MQTT 协议 3.1.1 中文版

OASIS 标准

2014 年 10 月 29 日

规范链接

当前版本：

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.doc (Authoritative)
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.pdf

以前的版本：

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/cos01/mqtt-v3.1.1-cos01.doc (Authoritative)
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/cos01/mqtt-v3.1.1-cos01.html
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/cos01/mqtt-v3.1.1-cos01.pdf

最新版本：

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.doc (Authoritative)
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf

技术委员会：

结构化信息标准促进组织 MQTT 技术委员会

主席：

拉斐尔・J・科恩 (raphael.cohn@stormmq.com), 个人
理查德・J・科彭 (coppen@uk.ibm.com), IBM

编辑：

安德鲁・班克斯 (Andrew_Banks@uk.ibm.com), IBM
拉胡尔・吉普塔 (rahul.gupta@us.ibm.com), IBM

相关文档：

本规范与此有关：

- MQTT 和 NIST 网络安全框架 1.0 版。 编辑是杰夫・布朗和路易・菲利普・拉穆勒。最新版本： http://docs.oasis-open.org/mqtt/mqtt-nist-cybersecurity/v1.0/mqtt-nist-cybersecurity-v1.0.html.

摘要：

MQTT 是一个客户端服务端架构的发布/订阅模式的消息传输协议。它的设计思想是轻巧、开放、简单、规范，因此易于实现。这些特点使得它对很多场景来说都是很好的选择，包括受限的环境如机器与机器的通信（M2M）以及物联网环境（IoT），这些场景要求很小的代码封装或者网络带宽非常昂贵。

本协议运行在 TCP/IP，或其它提供了有序、可靠、双向连接的网络连接上。它有以下特点：

- 使用发布/订阅消息模式，提供了一对多的消息分发和应用之间的解耦。
- 消息传输不需要知道负载内容。
- 提供三种等级的服务质量：.

- “最多一次”，尽操作环境所能提供的最大努力分发消息。消息可能会丢失。例如，这个等级可用于环境传感器数据，单次的数据丢失没关系，因为不久之后会再次发送。
- “至少一次”，保证消息可以到达，但是可能会重复。
- “仅一次”，保证消息只到达一次。例如，这个等级可用在一个计费系统中，这里如果消息重复或丢失会导致不正确的收费。

- 很小的传输消耗和协议数据交换，最大限度减少网络流量
- 异常连接断开发生时，能通知到相关各方。

状态：

本文档最后由 OASIS 成员在上面标示的日期最终修订或批准。批准的级别也在上面列出了。如果要查看本文档最新的修订版请检查上面的 *最新版本* 位置。技术委员会产生的其它修订版和其它技术文档都列在这里：https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt#technical 。

技术委员会成员对本规范的评论应该发送到技术委员会的邮件列表。其他人应该发送评论到技术委员会的公共评论列表，方法是点击技术委员会网站的 发送评论 按钮，网页地址是 https://www.oasis-open.org/committees/mqtt/ 。

关于实现本规范必不可少的任何专利是否已公开，以及其它的专利许可条款相关的信息，请参考技术委员会网站的知识产权部分（(https://www.oasis-open.org/committees/mqtt/ipr.php）。

引用格式：

引用此规范时应该使用下面的引文格式：

[mqtt-v3.1.1]

MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. 29 October 2014. OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

文档链接

- MQTT 协议 3.1.1 中文翻译项目
- MQTT 协议 3.1.1 中文版 PDF

修订记录

版 本	日 期	发布说明
1.0.0	2015-07-30	翻译全部文本，完成初步审校，公开发布第一版
1.0.1	2015-10-22	修订几处笔误，增补几处未翻译的文本

关于译者

- GitHub
- Blog
- Email

目录

1 概述

1.1 MQTT 协议的组织结构

本规范分为七个章节：

- 第一章 - 介绍
- 第二章 – MQTT 控制报文格式
- 第三章 – MQTT 控制报文
- 第四章 – 操作行为
- 第五章 – 安全
- 第六章 – 使用 WebSocket 作为网络传输层 t
- 第七章 – 一致性目标

1.2 术语

本规范中用到的关键字 **必须** MUST，**不能** MUST NOT，**要求** REQUIRED，**将会** SHALL，**不会** SHALL NOT，**应该** SHOULD，**不应该** SHOULD NOT，**推荐** RECOMMENDED，**可以** MAY，**可选** OPTIONAL 都是按照 IETF RFC 2119 [RFC2119] 中的描述解释。

网络连接（Network Connection）：

MQTT 使用的底层传输协议基础设施。

- 客户端使用它连接服务端。
- 它提供有序的、可靠的、双向字节流传输。

例子见 4.2 节。

应用消息（Application Message）：

MQTT 协议通过网络传输应用数据。应用消息通过 MQTT 传输时，它们有关联的服务质量（QoS）和主题（Topic）。

客户端（Client）：

使用 MQTT 的程序或设备。客户端总是通过网络连接到服务端。它可以

- 发布应用消息给其它相关的客户端。.
- 订阅以请求接受相关的应用消息
- 取消订阅以移除接受应用消息的请求。
- 从服务端断开连接。

服务端（Server）：

一个程序或设备，作为发送消息的客户端和请求订阅的客户端之间的中介。服务端

- 接受来自客户端的网络连接
- 接受客户端发布的应用消息

- 处理客户端的订阅和取消订阅请求。
- 转发应用消息给符合条件的客户端订阅。

订阅（Subscription）：
订阅包含一个主题过滤器（Topic Filter）和一个最大的服务质量（QoS）等级。订阅与单个会话（Session）关联。会话可以包含多于一个的订阅。会话的每个订阅都有一个不同的主题过滤器。

主题名（Topic Name）：
附加在应用消息上的一个标签，服务端已知且与订阅匹配。服务端发送应用消息的一个副本给每一个匹配的客户端订阅。

主题过滤器（Topic Filter:）：
订阅中包含的一个表达式，用于表示相关的一个或多个主题。主题过滤器可以使用通配符。

会话（Session）：
客户端和服务端之间的状态交互。一些会话持续时长与网络连接一样，另一些可以在客户端和服务端的多个连续网络连接间扩展。

控制报文（MQTT Control Packet）：
通过网络连接发送的信息数据包。MQTT 规范定义了十四种不同类型的控制报文，其中一个（PUBLISH 报文）用于传输应用消息。

1.3 规范引用

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
http://www.ietf.org/rfc/rfc2119.txt

[RFC3629]

Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003
http://www.ietf.org/rfc/rfc3629.txt

[RFC5246]

Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

http://www.ietf.org/rfc/rfc5246.txt

[RFC6455]

Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

http://www.ietf.org/rfc/rfc6455.txt

[Unicode]

The Unicode Consortium. The Unicode Standard.

http://www.unicode.org/versions/latest/

1.4 非规范引用

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

[AES]

Advanced Encryption Standard (AES) (FIPS PUB 197).
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[DES]

Data Encryption Standard (DES).
http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[FIPS1402]

Security Requirements for Cryptographic Modules (FIPS PUB 140-2)

http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[IEEE 802.1AR]

IEEE Standard for Local and metropolitan area networks - Secure Device Identity
http://standards.ieee.org/findstds/standard/802.1AR-2009.html

[ISO29192]

ISO/IEC 29192-1:2012 Information technology -- Security techniques -- Lightweight cryptography -- Part 1: General
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56425

[MQTT NIST]

MQTT supplemental publication, MQTT and the NIST Framework for Improving Critical Infrastructure Cybersecurity
http://docs.oasis-open.org/mqtt/mqtt-nist-cybersecurity/v1.0/mqtt-nist-cybersecurity-v1.0.html

[MQTTV31]

MQTT V3.1 Protocol Specification.

http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html

[NISTCSF]

Improving Critical Infrastructure Cybersecurity Executive Order 13636
http://www.nist.gov/itl/upload/preliminary-cybersecurity-framework.pdf

[NIST7628]

NISTIR 7628 Guidelines for Smart Grid Cyber Security
http://www.nist.gov/smartgrid/upload/nistir-7628_total.pdf

[NSAB]

NSA Suite B Cryptography
http://www.nsa.gov/ia/programs/suiteb_cryptography/

[PCIDSS]

PCI-DSS Payment Card Industry Data Security Standard
https://www.pcisecuritystandards.org/security_standards/

[RFC1928]

Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, March 1996.

http://www.ietf.org/rfc/rfc1928.txt

[RFC4511]

Sermersheim, J., Ed., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, June 2006.

http://www.ietf.org/rfc/rfc4511.txt

[RFC5077]

Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

http://www.ietf.org/rfc/rfc5077.txt

[RFC5280]

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

http://www.ietf.org/rfc/rfc5280.txt

[RFC6066]

Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

http://www.ietf.org/rfc/rfc6066.txt

[RFC6749]

Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

http://www.ietf.org/rfc/rfc6749.txt

[RFC6960]

Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, June 2013.
http://www.ietf.org/rfc/rfc6960.txt

[SARBANES]

Sarbanes-Oxley Act of 2002.
http://www.gpo.gov/fdsys/pkg/PLAW-107publ204/html/PLAW-107publ204.htm

[USEUSAFEHARB]

U.S.-EU Safe Harbor
http://export.gov/safeharbor/eu/eg_main_018365.asp

1.5 数据表示

1.5.1 二进制位

字节中的位从 0 到 7。第 7 位是最高有效位，第 0 位是最低有效位。

1.5.2 整数数值

整数数值是 16 位，使用大端序（big-endian，高位字节在低位字节前面）。这意味着一个 16 位的字在网络上表示为最高有效字节（MSB），后面跟着最低有效字节（LSB）。

1.5.3 UTF-8 编码字符串

后面会描述的控制报文中的文本字段编码为 UTF-8 格式的字符串。UTF-8 [RFC3629] 是一个高效的 Unicode 字符编码格式，为了支持基于文本的通信，它对 ASCII 字符的编码做了优化。

每一个字符串都有一个两字节的长度字段作为前缀，它给出这个字符串 UTF-8 编码的字节数，它们在图例 1.1 UTF-8 编码字符串的结构 中描述。因此可以传送的 UTF-8 编码的字符串大小有一个限制，不能超过 65535 字节。

除非另有说明，所有的 UTF-8 编码字符串的长度都必须在 0 到 65535 字节这个范围内。

图例 1.1 UTF-8 编码字符串的结构

二进制位	7	6	5	4	3	2	1	0
byte 1	字符串长度的最高有效字节（MSB）							
byte 2	字符串长度的最低有效字节（LSB）							
byte 3 ….	如果长度大于 0，这里是 UTF-8 编码的字符数据。							

UTF-8 编码字符串中的字符数据**必须**是按照 Unicode 规范 [Unicode] 定义的和在 RFC3629 [RFC3629] 中重申的有效的 UTF-8 格式。特别需要指出的是，这些数据**不能**包含字符码在 U+D800 和 U+DFFF 之间的数据。如果服务端或客户端收到了一个包含无效 UTF-8 字符的控制报文，它**必须**关闭网络连接 [MQTT-1.5.3-1]。

