pubRelMessage = MqttMessageFactory.newMessage(

new MqttFixedHeader(MqttMessageType.PUBREC, false, MqttQoS.AT\_LEAST\_ONCE, false, 2),

MqttMessageIdVariableHeader.from(variableHeader.messageId()),

null);

## PUBCOMP – 发布完成（QoS 2，第三步）

PUBCOMP报文是对PUBREL报文的响应。它是QoS 2等级协议交换的第四个也是最后一个报文。

****剩余长度字段****

表示可变报头的长度。对PUBCOMP报文这个值等于2。

### 可变报头

可变报头包含与等待确认的PUBREL报文相同的报文标识符。

### 有效载荷

PUBCOMP报文没有有效载荷。

### Netty实现

MqttMessage pubCompMessage = MqttMessageFactory.newMessage(

new MqttFixedHeader(MqttMessageType.PUBCOMP, false, MqttQoS.AT\_MOST\_ONCE, false, 2),

MqttMessageIdVariableHeader.from(variableHeader.messageId()),

null);

## SUBSCRIBE - 订阅主题

客户端向服务端发送SUBSCRIBE报文用于创建一个或多个订阅。每个订阅注册客户端关心的一个或多个主题。为了将应用消息转发给与那些订阅匹配的主题，服务端发送PUBLISH报文给客户端。SUBSCRIBE报文也（为每个订阅）指定了最大的QoS等级，服务端根据这个发送应用消息给客户端。

****剩余长度字段****

等于可变报头的长度（2字节）加上有效载荷的长度。

### 可变报头

可变报头包含报文标识符。

### 有效载荷

SUBSCRIBE报文的有效载荷包含了一个主题过滤器列表，它们表示客户端想要订阅的主题。

SUBSCRIBE报文有效载荷中的主题过滤器列表****必须****是1.5.3节定义的UTF-8字符串 [MQTT-3.8.3-1]。

服务端****应该****支持包含通配符（4.7.1节定义的）的主题过滤器。如果服务端选择不支持包含通配符的主题过滤器，****必须****拒绝任何包含通配符过滤器的订阅请求 [MQTT-3.8.3-2]。

每一个过滤器后面跟着一个字节，这个字节被叫做 服务质量要求（Requested QoS）。它给出了服务端向客户端发送应用消息所允许的最大QoS等级。

SUBSCRIBE报文的有效载荷****必须****包含至少一对主题过滤器 和 QoS等级字段组合。没有有效载荷的SUBSCRIBE报文是违反协议的 [MQTT-3.8.3-3]。有关错误处理的信息请查看4.8节。

请求的最大服务质量等级字段编码为一个字节，它后面跟着UTF-8编码的主题名，那些主题过滤器 /和QoS等级组合是连续地打包。



当前版本的协议没有用到服务质量要求（Requested QoS）字节的高六位。如果有效载荷中的任何位是非零值，或者QoS不等于0,1或2，服务端****必须****认为SUBSCRIBE报文是不合法的并关闭网络连接 。

## SUBACK – 订阅确认

服务端发送SUBACK报文给客户端，用于确认它已收到并且正在处理SUBSCRIBE报文。

SUBACK报文包含一个返回码清单，它们指定了SUBSCRIBE请求的每个订阅被授予的最大QoS等级。

****剩余长度字段****

等于可变报头的长度加上有效载荷的长度。

### 可变报头

可变报头包含等待确认的SUBSCRIBE报文的报文标识符。

### 有效载荷

有效载荷包含一个返回码清单。每个返回码对应等待确认的SUBSCRIBE报文中的一个主题过滤器。返回码的顺序****必须****和SUBSCRIBE报文中主题过滤器的顺序相同 。

| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 返回码 |  |  |  |  |  |  |  |
| byte 1 | X | 0 | 0 | 0 | 0 | 0 | X | X |

允许的返回码值：

0x00 - 最大QoS 0

0x01 - 成功 – 最大QoS 1

0x02 - 成功 – 最大 QoS 2

0x80 - Failure 失败

0x00, 0x01, 0x02, 0x80之外的SUBACK返回码是保留的，****不能****使用。

## UNSUBSCRIBE –取消订阅

客户端发送UNSUBSCRIBE报文给服务端，用于取消订阅主题。

UNSUBSCRIBE报文固定报头的第3,2,1,0位是保留位且****必须****分别设置为0,0,1,0。

即:

DUP =控制报文的重复分发标志 : ****false****

QoS = PUBLISH报文的服务质量等级: ****Qos1****(至少分发一次)