UTF-8 编码的字符串**不能**包含空字符 U+0000。如果客户端或服务端收到了一个包含 U+0000 的控制报文，它**必须**关闭网络连接 [MQTT-1.5.3-2]。

数据中**不应该**包含下面这些 Unicode 代码点的编码。如果一个接收者（服务端或客户端）收到了包含下列任意字符的控制报文，它**可以**关闭网络连接：

U+0001 和 U+001F 之间的控制字符

U+007F 和 U+009F 之间的控制字符

Unicode 规范定义的非字符代码点（例如 U+0FFFF）

Unicode 规范定义的保留字符（例如 U+0FFFF）

UTF-8 编码序列 0XEF 0xBB 0xBF 总是被解释为 U+FEFF（零宽度非换行空白字符），无论它出现在字符串的什么位置，报文接收者都不能跳过或者剥离它 [MQTT-1.5.3-3]。

1.5.3.1 非规范示例

例如，字符串 A 𠮷 是一个拉丁字母 A 后面跟着一个代码点 U+2A6D4(它表示一个中日韩统一表意文字扩展 B 中的字符)，这个字符串编码如下：

图例 1.2 UTF-8 编码字符串非规范示例

Bit	7	6	5	4	3	2	1	0
byte 1	字符串长度 MSB (0x00)							
	0	0	0	0	0	0	0	0
byte 2	字符串长度 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

1.6 编辑约定

本规范用黄色高亮的文本标识一致性声明，每个一致性声明都分配了一个这种格式的引用：[MQTT-x.x.x-y]。

2 MQTT 控制报文格式

2.1 MQTT 控制报文的结构

MQTT 协议通过交换预定义的 MQTT 控制报文来通信。这一节描述这些报文的格式。

MQTT 控制报文由三部分组成，按照 图例 2.1 –MQTT 控制报文的结构 描述的顺序：

图例 2.1 –MQTT 控制报文的结构

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

2.2 固定报头

每个 MQTT 控制报文都包含一个固定报头。图例 2.2 -固定报头的格式 描述了固定报头的格式。

图例 2.2 -固定报头的格式

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文的类型				用于指定控制报文类型的标志位			
byte 2…	剩余长度							

2.2.1 MQTT 控制报文的类型

位置：第 1 个字节，二进制位 7-4。

表示为 4 位无符号值，这些值的定义见 表格 2.1 -控制报文的类型

表格 2.1 -控制报文的类型

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

2.2.2 标志

固定报头第 1 个字节的剩余的 4 位 [3-0]包含每个 MQTT 控制报文类型特定的标志，见 表格 2.2 -标志位。表格 2.2 中任何标记为“保留”的标志位，都是保留给以后使用的，**必须**设置为表格中列出的值 [MQTT-2.2.2-1]。如果收到非法的标志，接收者**必须**关闭网络连接。有关错误处理的详细信息见 4.8 节 [MQTT-2.2.2-2]。

表格 2.2 -标志位

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	DUP[1]	QoS[2]	QoS[2]	RETAIN[3]
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0

DUP[1] =控制报文的重复分发标志

QoS[2] = PUBLISH 报文的服务质量等级

RETAIN[3] = PUBLISH 报文的保留标志

PUBLISH 控制报文中的 DUP, QoS 和 RETAIN 标志的描述见 3.3.1 节。

2.2.3 剩余长度

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。

剩余长度字段使用一个变长度编码方案，对小于 128 的值它使用单字节编码。更大的值按下面的方式处理。低 7 位有效位用于编码数据，最高有效位用于指示是否有更多的字节。因此每个字节可以编码 128 个数值和一个*延续位（continuation bit）*。剩余长度字段最大 4 个字节。

非规范评注

例如，十进制数 64 会被编码为一个字节，数值是 64，十六进制表示为 0x40,。十进制数字 321(=65+2*128)被编码为两个字节，最低有效位在前。第一个字节是 65+128=193。注意最高位为 1 表示后面至少还有一个字节。第二个字节是 2。

非规范评注

这允许应用发送最大 256MB(268,435,455)大小的控制报文。这个数值在报文中的表示是：0xFF,0xFF,0xFF,0x7F。

表格 2.4 剩余长度字段的大小展示了剩余长度字段所表示的值随字节增长。

表格 2.4 剩余长度字段的大小

字节数	最小值	最大值
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)

分别表示（每个字节的低 7 位用于编码数据，最高位是标志位）：

1 个字节时，从 0(0x00)到 127(0x7f)

2 个字节时，从 128(0x80,0x01)到 16383(0Xff,0x7f)

3 个字节时，从 16384(0x80,0x80,0x01)到 2097151(0xFF,0xFF,0x7F)

4 个字节时，从 2097152(0x80,0x80,0x80,0x01)到 268435455(0xFF,0xFF,0xFF,0x7F)

非规范评注

非负整数 X 使用变长编码方案的算法如下：

```
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 )
```

MOD 是模运算，DIV 是整数除法，OR 是位操作或（C 语言中分别是%，/，|）

非规范评注

剩余长度字段的解码算法如下：

```
multiplier = 1
value = 0
do
    encodedByte = 'next byte from stream'
    value += (encodedByte AND 127) * multiplier
    multiplier *= 128
    if (multiplier > 128*128*128)
      throw Error(Malformed Remaining Length)
while ((encodedByte AND 128) != 0)
```

AND 是位操作与（C 语言中的&）

这个算法终止时，value 包含的就是剩余长度的值。

2.3 可变报头

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。

2.3.1 报文标识符

图例 2.3 -报文标识符类型

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

很多控制报文的可变报头部分包含一个两字节的报文标识符字段。这些报文是 PUBLISH（QoS>0 时），PUBACK，PUBREC，PUBREL，PUBCOMP，SUBSCRIBE, SUBACK，UNSUBSCIBE，UNSUBACK。

SUBSCRIBE，UNSUBSCRIBE 和 PUBLISH（QoS 大于 0）控制报文**必须**包含一个非零的 16 位报文标识符（Packet Identifier）[MQTT-2.3.1-1]。客户端每次发送一个新的这些类型的报文时都**必须**分配一个当前未使用的报文标识符 [MQTT-2.3.1-2]。如果一个客户端要重发这个特殊的控制报文，在随后重发那个报文时，它**必须**使用相同的标识符。当客户端处理完这个报文对应的确认后，这个报文标识符就释放可重用。QoS 1 的 PUBLISH 对应的是 PUBACK，QoS 2 的 PUBLISH 对应的是 PUBCOMP，与 SUBSCRIBE 或 UNSUBSCRIBE 对应的分别是 SUBACK 或 UNSUBACK [MQTT-2.3.1-3]。发送一个 QoS 0 的 PUBLISH 报文时，相同的条件也适用于服务端 [MQTT-2.3.1-4]。

QoS 设置为 0 的 PUBLISH 报文**不能**包含报文标识符 [MQTT-2.3.1-5]。

PUBACK, PUBREC, PUBREL 报文**必须**包含与最初发送的 PUBLISH 报文相同的报文标识符 [MQTT-2.3.1-6]。类似地，SUBACK 和 UNSUBACK **必须**包含在对应的 SUBSCRIBE 和 UNSUBSCRIBE 报文中使用的报文标识符 [MQTT-2.3.1-7]。

需要报文标识符的控制报文在 表格 2.5 -包含报文标识符的控制报文 中列出。

表格 2.5 -包含报文标识符的控制报文

控制报文	报文标识符字段
CONNECT	不需要
CONNACK	不需要
PUBLISH	需要（如果 QoS > 0）
PUBACK	需要
PUBREC	需要
PUBREL	需要
PUBCOMP	需要
SUBSCRIBE	需要
SUBACK	需要
UNSUBSCRIBE	需要
UNSUBACK	需要
PINGREQ	不需要
PINGRESP	不需要
DISCONNECT	不需要

客户端和服务端彼此独立地分配报文标识符。因此，客户端服务端组合使用相同的报文标识符可以实现并发的消息交换。

非规范评注

客户端发送标识符为 0x1234 的 PUBLISH 报文，它有可能会在收到那个报文的 PUBACK 之前，先收到服务端发送的另一个不同的但是报文标识符也为 0x1234 的 PUBLISH 报文。

```
Client                  Server
PUBLISH Packet Identifier=0x1234--->
<---PUBLISH Packet Identifier=0x1234
PUBACK Packet Identifier=0x1234--->
<---PUBACK Packet Identifier=0x1234
```

2.4 有效载荷

某些 MQTT 控制报文在报文的最后部分包含一个有效载荷，这将在第三章论述。对于 PUBLISH 来说有效载荷就是应用消息。表格 2.6 – 包含有效载荷的控制报文 列出了需要有效载荷的控制报文。

表格 2.6 – 包含有效载荷的控制报文

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

3 MQTT 控制报文

3.1 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 [MQTT-3.1.0-1]。

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 [MQTT-3.1.0-2]。有关错误处理的信息请查看 4.8 节。

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。

3.1.1 固定报头

图例 3.1 –CONNECT 报文的固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 报文类型 (1)				Reserved 保留位			
	0	0	0	1	0	0	0	0
byte 2…	剩余长度值							

剩余长度字段

剩余长度等于可变报头的长度（10 字节）加上有效载荷的长度。编码方式见 2.2.3 节的说明。

3.1.2 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）。

3.1.2.1 协议名

图例 3.2 -协议名字节构成

	说明	7	6	5	4	3	2	1	0
协议名									
byte 1	长度 MSB (0)	0	0	0	0	0	0	0	0
byte 2	长度 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

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 [MQTT-3.1.2-1]。

非规范评注

数据包检测工具，例如防火墙，可以使用协议名来识别 MQTT 流量。

3.1.2.2 协议级别

图例 3.3 - Protocol Level byte 协议级别字节构成

	说明	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
协议级别									
byte 7	Level(4)	0	0	0	0	0	1	0	0

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 [MQTT-3.1.2-2]。

3.1.2.3 连接标志

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。

图例 3.4 -连接标志位

Bit	**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

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 [MQTT-3.1.2-3]。

3.1.2.4 清理会话

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。

如果清理会话（CleanSession）标志被设置为 0，服务端**必须**基于当前会话（使用客户端标识符识别）的状态恢复与客户端的通信。如果没有与这个客户端标识符关联的会话，服务端**必须**创建一个新的会话。在连接断开之后，当连接断开后，客户端和服务端**必须**保存会话信息 [MQTT-3.1.2-4]。当清理会话标志为 0 的会话连接断开之后，服务端**必须**将之后的 QoS 1 和 QoS 2 级别的消息保存为会话状态的一部分，如果这些消息匹配断开连接时客户端的任何订阅 [MQTT-3.1.2-5]。服务端也**可以**保存满足相同条件的 QoS 0 级别的消息。

如果清理会话（CleanSession）标志被设置为 1，客户端和服务端**必须**丢弃之前的任何会话并开始一个新的会话。会话仅持续和网络连接同样长的时间。与这个会话关联的状态数据**不能**被任何之后的会话重用 [MQTT-3.1.2-6]。

客户端的会话状态包括：

- 已经发送给服务端，但是还没有完成确认的 QoS 1 和 QoS 2 级别的消息
- 已从服务端接收，但是还没有完成确认的 QoS 2 级别的消息。

服务端的会话状态包括：

- 会话是否存在，即使会话状态的其它部分都是空。
- 客户端的订阅信息。
- 已经发送给客户端，但是还没有完成确认的 QoS 1 和 QoS 2 级别的消息。
- 即将传输给客户端的 QoS 1 和 QoS 2 级别的消息。
- 已从客户端接收，但是还没有完成确认的 QoS 2 级别的消息。
- 可选，准备发送给客户端的 QoS 0 级别的消息。

保留消息不是服务端会话状态的一部分，会话终止时**不能**删除保留消息 [MQTT-3.1.2.7]。

有关状态存储的限制和细节见第 4.1 节。

当清理会话标志被设置为 1 时，客户端和服务端的状态删除不需要是原子操作。

非规范评注

为了确保在发生故障时状态的一致性，客户端应该使用会话状态标志 1 重复请求连接，直到连接成功。

非规范评注

一般来说，客户端连接时总是将清理会话标志设置为 0 或 1，并且不交替使用两种值。这个选择取决于具体的应用。清理会话标志设置为 1 的客户端不会收到旧的应用消息，而且在每次连接成功后都需要重新订阅任何相关的主题。清理会话标志设置为 0 的客户端会收到所有在它连接断开期间发布的 QoS 1 和 QoS 2 级别的消息。因此，要确保不丢失连接断开期间的消息，需要使用 QoS 1 或 QoS 2 级别，同时将清理会话标志设置为 0。

非规范评注

清理会话标志 0 的客户端连接时，它请求服务端在连接断开后保留它的 MQTT 会话状态。如果打算在之后的某个时间点重连到这个服务端，客户端连接应该只使用清理会话标志 0。当客户端决定之后不再使用这个会话时，应该将清理会话标志设置为 0 最后再连接一次，然后断开连接。

3.1.2.5 遗嘱标志

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 [MQTT-3.1.2-8] 。

遗嘱消息发布的条件，包括但不限于：

- 服务端检测到了一个 I/O 错误或者网络故障。
- 客户端在保持连接（Keep Alive）的时间内未能通讯。
- 客户端没有先发送 DISCONNECT 报文直接关闭了网络连接。
- 由于协议错误服务端关闭了网络连接。

如果遗嘱标志被设置为 1，连接标志中的 Will QoS 和 Will Retain 字段会被服务端用到，同时有效载荷中**必须**包含 Will Topic 和 Will Message 字段 [MQTT-3.1.2-9]。

一旦被发布或者服务端收到了客户端发送的 DISCONNECT 报文，遗嘱消息就**必须**从存储的会话状态中移除 [MQTT-3.1.2-10]。

如果遗嘱标志被设置为 0，连接标志中的 Will QoS 和 Will Retain 字段**必须**设置为 0，并且有效载荷中**不能**包含 Will Topic 和 Will Message 字段 [MQTT-3.1.2-11]。

如果遗嘱标志被设置为 0，网络连接断开时，**不能**发送遗嘱消息 [MQTT-3.1.2-12]。

服务端应该迅速发布遗嘱消息。在关机或故障的情况下，服务端可以推迟遗嘱消息的发布直到之后的重启。如果发生了这种情况，在服务器故障和遗嘱消息被发布之间可能会有一个延迟。

3.1.2.6 遗嘱 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。

如果遗嘱标志被设置为 0，遗嘱 QoS 也**必须**设置为 0(0x00) [MQTT-3.1.2-13]。

如果遗嘱标志被设置为 1，遗嘱 QoS 的值可以等于 0(0x00)，1(0x01)，2(0x02)。它的值**不能**等于 3 [MQTT-3.1.2-14]。

3.1.2.7 遗嘱保留

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。

如果遗嘱标志被设置为 0，遗嘱保留（Will Retain）标志也**必须**设置为 0 [MQTT-3.1.2-15]。

如果遗嘱标志被设置为 1：

- 如果遗嘱保留被设置为 0，服务端**必须**将遗嘱消息当作非保留消息发布 [MQTT-3.1.2-16]。
- 如果遗嘱保留被设置为 1，服务端**必须**将遗嘱消息当作保留消息发布 [MQTT-3.1.2-17]。

3.1.2.8 用户名标志

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 [MQTT-3.1.2-18]。

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 [MQTT-3.1.2-19]。

3.1.2.9 密码标志

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 [MQTT-3.1.2-20]。

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 [MQTT-3.1.2-21]。

如果用户名标志被设置为 0，密码标志也**必须**设置为 0 [MQTT-3.1.2-22]。

3.1.2.10 保持连接

图例 3.5 保持连接字节

Bit	7	6	5	4	3	2	1	0
byte 9	保持连接 Keep Alive MSB							
byte 10	保持连接 Keep Alive LSB							

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 [MQTT-3.1.2-23]。

不管保持连接的值是多少，客户端任何时候都可以发送 PINGREQ 报文，并且使用 PINGRESP 报文判断网络和服务端的活动状态。

如果保持连接的值非零，并且服务端在一点五倍的保持连接时间内没有收到客户端的控制报文，它**必须**断开客户端的网络连接，认为网络连接已断开 [MQTT-3.1.2-24]。

客户端发送了 PINGREQ 报文之后，如果在合理的时间内仍没有收到 PINGRESP 报文，它**应该**关闭到服务端的网络连接。

保持连接的值为零表示关闭保持连接功能。这意味着，服务端不需要因为客户端不活跃而断开连接。注意：不管保持连接的值是多少，任何时候，只要服务端认为客户端是不活跃或无响应的，可以断开客户端的连接。

非规范评注

保持连接的实际值是由应用指定的，一般是几分钟。允许的最大值是 18 小时 12 分 15 秒。

3.1.2.11 可变报头非规范示例

图例 3.6 -可变报头非规范示例

	描述	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
协议名									
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
协议级别									
	描述	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 7	Level (4) 级别	0	0	0	0	0	1	0	0
连接标志 Connect Flags									
byte 8	User Name Flag (1) 用户名标志 Password Flag (1) 密码标志 Will Retain (0) Will 保留标志 Will QoS (01) Will 服务质量 Will Flag (1) Will 标志 Clean Session (1) 清理会话 *Reserved* (0) 保留位	1	1	0	0	1	1	1	0
保持连接时间									
byte 9	保持连接 MSB (0)	0	0	0	0	0	0	0	0
byte 10	保持连接 LSB (10)	0	0	0	0	1	0	1	0

3.1.3 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 [MQTT-3.1.3-1]。

3.1.3.1 客户端标识符

服务端使用客户端标识符 (ClientId) 识别客户端。连接服务端的每个客户端都有唯一的客户端标识符（ClientId）。客户端和服务端都必须使用 ClientId 识别两者之间的 MQTT 会话相关的状态 [MQTT-3.1.3-2]。

客户端标识符 (ClientId) **必须**存在而且**必须**是 CONNECT 报文有效载荷的第一个字段 [MQTT-3.1.3-3]。

客户端标识符**必须**是 1.5.3 节定义的 UTF-8 编码字符串 [MQTT-3.1.3-4]。

服务端**必须**允许 1 到 23 个字节长的 UTF-8 编码的客户端标识符，客户端标识符只能包含这些字符：“0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ”（大写字母，小写字母和数字）[MQTT-3.1.3-5]。

服务端**可以**允许编码后超过 23 个字节的客户端标识符 (ClientId)。服务端**可以**允许包含不是上面列表字符的客户端标识符 (ClientId)。

服务端**可以**允许客户端提供一个零字节的客户端标识符 (ClientId)，如果这样做了，服务端**必须**将这看作特殊情况并分配唯一的客户端标识符给那个客户端。然后它**必须**假设客户端提供了那个唯一的客户端标识符，正常处理这个 CONNECT 报文 [MQTT-3.1.3-6]。

如果客户端提供了一个零字节的客户端标识符，它**必须**同时将清理会话标志设置为 1 [MQTT-3.1.3-7]。

如果客户端提供的 ClientId 为零字节且清理会话标志为 0，服务端**必须**发送返回码为 0x02（表示标识符不合格）的 CONNACK 报文响应客户端的 CONNECT 报文，然后关闭网络连接 [MQTT-3.1.3-8]。

如果服务端拒绝了这个 ClientId，它**必须**发送返回码为 0x02（表示标识符不合格）的 CONNACK 报文响应客户端的 CONNECT 报文，然后关闭网络连接 [MQTT-3.1.3-9]。

非规范评注

客户端实现可以提供一个方便的方法用于生成随机的 ClientId。当清理会话标志被设置为 0 时应该主动放弃使用这种方法。

3.1.3.2 遗嘱主题

如果遗嘱标志被设置为 1，有效载荷的下一个字段是遗嘱主题（Will Topic）。遗嘱主题**必须**是 1.5.3 节定义的 UTF-8 编码字符串 [MQTT-3.1.3-10]。

3.1.3.3 遗嘱消息

如果遗嘱标志被设置为 1，有效载荷的下一个字段是遗嘱消息。遗嘱消息定义了将被发布到遗嘱主题的应用消息，见 3.1.2.5 节的描述。这个字段由一个两字节的长度和遗嘱消息的有效载荷组成，表示为零字节或多个字节序列。长度给出了跟在后面的数据的字节数，不包含长度字段本身占用的两个字节。

遗嘱消息被发布到遗嘱主题时，它的有效载荷只包含这个字段的数据部分，不包含开头的两个长度字节。

3.1.3.4 用户名

如果用户名（User Name）标志被设置为 1，有效载荷的下一个字段就是它。用户名**必须**是 1.5.3 节定义的 UTF-8 编码字符串 [MQTT-3.1.3-11]。服务端可以将它用于身份验证和授权。

3.1.3.5 密码

如果密码（Password）标志被设置为 1，有效载荷的下一个字段就是它。密码字段包含一个两字节的长度字段，长度表示二进制数据的字节数（不包含长度字段本身占用的两个字节），后面跟着 0 到 65535 字节的二进制数据。

图例 3.7 - 密码字节

Bit	7	6	5	4	3	2	1	0
byte 1	数据长度 MSB							
byte 2	数据长度 LSB							
byte 3 ….	如果长度大于 0，这里就是数据部分							

3.1.4 响应

注意：服务器可以在同一个 TCP 端口或其他网络端点上支持多种协议（包括本协议的早期版本）。如果服务器确定协议是 MQTT 3.1.1，那么它按照下面的方法验证连接请求。

1. 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。
2. 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]。
3. 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。

如果验证成功，服务端会执行下列步骤。

1. 如果 ClientId 表明客户端已经连接到这个服务端，那么服务端**必须**断开原有的客户端连接 [MQTT-3.1.4-2]。
2. 服务端**必须**按照 3.1.2.4 节的描述执行清理会话的过程 [MQTT-3.1.4-3]。
3. 服务端**必须**发送返回码为零的 CONNACK 报文作为 CONNECT 报文的确认响应 [MQTT-3.1.4-4]。
4. 开始消息分发和保持连接状态监视。