RETAIN = PUBLISH报文的保留标志: ****false****

服务端****必须****认为任何其它的值都是不合法的并关闭网络连接 。

****剩余长度字段****

等于可变报头的长度加上有效载荷的长度。

### 可变报头

### 有效载荷

UNSUBSCRIBE报文的有效载荷包含客户端想要取消订阅的主题过滤器列表。UNSUBSCRIBE报文中的主题过滤器****必须****是连续打包的。

UNSUBSCRIBE报文的有效载荷****必须****至少包含一个消息过滤器。没有有效载荷的UNSUBSCRIBE报文是违反协议的。

### 响应

UNSUBSCRIBE报文提供的主题过滤器（无论是否包含通配符）****必须****与服务端持有的这个客户端的当前主题过滤器集合逐个字符比较。如果有任何过滤器完全匹配，那么它（服务端）自己的订阅将被删除，否则不会有进一步的处理 。

如果服务端删除了一个订阅：

它****必须****停止分发任何新消息给这个客户端 。

它****必须****完成分发任何已经开始往客户端发送的QoS 1和QoS 2的消息。

它****可以****继续发送任何现存的准备分发给客户端的缓存消息。

服务端****必须****发送UNSUBACK报文响应客户端的UNSUBSCRIBE请求。UNSUBACK报文****必须****包含和UNSUBSCRIBE报文相同的报文标识符 。即使没有删除任何主题订阅，服务端也****必须****发送一个UNSUBACK响应 。

如果服务端收到包含多个主题过滤器的UNSUBSCRIBE报文，它****必须****如同收到了一系列的多个UNSUBSCRIBE报文一样处理那个报文，除了将它们的响应合并到一个单独的UNSUBACK报文外。

## UNSUBACK – 取消订阅确认

服务端发送UNSUBACK报文给客户端用于确认收到UNSUBSCRIBE报文。

****剩余长度字段****

表示可变报头的长度，对UNSUBACK报文这个值等于2。

### 可变报头

可变报头包含等待确认的UNSUBSCRIBE报文的报文标识符。

### 有效载荷

UNSUBACK报文没有有效载荷。

### Netty实现

MqttUnsubAckMessage unsubAckMessage = (MqttUnsubAckMessage)MqttMessageFactory.newMessage(

new MqttFixedHeader(MqttMessageType.UNSUBACK, false, MqttQoS.AT\_MOST\_ONCE, false, 2),

MqttMessageIdVariableHeader.from(msg.variableHeader().messageId()),

null);

## PINGREQ – 心跳请求

客户端发送PINGREQ报文给服务端的。用于：

在没有任何其它控制报文从客户端发给服务的时，告知服务端客户端还活着。

请求服务端发送 响应确认它还活着。

使用网络以确认网络连接没有断开。

保持连接（Keep Alive）处理中用到这个报文。

### 可变报头

PINGREQ报文没有可变报头。

### 有效载荷

PINGREQ报文没有有效载荷。

### 响应

服务端****必须****发送 PINGRESP报文响应客户端的****PINGREQ报文****

### Netty实现

[IMG_274](http://www.likecs.com/javascript:void(0);)

MqttMessage pingReqMessage = MqttMessageFactory.newMessage(

new MqttFixedHeader(MqttMessageType.PINGREQ, false,MqttQoS.AT\_MOST\_ONCE, false, 0),

null,

null);

## PINGRESP – 心跳响应

服务端发送PINGRESP报文响应客户端的PINGREQ报文。表示服务端还活着。保持连接（Keep Alive）处理中用到这个报文。

### 可变报头

PINGRESP报文没有可变报头。

### 有效载荷

PINGRESP报文没有有效载荷。

### Netty实现

MqttMessage pingRespMessage = MqttMessageFactory.newMessage(

new MqttFixedHeader(MqttMessageType.PINGREQ, false

,MqttQoS.AT\_MOST\_ONCE, false, 0),

null,

null);

## DISCONNECT –断开连接

DISCONNECT报文是客户端发给服务端的最后一个控制报文。表示客户端正常断开连接。

### 可变报头

DISCONNECT报文没有可变报头。

### 有效载荷

DISCONNECT报文没有有效载荷。

### 响应

客户端发送DISCONNECT报文之后：

****必须****关闭网络连接 。

****不能****通过那个网络连接再发送任何控制报文 。

服务端在收到DISCONNECT报文时：

****必须****丢弃任何与当前连接关联的未发布的遗嘱消息

****应该****关闭网络连接，如果客户端 还没有这么做。

### Netty实现

MqttMessage disConnectMessage = MqttMessageFactory.newMessage(

new MqttFixedHeader(MqttMessageType.DISCONNECT,

false,MqttQoS.AT\_MOST\_ONCE, false,0),

null,

null);