允许客户端在发送 CONNECT 报文之后立即发送其它的控制报文；客户端不需要等待服务端的 CONNACK 报文。如果服务端拒绝了 CONNECT，它**不能**处理客户端在 CONNECT 报文之后发送的任何数据 [MQTT-3.1.4-5]。

非规范评注

客户端通常会等待一个 CONNACK 报文。然而客户端有权在收到 CONNACK 之前发送控制报文，由于不需要维持连接状态，这可以简化客户端的实现。

3.2 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 [MQTT-3.2.0-1]。

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。

3.2.1 固定报头

固定报头的格式见 图例 3.8 – CONNACK 报文固定报头 的描述。

图例 3.8 – CONNACK 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (2)				Reserved 保留位			
	0	0	1	0	0	0	0	0
byte 2	剩余长度 (2)							
	0	0	0	0	0	0	1	0

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。

3.2.2 可变报头

可变报头的格式见 图例 3.9 – CONNACK 报文可变报头 的描述。

图例 3.9 –CONNACK 报文可变报头

	描述	7	6	5	4	3	2	1	0
连接确认标志		Reserved 保留位							SP[1]
byte 1		0	0	0	0	0	0	0	X
连接返回码									
byte 2		X	X	X	X	X	X	X	X

3.2.2.1 连接确认标志

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。

3.2.2.2 当前会话

位置：连接确认标志的第 0 位。

如果服务端收到清理会话（CleanSession）标志为 1 的连接，除了将 CONNACK 报文中的返回码设置为 0 之外，还**必须**将 CONNACK 报文中的当前会话设置（Session Present）标志为 0 [MQTT-3.2.2-1]。

如果服务端收到一个 CleanSession 为 0 的连接，当前会话标志的值取决于服务端是否已经保存了 ClientId 对应客户端的会话状态。如果服务端已经保存了会话状态，它**必须**将 CONNACK 报文中的当前会话标志设置为 1 [MQTT-3.2.2-2]。如果服务端没有已保存的会话状态，它**必须**将 CONNACK 报文中的当前会话设置为 0。还需要将 CONNACK 报文中的返回码设置为 0 [MQTT-3.2.2-3]。

当前会话标志使服务端和客户端在是否有已存储的会话状态上保持一致。

一旦完成了会话的初始化设置，已经保存会话状态的客户端将期望服务端维持它存储的会话状态。如果客户端从服务端收到的当前的值与预期的不同，客户端可以选择继续这个会话或者断开连接。客户端可以丢弃客户端和服务端之间的会话状态，方法是，断开连接，将清理会话标志设置为 1，再次连接，然后再次断开连接。

如果服务端发送了一个包含非零返回码的 CONNACK 报文，它**必须**将当前会话标志设置为 0 [MQTT-3.2.2-4]。

3.2.2.3 连接返回码

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，在 表格 3.1 – 连接返回码的值 中列出。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 [MQTT-3.2.2-5].。

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

如果认为上表中的所有连接返回码都不太合适，那么服务端**必须**关闭网络连接，不需要发送 CONNACK 报文 [MQTT-3.2.2-6]。

3.2.3 有效载荷

CONNACK 报文没有有效载荷。

3.3 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。

3.3.1 固定报头

图例 3.10 – PUBLISH 报文固定报头描述了固定报头的格式

图例 3.10 – PUBLISH 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (3)				DUP	QoS 等级		RETAIN
	0	0	1	1	X	X	X	X
byte 2	剩余长度							

3.3.1.1 重发标志

位置：第 1 个字节，第 3 位

如果 DUP 标志被设置为 0，表示这是客户端或服务端第一次请求发送这个 PUBLISH 报文。如果 DUP 标志被设置为 1，表示这可能是一个早前报文请求的重发。

客户端或服务端请求重发一个 PUBLISH 报文时，**必须**将 DUP 标志设置为 1 [MQTT-3.3.1.-1].。对于 QoS 0 的消息，DUP 标志**必须**设置为 0 [MQTT-3.3.1-2]。

服务端发送 PUBLISH 报文给订阅者时，收到（入站）的 PUBLISH 报文的 DUP 标志的值不会被传播。发送（出站）的 PUBLISH 报文与收到（入站）的 PUBLISH 报文中的 DUP 标志是独立设置的，它的值**必须**单独的根据发送（出站）的 PUBLISH 报文是否是一个重发来确定 [MQTT-3.3.1-3]。

非规范评注

接收者收到一个 DUP 标志为 1 的控制报文时，不能假设它看到了一个这个报文之前的一个副本。

非规范评注

需要特别指出的是，DUP 标志关注的是控制报文本身，与它包含的应用消息无关。当使用 QoS 1 时，客户端可能会收到一个 DUP 标志为 0 的 PUBLISH 报文，这个报文包含一个它之前收到过的应用消息的副本，但是用的是不同的报文标识符。 2.3.1 节提供了有关报文标识符的更多信息。

3.3.1.2 服务质量等级

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。服务质量等级在 表格 3.2 -服务质量定义 中列出。

表格 3.2 -服务质量定义

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

PUBLISH 报文**不能**将 QoS 所有的位设置为 1。如果服务端或客户端收到 QoS 所有位都为 1 的 PUBLISH 报文，它**必须**关闭网络连接 [MQTT-3.3.1-4]。

3.3.1.3 保留标志

位置：第 1 个字节，第 0 位。

如果客户端发给服务端的 PUBLISH 报文的保留（RETAIN）标志被设置为 1，服务端**必须**存储这个应用消息和它的服务质量等级（QoS），以便它可以被分发给未来的主题名匹配的订阅者 [MQTT-3.3.1-5]。一个新的订阅建立时，对每个匹配的主题名，如果存在最近保留的消息，它**必须**被发送给这个订阅者 [MQTT-3.3.1-6]。如果服务端收到一条保留（RETAIN）标志为 1 的 QoS 0 消息，它**必须**丢弃之前为那个主题保留的任何消息。它**应该**将这个新的 QoS 0 消息当作那个主题的新保留消息，但是任何时候都**可以**选择丢弃它 — 如果这种情况发生了，那个主题将没有保留消息 [MQTT-3.3.1-7]。有关存储状态的更多信息见 4.1 节。

服务端发送 PUBLISH 报文给客户端时，如果消息是作为客户端一个新订阅的结果发送，它**必须**将报文的保留标志设为 1 [MQTT-3.3.1-8]。当一个 PUBLISH 报文发送给客户端是因为匹配一个已建立的订阅时，服务端**必须**将保留标志设为 0，不管它收到的这个消息中保留标志的值是多少 [MQTT-3.3.1-9]。

保留标志为 1 且有效载荷为零字节的 PUBLISH 报文会被服务端当作正常消息处理，它会被发送给订阅主题匹配的客户端。此外，同一个主题下任何现存的保留消息必须被移除，因此这个主题之后的任何订阅者都不会收到一个保留消息 [MQTT-3.3.1-10]。*当作正常* 意思是现存的客户端收到的消息中保留标志未被设置。服务端**不能**存储零字节的保留消息 [MQTT-3.3.1-11]。

如果客户端发给服务端的 PUBLISH 报文的保留标志位 0，服务端**不能**存储这个消息也**不能**移除或替换任何现存的保留消息 [MQTT-3.3.1-12]。

非规范评注

对于发布者不定期发送状态消息这个场景，保留消息很有用。新的订阅者将会收到最近的状态。

剩余长度字段

等于可变报头的长度加上有效载荷的长度。

3.3.2 可变报头

可变报头按顺序包含主题名和报文标识符。

3.3.2.1 主题名

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 [MQTT-3.3.2-1]。

PUBLISH 报文中的主题名**不能**包含通配符 [MQTT-3.3.2-2]。

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）[MQTT-3.3.2-3]。

3.3.2.2 报文标识符

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。

3.3.2.3 可变报头非规范示例

图例 3.11 – PUBLISH 报文可变报头非规范示例 举例说明了 表格 3.3 - PUBLISH 报文非规范示例 中简要描述的 PUBLISH 报文的可变报头。

表格 3.3 - PUBLISH 报文非规范示例

Field	Value
主题名	a/b
报文标识符	10

图例 3.11 – PUBLISH 报文可变报头非规范示例

	描述	7	6	5	4	3	2	1	0
Topic Name 主题名									
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
报文标识符									
byte 6	报文标识符 MSB (0)	0	0	0	0	0	0	0	0
byte 7	报文标识符 LSB (10)	0	0	0	0	1	0	1	0

示例中的主题名为 “a/b”，长度等于 3，报文标识符为 “10”

3.3.3 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。

3.3.4 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，见下面表格的描述 [MQTT-3.3.4-1]。

表格 3.4 – PUBLISH 报文的预期响应

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

3.3.5 动作

客户端使用 PUBLISH 报文发送应用消息给服务端，目的是分发到其它订阅匹配的客户端。

服务端使用 PUBLISH 报文发送应用消息给每一个订阅匹配的客户端。

客户端使用带通配符的主题过滤器请求订阅时，客户端的订阅可能会重复，因此发布的消息可能会匹配多个过滤器。对于这种情况，服务端**必须**将消息分发给所有订阅匹配的 QoS 等级最高的客户端 [MQTT-3.3.5-1]。服务端之后可以按照订阅的 QoS 等级，分发消息的副本给每一个匹配的订阅者。

收到一个 PUBLISH 报文时，接收者的动作取决于 4.3 节描述的 QoS 等级。

如果服务端实现不授权某个客户端发布 PUBLISH 报文，它没有办法通知那个客户端。它**必须**按照正常的 QoS 规则发送一个正面的确认，或者关闭网络连接 [MQTT-3.3.5-2]。

3.4 PUBACK –发布确认

PUBACK 报文是对 QoS 1 等级的 PUBLISH 报文的响应。

3.4.1 固定报头

图例 3.12 - PUBACK 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (4)				保留位			
	0	1	0	0	0	0	0	0

byte 2	剩余长度(2)							
	0	0	0	0	0	0	1	0

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.

3.4.2 可变报头

包含等待确认的 PUBLISH 报文的报文标识符。

图例 3.13 – PUBACK 报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.4.3 有效载荷

PUBACK 报文没有有效载荷。

3.4.4 动作

完整的描述见 4.3.2 节。

3.5 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。

3.5.1 固定报头

图例 3.14 – PUBREC 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 e (5)				保留位			
	0	1	0	1	0	0	0	0
byte 2	剩余长度 (2)							
	0	0	0	0	0	0	1	0

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。

3.5.2 可变报头

可变报头包含等待确认的 PUBLISH 报文的报文标识符。

图例 3.15 – PUBREC 报文可变报头

Bit	7	6		5	4	3	2	1	0
byte 1		报文标识符 MSB							
byte 2		报文标识符 LSB							

3.5.3 有效载荷

PUBREC 报文没有有效载荷。

3.5.4 动作

完整的描述见 4.3.3 节。

3.6 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。

3.6.1 固定报头

图例 3.16 – PUBREL 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (6)				保留位			
	0	1	1	0	0	0	1	0
byte 2	剩余长度 (2)							
	0	0	0	0	0	0	1	0

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 [MQTT-3.6.1-1]。

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.

3.6.2 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。

图例 3.17 – PUBREL 报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.6.3 有效载荷

PUBREL 报文没有有效载荷。

3.6.4 动作

完整的描述见 4.3.3 节。

3.7 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。

3.7.1 固定报头

图例 3.18 – PUBCOMP 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (7)				保留位			
	0	1	1	1	0	0	0	0
byte 2	剩余长度 (2)							
	0	0	0	0	0	0	1	0

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。

3.7.2 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。

图例 3.19 – PUBCOMP 报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.7.3 有效载荷

PUBCOMP 报文没有有效载荷。

3.7.4 动作

完整的描述见 4.3.3 节。

3.8 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。

3.8.1 固定报头

图例 3.20 – SUBSCRIBE 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (8)				保留位			
	1	0	0	0	0	0	1	0
byte 2	剩余长度							

SUBSCRIBE 控制报固定报头的第 3,2,1,0 位是保留位，**必须**分别设置为 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 [MQTT-3.8.1-1]。

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。

3.8.2 可变报头

可变报头包含客户端标识符。2.3.1 提供了有关报文标识符的更多信息。

3.8.2.1 可变报头非规范示例

图例 3.21 – 报文标识符等于 10 的可变报头，非规范示例 展示了报文标识符设置为 10 时的可变报头。

图例 3.21 – 报文标识符等于 10 的可变报头，非规范示例

	描述	7	6	5	4	3	2	1	0
报文标识符									
byte 1	报文标识符 MSB (0)	0	0	0	0	0	0	0	0
byte 2	报文标识符 LSB (10)	0	0	0	0	1	0	1	0

3.8.3 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。

图例 3.22 – SUBSCRIBE 报文有效载荷格式

描述	7	6	5	4	3	2	1	0
主题过滤器								
byte 1	长度 MSB							
byte 2	长度 LSB							
bytes 3..N	主题过滤器（Topic Filter）							
服务质量要求（Requested QoS）								
	保留位						服务质量等级	
byte N+1	0	0	0	0	0	0	X	X

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 [MQTT-3-8.3-4]。

3.8.3.1 有效载荷非规范示例

图例 3.23 – 有效载荷字节格式非规范示例 展示了 表格 3.5 – 有效载荷非规范示例 中简略描述的 SUBSCRIBE 报文的有效载荷。

表格 3.5 – 有效载荷非规范示例

主题名	“a/b”
服务质量要求	0x01
主题名	“c/d”
服务质量要求	0x02

图例 3.23 – 有效载荷字节格式非规范示例

	描述	7	6	5	4	3	2	1	0
主题过滤器（Topic Filter）									
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
服务质量要求（Requested QoS）									
byte 6	Requested QoS(1)	0	0	0	0	0	0	0	1
主题过滤器（Topic Filter）									
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
服务质量要求（Requested QoS）									
byte 12	Requested QoS(2)	0	0	0	0	0	0	1	0

3.8.4 响应

服务端收到客户端发送的一个 SUBSCRIBE 报文时，**必须**使用 SUBACK 报文响应 [MQTT-3.8.4-1]。SUBACK 报文**必须**和等待确认的 SUBSCRIBE 报文有相同的报文标识符 [MQTT-3.8.4-2]。

允许服务端在发送 SUBACK 报文之前就开始发送与订阅匹配的 PUBLISH 报文。

如果服务端收到一个 SUBSCRIBE 报文，报文的主题过滤器与一个现存订阅的主题过滤器相同，那么**必须**使用新的订阅彻底替换现存的订阅。新订阅的主题过滤器和之前订阅的相同，但是它的最大 QoS 值可以不同。与这个主题过滤器匹配的任何现存的保留消息**必须**被重发，但是发布流程**不能**中断 [MQTT-3.8.4-3]。

如果主题过滤器不同于任何现存订阅的过滤器，服务端会创建一个新的订阅并发送所有匹配的保留消息。

如果服务端收到包含多个主题过滤器的 SUBSCRIBE 报文，它**必须**如同收到了一系列的多个 SUBSCRIBE 报文一样处理那个，除了需要将它们的响应合并到一个单独的 SUBACK 报文发送 [MQTT-3.8.4-4]。

服务端发送给客户端的 SUBACK 报文对每一对主题过滤器 和 QoS 等级都**必须**包含一个返回码。这个返回码**必须**表示那个订阅被授予的最大 QoS 等级，或者表示这个订阅失败 [MQTT-3.8.4-5]。服务端可以授予比订阅者要求的低一些的 QoS 等级。为响应订阅而发出的消息的有效载荷的 QoS **必须**是原始发布消息的 QoS 和服务端授予的 QoS 两者中的最小值。如果原始消息的 QoS 是 1 而被授予的最大 QoS 是 0，允许服务端重复发送一个消息的副本给订阅者 [MQTT-3.8.4-6]。

非规范示例

对某个特定的主题过滤器，如果正在订阅的客户端被授予的最大 QoS 等级是 1，那么匹配这个过滤器的 QoS 等级 0 的应用消息会按 QoS 等级 0 分发给这个客户端。这意味着客户端最多收到这个

消息的一个副本。从另一方面说，发布给同一主题的 QoS 等级 2 的消息会被服务端降级到 QoS 等级 1 再分发给客户端，因此客户端可能会收到重复的消息副本。

如果正在订阅的客户端被授予的最大 QoS 等级是 0，那么原来按 QoS 等级 2 发布给客户端的应用消息在繁忙时可能会丢失，但是服务端不应该发送重复的消息副本。发布给同一主题的 QoS 等级 1 的消息在传输给客户端时可能会丢失或重复。

非规范评注

使用 QoS 等级 2 订阅一个主题过滤器等于是说：*我想要按照它们发布时的 QoS 等级接受匹配这个过滤器的消息* 。这意味着，确定消息分发时可能的最大 QoS 等级是发布者的责任，而订阅者可以要求服务端降低 QoS 到更适合它的等级。

3.9 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。

3.9.1 固定报头

图例 3.24 – SUBACK 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (9)				保留位			
	1	0	0	1	0	0	0	0
byte 2	剩余长度							

剩余长度字段

等于可变报头的长度加上有效载荷的长度。

3.9.2 可变报头

可变报头包含等待确认的 SUBSCRIBE 报文的报文标识符。图例 3.25 – SUBACK 报文可变报头 描述了可变报头的格式。

图例 3.25 – SUBACK 报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.9.3 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 [MQTT-3.9.3-1]。

图例 3.26 – SUBACK 报文有效载荷格式 描述了有效载荷中单字节编码的返回码字段。

图例 3.26 – SUBACK 报文有效载荷格式

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 [MQTT-3.9.3-2]。

3.9.3.1 有效载荷非规范示例

图例 3.27 -有效载荷字节格式非规范示例 展示了在 表格 3.6 -有效载荷非规范示例 简要描述的 SUBACK 报文的有效载荷。

表格 3.6 -有效载荷非规范示例

Success - Maximum QoS 0	0
Success - Maximum QoS 2	2
Failure	128

图例 3.27 -有效载荷字节格式非规范示例

	描述	7	6	5	4	3	2	1	0
byte 1	Success - Maximum QoS 0	0	0	0	0	0	0	0	0
byte 2	Success - Maximum QoS 2	0	0	0	0	0	0	1	0
byte 3	Failure	1	0	0	0	0	0	0	0

3.10 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。

3.10.1 固定报头

图例 3.28 – UNSUBSCRIBE 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (10)				保留位			
	1	0	1	0	0	0	1	0
byte 2	剩余长度							

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。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 [MQTT-3.10.1-1]。

剩余长度字段

等于可变报头的长度加上有效载荷的长度。

3.10.2 可变报头

可变报头包含一个报文标识符。2.3.1 节提供了有关报文标识符的更多信息。

图例 3.29 – UNSUBSCRIBE 报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.10.3 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、按照 1.5.3 节定义的 UTF-8 编码字符串 [MQTT-3.10.3-1]。

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 [MQTT-3.10.3-2]。有关错误处理的更多信息请查看 4.8 节。

3.10.3.1 有效载荷非规范示例

图例 3.30 -有效载荷字节格式非规范示例 展示了 表格 3.7 -有效载荷非规范示例 简要描述的 UNSUBSCRIBE 报文的有效载荷。

表格 3.7 -有效载荷非规范示例

主题过滤器	“a/b”
主题过滤器	“c/d”

图例 3.30 -有效载荷字节格式非规范示例

	描述	7	6	5	4	3	2	1	0
主题过滤器									
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
主题过滤器									
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

3.10.4 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 [MQTT-3.10.4-1]。

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 [MQTT-3.10.4-2]。
- 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 [MQTT-3.10.4-3]。
- 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 [MQTT-3.10.4-4]。即使没有删除任何主题订阅，服务端也**必须**发送一个 SUBACK 响应 [MQTT-3.10.4-5]。

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。[MQTT-3.10.4-6]。

3.11 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。

3.11.1 固定报头

图例 3.31 – UNSUBACK 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (11)				保留位			
	1	0	1	1	0	0	0	0
byte 2	剩余长度 (2)							
	0	0	0	0	0	0	1	0

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。

3.11.2 可变报头

可变报头包含等待确认的 UNSUBSCRIBE 报文的报文标识符。

图例 3.32 – UNSUBACK 报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.11.3 有效载荷

UNSUBACK 报文没有有效载荷。

3.12 PINGREQ – 心跳请求

客户端发送 PINGREQ 报文给服务端的。用于：

1. 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。
2. 请求服务端发送 响应确认它还活着。
3. 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，详细信息请查看 3.1.2.10 节。

3.12.1 固定报头

图例 3.33 – PINGREQ 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (12)				保留位			

	1	1	0	0	0	0	0	0
byte 2	剩余长度 (0)							
	0	0	0	0	0	0	0	0

3.12.2 可变报头

PINGREQ 报文没有可变报头。

3.12.3 有效载荷

PINGREQ 报文没有有效载荷。

3.12.4 响应

服务端**必须**发送 PINGRESP 报文响应客户端的 PINGREQ 报文 [MQTT-3.12.4-1]。

3.13 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，详情请查看 3.1.2.10 节。

3.13.1 固定报头

图例 3.34 – PINGRESP 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (13)				保留位			
	1	1	0	1	0	0	0	0
byte 2	剩余长度 (0)							
	0	0	0	0	0	0	0	0

3.13.2 可变报头

PINGRESP 报文没有可变报头。

3.13.3 有效载荷

PINGRESP 报文没有有效载荷。

3.14 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。

3.14.1 固定报头

图例 3.35 – DISCONNECT 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (14)				保留位			
	1	1	1	0	0	0	0	0
byte 2	剩余长度 (0)							
	0	0	0	0	0	0	0	0

服务端**必须**验证所有的保留位都被设置为 0，如果它们不为 0 **必须**断开连接 [MQTT-3.14.1-1]。

3.14.2 可变报头

DISCONNECT 报文没有可变报头。

3.14.3 有效载荷

DISCONNECT 报文没有有效载荷。

3.14.4 响应

客户端发送 DISCONNECT 报文之后：

- **必须**关闭网络连接 [MQTT-3.14.4-1]。
- **不能**通过那个网络连接再发送任何控制报文 [MQTT-3.14.4-2]。

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，具体描述见 3.1.2.5 节 [MQTT-3.14.4-3]。
- **应该**关闭网络连接，如果客户端 还没有这么做。

4 操作行为

4.1 状态存储

为了提供服务质量保证，客户端和服务端有必要存储会话状态。在整个会话期间，客户端和服务端都**必须**存储会话状态 [MQTT-4.1.0-1]。会话**必须**至少持续和它的活跃网络连接同样长的时间 [MQTT-4.1.0-2]。

服务端的保留消息不是会话状态的组成部分。服务端**应该**保留那种消息直到客户端删除它。

非规范评注

客户端和服务端实现的存储容量必然是有限的，还可能要受管理策略的限制，比如跨网络连接的会话状态的最大存储时间。已保存的会话状态丢失可能是某个管理操作造成的，例如对某个预定义条件的自动响应。它造成的后果就是会话终止。这些操作可能是资源限制或其他操作原因引发的。需要谨慎的评估客户端和服务端的存储容量，以确保存储空间够用。

非规范评注

客户端或服务端的软硬件故障都可能导致会话状态的丢失或损坏。

非规范评注

服务器和客户端操作正常可能意味着，已保存的状态丢失或损坏是管理操作或软硬件故障造成的。管理操作可能是对某个预定义条件的自动响应。这些操作可能是资源限制或其他操作原因引发的。例如，服务端可能会基于外部条件，决定不再将某个消息或某些消息分发给任何当前的或以后的客户端。

非规范评注

MQTT 用户应该评估 MQTT 客户端和服务端实现的存储容量，确保能满足需求。

4.1.1 非规范示例

例如，想要收集电表读数的用户可能会决定使用 QoS 1 等级的消息，因为他们不能接受数据在网络传输途中丢失，但是，他们可能认为客户端和服务端的数据可以存储在内存（易失性存储器）中，因为（他们觉得）电力供应是非常可靠的，不会有太大的数据丢失风险。

与之相反，停车计费支付应用的提供商可能决定任何情况下都不能让数据支付消息丢失，因此他们要求在通过网络传输之前，所有的数据必须写入到非易失性存储器中（如硬盘）。

4.2 网络连接

MQTT 协议要求基础传输层能够提供有序的、可靠的、双向传输（从客户端到服务端 和从服务端到客户端）的字节流。

非规范评注

MQTT 3.1 使用的传输层协议是 [RFC793] 定义的 TCP/IP 协议。下面的协议也支持：

- TLS 协议 [RFC5246]
- WebSocket 协议 [RFC6455]

非规范评注

TCP 端口 8883 和 1883 已在 IANA 注册，分别用于 MQTT 的 TLS 和非 TLS 通信。

无连接的网络传输协议如 UDP 是不支持的，因为他们可能会丢失数据包或对数据包重排序。

4.3 服务质量等级和协议流程

MQTT 按照这里定义的服务质量 (QoS) 等级分发应用消息。分发协议是对称的，在下面的描述中，客户端和服务端既可以是发送者也可以是接收者。分发协议关注的是从单个发送者到单个接收者的应用消息。服务端分发应用消息给多个客户端时，每个客户端独立处理。分发给客户端的出站应用消息和入站应用消息的 QoS 等级可能是不同的。

下面的非规范流程图展示了可能的实现方法。

4.3.1 QoS 0:最多分发一次

消息的分发依赖于底层网络的能力。接收者不会发送响应，发送者也不会重试。消息可能送达一次也可能根本没送达。

对于 QoS 0 的分发协议，发送者

- **必须**发送 QoS 等于 0，DUP 等于 0 的 PUBLISH 报文 [MQTT-4.3.1-1]。

对于 QoS 0 的分发协议，接收者

- 接受 PUBLISH 报文时同时接受消息的所有权。

图例 4.1 – QoS 0 协议流程图，非规范示例

发送者动作	控制报文	接收者动作
PUBLISH 报文 QoS 0, DUP=0		
	---------->	
		分发应用消息给适当的后续接收者（们）

4.3.2 QoS 1: 至少分发一次

服务质量确保消息至少送达一次。QoS 1 的 PUBLISH 报文的可变报头中包含一个报文标识符，需要 PUBACK 报文确认。2.3.1 节提供了有关报文标识符的更多信息。

对于 QoS 1 的分发协议，发送者

- 每次发送新的应用消息都**必须**分配一个未使用的报文标识符。
- 发送的 PUBLISH 报文**必须**包含报文标识符且 QoS 等于 1，DUP 等于 0。

- **必须**将这个 PUBLISH 报文看作是 *未确认的*，直到从接收者那收到对应的 PUBACK 报文。4.4 节有一个关于未确认消息的讨论。

[MQTT-4.3.2-1].

一旦发送者收到 PUBACK 报文，这个报文标识符就可以重用。

注意：允许发送者在等待确认时使用不同的报文标识符发送后续的 PUBLISH 报文。

对于 QoS 1 的分发协议，接收者

- 响应的 PUBACK 报文**必须**包含一个报文标识符，这个标识符来自接收到的、已经接受所有权的 PUBLISH 报文。
- 发送了 PUBACK 报文之后，接收者必须将任何包含相同报文标识符的入站 PUBLISH 报文当作一个新的消息，并忽略它的 DUP 标志的值。

[MQTT-4.3.2-2].

图例 4.2 – QoS 1 协议流程图，非规范示例

发送者动作	控制报文	接收者动作
存储消息		
发送 PUBLISH 报文 QoS=1, DUP=0，带报文标识符	---------->	
		开始应用消息的后续分发 [1]
	<----------	发送 PUBACK 报文，带报文标识符
丢弃消息		

[1] 不要求接收者在发送 PUBACK 之前完整分发应用消息。原来的发送者收到 PUBACK 报文之后，应用消息的所有权就会转移给这个接收者。

4.3.3 QoS 2: 仅分发一次

这是最高等级的服务质量，消息丢失和重复都是不可接受的。使用这个服务质量等级会有额外的开销。

QoS 2 的消息可变报头中有报文标识符。2.3.1 节提供了有关报文标识符的更多信息。QoS 2 的 PUBLISH 报文的接收者使用一个两步确认过程来确认收到。

对于 QoS 2 的分发协议，发送者

- 必须给要发送的新应用消息分配一个未使用的报文标识符。
- 发送的 PUBLISH 报文**必须**包含报文标识符且报文的 QoS 等于 2,，DUP 等于 0。
- **必须**将这个 PUBLISH 报文看作是 *未确认的*，直到从接收者那收到对应的 PUBREC 报文。4.4 节有一个关于未确认消息的讨论。

- 收到 PUBREC 报文后**必须**发送一个 PUBREL 报文。PUBREL 报文必须包含与原始 PUBLISH 报文相同的报文标识符。
- **必须**将这个 PUBREL 报文看作是 *未确认的*，直到从接收者那收到对应的 PUBCOMP 报文。
- 一旦发送了对应的 PUBREL 报文就**不能**重发这个 PUBLISH 报文。

[MQTT-4.3.3-1].

一旦发送者收到 PUBCOMP 报文，这个报文标识符就可以重用。

注意：允许发送者在等待确认时使用不同的报文标识符发送后续的 PUBLISH 报文。

对于QoS 2的分发协议，接收者

- 响应的 PUBREC 报文**必须**包含报文标识符，这个标识符来自接收到的、已经接受所有权的 PUBLISH 报文。
- 在收到对应的 PUBREL 报文之前，接收者**必须**发送 PUBREC 报文确认任何后续的具有相同标识符的 PUBLISH 报文。 在这种情况下，它**不能**重复分发消息给任何后续的接收者。
- 响应 PUBREL 报文的 PUBCOMP 报文**必须**包含与 PUBREL 报文相同的标识符。
- 发送 PUBCOMP 报文之后，接收者必须将包含相同报文标识符的任何后续 PUBLISH 报文当作一个新的发布。

[MQTT-4.3.3-2].

图例 4.3 – QoS 2 协议流程图，非规范示例

发送者动作	控制报文	接收者动作
存储消息		
发送 PUBLISH 报文，QoS=2, DUP=0，带报文标识符		
	---------->	
		方法 A：存储消息，或方法 B：存储报文标识符，然后开始向前分发这个应用消息 [1]。
		发送 PUBREC 报文，带报文标识符。
	<----------	
丢弃消息，存储 PUBREC 中的报文标识符		
发送 PUBREL 报文，带报文标识符		
	---------->	

		方法 A：开始向前分发应用消息[1]然后丢弃消息 或方法 B：丢弃报文标识符
		发送 PUBCOMP 报文，带报文标识符
	<----------	
丢弃已保存的状态		

[1] 不要求接收者在发送 PUBREC 或 PUBCOMP 之前完整分发应用消息。原来的发送者收到 PUBREC 报文之后，应用消息的所有权就会转移给这个接收者。

图例 4.3 – QoS 2 协议流程图，非规范示例 展示了接收者对 QoS 2 等级消息的两种处理方法。他们的区别是消息什么时候可以开始分发。实现者可以决定使用哪种方法。只要实现者只选择了一种方法，就不会影响 QoS 流程的可靠性。

4.4 消息分发重试

客户端设置清理会话（CleanSession）标志为 0 重连时，客户端和服务端**必须**使用原始的报文标识符重发任何未确认的 PUBLISH 报文（如果 QoS>0）和 PUBREL 报文 [MQTT-4.4.0-1]。这是唯**一要求**客户端或服务端重发消息的情况。

非规范评注

控制报文的重发曾经需要克服某些陈旧 TCP 网络上的数据丢失问题。部署在那些环境中的 MQTT 3.1.1 实现可能仍然需要关注这个问题。

4.5 消息收到

服务端接管入站应用消息的所有权时，它**必须**将消息添加到订阅匹配的客户端的会话状态中。匹配规则定义见 4.7 节 [MQTT-4.5.0-1]。

正常情况下，客户端收到发送给它的订阅的消息。客户端也可能收到不是与它的订阅精确匹配的消息。如果服务端自动给客户端分配了一个订阅，可能发生这种情况。正在处理 UBSUBSCRIBE 请求时也可能收到消息。客户端**必须**按照可用的服务质量（QoS）规则确认它收到的任何 PUBLISH 报文，不管它选择是否处理报文包含的应用消息 [MQTT-4.5.0-2]。

4.6 消息排序

实现本章定义的协议流程时，客户端**必须**遵循下列规则：

- 重发任何之前的 PUBLISH 报文时，**必须**按原始 PUBLISH 报文的发送顺序重发（适用于 QoS 1 和 QoS 2 消息）[MQTT-4.6.0-1]。
- **必须**按照对应的 PUBLISH 报文的顺序发送 PUBACK 报文（QoS 1 消息）[MQTT-4.6.0-2]。
- **必须**按照对应的 PUBLISH 报文的顺序发送 PUBREC 报文（QoS 2 消息）[MQTT-4.6.0-3]。
- **必须**按照对应的 PUBREC 报文的顺序发送 PUBREL 报文（QoS 2 消息）[MQTT-4.6.0-4]。

服务端**必须**默认认为每个主题都是有序的。它**可以**提供一个管理功能或其它机制，以允许将一个或多个主题当作是无序的 [MQTT-4.6.0-5]。

服务端处理发送给有序主题的消息时，**必须**按照上面的规则将消息分发给每个订阅者。此外，它**必须**按照从客户端收到的顺序发送 PUBLISH 报文给消费者（对相同的主题和 QoS）[MQTT-4.6.0-6]。

非规范评注

上面列出的规则确保，使用 QoS 1 发布和订阅的消息流，订阅者按照消息发布时的顺序收到每条消息的最终副本，但是消息可能会重复，这可能导致在它的后继消息之后收到某个已经收到消息的重发版本。例如，发布者按顺序 1,2,3,4 发送消息，订阅者收到的顺序可能是 1,2,3,2,3,4。

如果客户端和服务端能保证任何时刻最多有一条消息在 *传输中（in-flight）*（在某条消息被确认前不发送后面的那条消息），那么，不会有 QoS 1 的消息会在它的任何后续消息之后收到。 例如，订阅者收到的顺序可能是 1,2,3,3,4，而不是 1,2,3,2,3,4。将传输窗口 （in-flight window）设为 1 意味着，在同一个主题上，即使发布者发送了一系列不同 QoS 等级的消息，它们的顺序也被保留。

4.7 主题名和主题过滤器

4.7.1 主题通配符

主题层级（topic level）分隔符用于将结构化引入主题名。如果存在分隔符，它将主题名分割为多个*主题层级 topic level* 。

订阅的主题过滤器可以包含特殊的通配符，允许你一次订阅多个主题。

主题过滤器中可以使用通配符，但是主题名**不能**使用通配符 [MQTT-4.7.1-1]。

4.7.1.1 主题层级分隔符

斜杠（‘/’ U+002F）用于分割主题的每个层级，为主题名提供一个分层结构。当客户端订阅指定的主题过滤器包含两种通配符时，主题层级分隔符就很有用了。主题层级分隔符可以出现在主题过滤器或主题名字的任何位置。相邻的主题层次分隔符表示一个零长度的主题层级。

4.7.1.2 多层通配符

数字标志（‘#’ U+0023）是用于匹配主题中任意层级的通配符。多层通配符表示它的父级和任意数量的子层级。多层通配符必须位于它自己的层级或者跟在主题层级分隔符后面。不管哪种情况，它都**必须**是主题过滤器的最后一个字符 [MQTT-4.7.1-2]。

非规范评注

例如，如果客户端订阅主题 “sport/tennis/player1/#”，它会收到使用下列主题名发布的消息：

- “sport/tennis/player1”
- “sport/tennis/player1/ranking”
- “sport/tennis/player1/score/wimbledon”

非规范评注

- “sport/#”也匹配单独的 “sport” ，因为 # 包括它的父级。
- “#”是有效的，会收到所有的应用消息。
- “sport/tennis/#”也是有效的。
- “sport/tennis#”是无效的。

- “sport/tennis/#/ranking”是无效的。

4.7.1.3 单层通配符

加号（‘+’ U+002B) 是只能用于单个主题层级匹配的通配符。

在主题过滤器的任意层级都可以使用单层通配符，包括第一个和最后一个层级。然而它**必须**占据过滤器的整个层级 [MQTT-4.7.1-3]。可以在主题过滤器中的多个层级中使用它，也可以和多层通配符一起使用。

非规范评注

例如， “sport/tennis/+” 匹配 “sport/tennis/player1” 和 “sport/tennis/player2”，但是不匹配 “sport/tennis/player1/ranking”。同时，由于单层通配符只能匹配一个层级， “sport/+” 不匹配 “sport” 但是却匹配 “sport/”。

非规范评注

- “+” 是有效的。
- “+/tennis/#” 是有效的。
- “sport+” 是无效的。
- “sport/+/player1” 也是有效的。
- “/finance” 匹配 “+/+” 和 “/+”，但是不匹配 “+”。

4.7.2 以$开头的主题

服务端**不能**将 $ 字符开头的主题名匹配通配符 (#或+) 开头的主题过滤器 [MQTT-4.7.2-1]。服务端应该阻止客户端使用这种主题名与其它客户端交换消息。服务端实现可以将 $ 开头的主题名用作其他目的。

非规范评注

- $SYS/ 被广泛用作包含服务器特定信息或控制接口的主题的前缀。
- 应用不能使用 $ 字符开头的主题。

非规范评注

- 订阅 “#” 的客户端不会收到任何发布到以 “$” 开头主题的消息。
- 订阅 “+/monitor/Clients” 的客户端不会收到任何发布到 “$SYS/monitor/Clients” 的消息。
- 订阅 “$SYS/#” 的客户端会收到发布到以 “$SYS/” 开头主题的消息。
- 订阅 “$SYS/monitor/+” 的客户端会收到发布到 “$SYS/monitor/Clients” 主题的消息。
- 如果客户端想同时接受以 “$SYS/” 开头主题的消息和不以 $ 开头主题的消息，它需要同时订阅 “#” 和 ““$SYS/#”。

4.7.3 主题语义和用法

主题名和主题过滤器必须符合下列规则：

- 所有的主题名和主题过滤器**必须**至少包含一个字符 [MQTT-4.7.3-1]。
- 主题名和主题过滤器是区分大小写的。
- 主题名和主题过滤器可以包含空格。

- 主题名或主题过滤器以前置或后置斜杠 “/” 区分。
- 只包含斜杠 “/” 的主题名或主题过滤器是合法的。
- 主题名和主题过滤器**不能**包含空字符 (Unicode U+0000) [Unicode] [MQTT-4.7.3-2]。
- 主题名和主题过滤器是 UTF-8 编码字符串，它们**不能**超过 65535 字节 [MQTT-4.7.3-3]。见 1.5.3 节。

除了不能超过 UTF-编码字符串的长度限制之外，主题名或主题过滤器的层级数量没有其它限制。

匹配订阅时，服务端**不能**对主题名或主题过滤器执行任何规范化（normalization）处理，不能修改或替换任何未识别的字符 [MQTT-4.7.3-4]。主题过滤器中的每个非通配符层级需要逐字符匹配主题名中对应的层级才算匹配成功。

非规范评注

使用 UTF-8 编码规则意味着，主题过滤器和主题名的比较可以通过比较编码后的 UTF-8 字节或解码后的 Unicode 字符。

非规范评注

- “ACCOUNTS” 和 “Accounts” 是不同的主题名。
- “Accounts payable” 是合法的主题名
- “/finance” 和 “finance” 是不同的。

如果订阅的主题过滤器与消息的主题名匹配，应用消息会被发送给每一个匹配的客户端订阅。主题可能是管理员在服务端预先定义好的，也可能是服务端收到第一个订阅或使用那个主题名的应用消息时动态添加的。服务端也可以使用一个安全组件有选择地授权客户端使用某个主题资源。

4.8 错误处理

除非另有说明，如果服务端或客户端遇到了协议违规的行为，它**必须**关闭传输这个协议违规控制报文的网络连接 [MQTT-4.8.0-1]。

客户端或服务端实现可能会遇到瞬时错误（Transient Error）（例如内部缓冲区满了的情况）导致无法成功处理 MQTT 报文。

如果客户端或服务端处理入站控制报文时遇到了瞬时错误，它**必须**关闭传输那个控制报文的网络连接 [MQTT-4.8.0-2]。如果服务端发现了瞬时错误，它**不应该**断开连接或者执行任何对其它客户端有影响的操作。

5 安全

5.1 概述

本章的内容仅供参考，是非规范化的。然而，强烈推荐提供 TLS 的服务端实现**应该**使用 TCP 端口 8883（IANA 服务名：secure-mqtt）。

解决方案提供者需要考虑很多风险。例如：

- 设备可能会被盗用
- 客户端和服务端的静态数据可能是可访问的（可能会被修改）
- 协议行为可能有副作用（如计时器攻击）
- 拒绝服务攻击
- 通信可能会被拦截、修改、重定向或者泄露
- 虚假控制报文注入

MQTT 方案通常部署在不安全的通信环境中。在这种情况下，协议实现通常需要提供这些机制：

- 用户和设备身份认证
- 服务端资源访问授权
- MQTT 控制报文和内嵌应用数据的完整性校验
- MQTT 控制报文和内嵌应用数据的隐私控制

作为传输层协议，MQTT 仅关注消息传输，提供合适的安全功能是实现者的责任。使用 TLS [RFC5246] 是比较普遍的选择。除了技术上的安全问题外，还有地理因素（例如美国欧盟安全港原则 [USEUSAFEHARB]），行业标准（例如第三方支付行业数据安全标准 [PCIDSS]），监管方面的考虑（例如萨班斯-奥克斯利法案 [SARBANES]）等问题。

5.2 MQTT 解决方案：安全和认证

协议实现可能想要符合特定的工业安全标准，如 NIST 网络安全框架 [NISTCSF]，第三方支付行业数据安全标准 [PCIDSS]，美国联邦信息处理标准 [FIPS1402] 和 NSA 加密组合 B [NSAB]。

在 MQTT 的补充出版物 （MQTT and the NIST Framework for Improving Critical Infrastructure Cybersecurity [MQTT NIST]） 中可以找到在 NIST 网络安全框架 [NISTCSF] 中使用 MQTT 的指导。使用行业证明、独立审计和认证技术有助于满足合规要求。

5.3 轻量级的加密与受限设备

广泛采用高级加密标准 [AES] 数据加密标准 [DES]。

推荐使用为受限的低端设备特别优化过的轻量级加密国际标准 ISO 29192 [ISO29192]。

5.4 实现注意事项

实现和使用 MQTT 时需要考虑许多安全问题。下面的部分不应该被当作是一个 核对清单 。

协议实现可以实现下面的一部分或全部：

5.4.1 客户端身份验证

CONNECT 报文包含用户名和密码字段。实现可以决定如何使用这些字段的内容。实现者可以提供自己的身份验证机制，或者使用外部的认证系统如 LDAP [RFC4511] 或 OAuth [RFC6749]，还可以利用操作系统的认证机制。

实现可以明文传递认证数据，混淆那些数据，或者不要求任何认证数据，但应该意识到这会增加中间人攻击和重放攻击的风险。5.4.5 节介绍了确保数据私密的方法。

在客户端和服务端之间使用虚拟专用网（VPN）可以确保数据只被授权的客户端收到。

使用 TLS [RFC5246] 时，服务端可以使用客户端发送的 SSL 证书验证客户端的身份。

实现可以允许客户端通过应用消息给服务端发送凭证用于身份验证。

5.4.2 客户端授权

基于客户端提供的信息如用户名、客户端标识符（ClientId）、客户端的主机名或 IP 地址，或者身份认证的结果，服务端可以限制对某些服务端资源的访问。

5.4.3 服务端身份验证

MQTT 协议不是双向信任的，它没有提供客户端验证服务端身份的机制。

但是使用 TLS [RFC5246] 时，客户端可以使用服务端发送的 SSL 证书验证服务端的身份。从单 IP 多域名提供 MQTT 服务的实现应该考虑 RFC6066 [RFC6066] 第 3 节定义的 TLS 的 SNI 扩展。SNI 允许客户端告诉服务端它要连接的服务端主机名。

实现可以允许服务端通过应用消息给客户端发送凭证用于身份验证。

在客户端和服务端之间使用虚拟专用网（VPN）可以确保客户端连接的是预期的服务器。

5.4.4 控制报文和应用消息的完整性

应用可以在应用消息中单独包含哈希值。这样做可以为 PUBLISH 控制报文的网络传输和静态数据提供内容的完整性检查。

TLS [RFC5246] 提供了对网络传输的数据做完整性校验的哈希算法。

在客户端和服务端之间使用虚拟专用网（VPN）连接可以在 VPN 覆盖的网络段提供数据完整性检查。

5.4.5 控制报文和应用消息的保密性

TLS [RFC5246] 可以对网络传输的数据加密。如果有效的 TLS 密码组合包含的加密算法为 NULL，那么它不会加密数据。要确保客户端和服务端的保密，应避免使用这些密码组合。

应用可以单独加密应用消息的内容。这可以提供应用消息传输途中和静态数据的私密性。但不能给应用消息的其它属性如主题名加密。

客户端和服务端实现可以加密存储静态数据，例如可以将应用消息作为会话的一部分存储。

在客户端和服务端之间使用虚拟专用网（VPN）连接可以在 VPN 覆盖的网络段保证数据的私密性。

5.4.6 消息传输的不可否认性

应用设计者可能需要考虑适当的策略，以实现端到端的不可否认性（non-repudiation）。

5.4.7 检测客户端和服务端的盗用

使用 TLS [RFC5246] 的客户端和服务端实现应该能够确保，初始化 TLS [RFC5246] 连接时提供的 SSL 证书是与主机名（客户端要连接的或服务端将被连接的）关联的。

使用 TLS [RFC5246] 的客户端和服务端实现，可以选择提供检查证书吊销列表（CRLs [RFC5280]) 和在线证书状态协议（OSCP) [RFC6960] 的功能，拒绝使用被吊销的证书。

物理部署可以将防篡改硬件与应用消息的特殊数据传输结合。例如，一个仪表可能会内置一个 GPS 以确保没有在未授权的地区使用。IEEE 安全设备认证 [IEEE 802.1AR] 就是用于实现这个机制的一个标准，它使用加密绑定标识符验证设备身份。

5.4.8 检测异常行为

服务端实现可以监视客户端的行为，检测潜在的安全风险。例如：

- 重复的连接请求
- 重复的身份验证请求
- 连接的异常终止
- 主题扫描（请求发送或订阅大量主题）
- 发送无法送达的消息（没有订阅者的主题）
- 客户端连接但是不发送数据

发现违反安全规则的行为，服务端实现可以断开客户端连接。

服务端实现检测不受欢迎的行为，可以基于 IP 地址或客户端标识符实现一个动态黑名单列表。

服务部署可以使用网络层次控制（如果可用）实现基于 IP 地址或其它信息的速率限制或黑名单。

5.4.9 其它的安全注意事项

如果客户端或服务端的 SSL 证书丢失，或者我们考虑证书被盗用或者被吊销(利用 CRLs [RFC5280] 和 OSCP [RFC6960])的情况。

客户端或服务端验证凭证时，如果发现用户名和密码丢失或被盗用，应该吊销或者重新发放。

在使用长连接时：

- 客户端和服务端使用 TLS [RFC5246] 时应该允许重新协商会话以确认新的加密参数（替换会话密钥，更换密码组合，更换认证凭证）。
- 服务端可以断开客户端连接，并要求他们使用新的凭证重新验证身份。

受限网络上的受限设备和客户端可以使用 TLS 会话恢复 [RFC5077] 降低 TLS 会话重连 [RFC5246] 的成本。

连接到服务端的客户端与其它连接到服务端的客户端 之间有一个信任传递关系，它们都有权在同一个主题上发布消息。

5.4.10 使用 SOCKS 代理

客户端实现应该意识到某些环境要求使用 SOCKSv5 [RFC1928] 代理创建出站的网络连接。某些 MQTT 实现可以利用安全隧道（如 SSH）通过 SOCKS 代理。一个实现决定支持 SOCKS 时，它们应该同时支持匿名的和用户名密码验证的 SOCKS 代理。对于后一种情况，实现应该意识到 SOCKS 可能使用明文认证，因此应该避免使用相同的凭证连接 MQTT 服务器。

5.4.11 安全配置文件

实现者和方案设计者可能希望将安全当作配置文件集合应用到 MQTT 协议中。下面描述的是一个分层的安全等级结构。

5.4.11.1 开放通信配置

使用开放通信配置时，MQTT 协议运行在一个没有内置额外安全通信机制的开放网络上。

5.4.11.2 安全网络通信配置

使用安全网络通信配置时，MQTT 协议运行在有安全控制的物理或虚拟网络上，如 VPN 或物理安全网络。

5.4.11.3 安全传输配置

使用安全传输配置时，MQTT 协议运行在使用 TLS [RFC5246] 的物理或虚拟网络上，它提供了身份认证，完整性和保密性。

使用内置的用户名和密码字段，TLS [RFC5246] 客户端身份认证可被用于（或者代替）MQTT 客户端认证。

5.4.11.4 工业标准的安全配置

可以预料的是，MQTT 被设计为支持很多工业标准的应用配置，每一种定义一个威胁模型和用于定位威胁的特殊安全机制。特殊的安全机制推荐从下面的方案中选择：

[NISTCSF] NIST 网络安全框架
[NIST7628] NISTIR 7628 智能电网网络安全指南
[FIPS1402] (FIPS PUB 140-2) 加密模块的安全要求
[PCIDSS] PCI-DSS 第三方支付行业数据安全标准
[NSAB] NSA 加密组合 B

6 使用 WebSocket 作为网络层

如果 MQTT 在 WebSocket [RFC6455] 连接上传输，**必须**满足下面的条件：

- MQTT 控制报文**必须**使用 WebSocket 二进制数据帧发送。如果收到任何其它类型的数据帧，接收者**必须**关闭网络连接 [MQTT-6.0.0-1]。
- 单个 WebSocket 数据帧可以包含多个或者部分 MQTT 报文。接收者**不能**假设 MQTT 控制报文按 WebSocket 帧边界对齐 [MQTT-6.0.0-2]。
- 客户端**必须**将字符串 **mqtt** 包含在它提供的 WebSocket 子协议列表里 [MQTT-6.0.0-3]。
- 服务端选择和返回的 WebSocket 子协议名**必须**是 **mqtt** [MQTT-6.0.0-4] 。
- 用于连接客户端和服务器的 WebSocket URI 对 MQTT 协议没有任何影响。

6.1 IANA 注意事项

本规范请求 IANA 在 WebSocket 子协议名条目下注册 WebSocket MQTT 子协议，使用下列数据：

图例 6.1 - IANA WebSocket 标识符

子协议标识符	mqtt
子协议通用名	mqtt
子协议定义	http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

7 一致性

MQTT 规范定义了 MQTT 客户端实现和 MQTT 服务端实现的一致性要求

MQTT 实现可以同时是 MQTT 客户端和 MQTT 服务端。接受入站连接和建立到其它服务端的出站连接的服务端必须同时符合 MQTT 客户端和 MQTT 服务端的要求 [MQTT-7.0.0-1]。

为了与任何其它的一致性实现交互操作，一致性实现不能要求使用在本规范之外定义的任何扩展 [MQTT-7.0.0-2]。

7.1 一致性目标

7.1.1 MQTT 服务端

一个 MQTT 服务端只有满足下面所有的要求才算是符合本规范：

1. 服务端发送的所有控制报文的格式必须符合第二章和第三章描述的格式
2. 遵守第 4.7 节描述的主题匹配规则。
3. 满足下列章节中所有**必须**级别的要求，明确仅适用于对客户端的除外：
    - 第一章 – 介绍
    - 第二章 – MQTT 控制报文格式
    - 第三章 – MQTT 控制报文
    - 第四章 – 操作行为
    - 第六章 –（如果 MQTT 的网络层是 WebSocket）
    - 第七章 – 一致性目标

满足一致性要求的服务端**必须**支持使用一个或多个底层传输协议，只要它提供有序的、可靠的、双向字节流（从客户端到服务端和从服务端到客户端）[MQTT-7.1.1-1]。但是一致性并不依赖于它支持任何特定的传输协议。服务端**可以**支持第 4.2 节列出的任何传输协议，或者任何其它满足 [MQTT-7.1.1-1] 要求的传输协议。

7.1.2 MQTT 客户端

一个 MQTT 客户端只有满足下面所有的要求才算是符合本规范：

1. 客户端发送的所有控制报文的格式必须符合第二章和第三章描述的格式
2. 满足下列章节中所有**必须**级别的要求，明确仅适用于对服务端的除外：
    - 第一章 – 介绍
    - 第二章 – MQTT 控制报文格式
    - 第三章 – MQTT 控制报文
    - 第四章 – 操作行为
    - 第六章 – （如果 MQTT 的网络层是 WebSocket）
    - 第七章 – 一致性目标

满足一致性要求的客户端**必须**支持使用一个或多个底层传输协议，只要它提供有序的、可靠的、双向字节流（从客户端到服务端和从服务端到客户端）[MQTT-7.1.2-1]。但是一致性并不依赖于它支持任何特定的

传输协议。客户端**可以**支持第 4.2 节列出的任何传输协议，或者任何其它满足 [MQTT-7.1.2-1] 要求的传输协议。

附录 B 强制性规范声明（非规范）

这个附录是非规范的，只作为本文档正文中可以找到的大量一致性声明的摘要提供。一致性要求的限制列表见第七章。

声明序号	规范声明
[MQTT-1.5.3-1]	UTF-8 编码字符串中的字符数据**必须**是按照 Unicode 规范 [Unicode] 定义的和在 RFC3629 [RFC3629] 中重申的有效的 UTF-8 格式。特别需要指出的是，这些数据**不能**包含字符码在 U+D800 和 U+DFFF 之间的数据。如果服务端或客户端收到了一个包含无效 UTF-8 字符的控制报文，它**必须**关闭网络连接。
[MQTT-1.5.3-2]	UTF-8 编码的字符串**不能**包含空字符 U+0000。如果客户端或服务端收到了一个包含 U+0000 的控制报文，它**必须**关闭网络连接。
[MQTT-1.5.3-3]	UTF-8 编码序列 0XEF 0xBB 0xBF 总是被解释为 U+FEFF（零宽度非换行空白字符），无论它出现在字符串的什么位置，报文接收者都不能跳过或者剥离它。
[MQTT-2.2.2-1]	表格 2.2 中任何标记为“保留”的标志位，都是保留给以后使用的，**必须**设置为表格中列出的值。
[MQTT-2.2.2-2]	如果收到非法的标志，接收者**必须**关闭网络连接。
[MQTT-2.3.1-1]	SUBSCRIBE，UNSUBSCRIBE 和 PUBLISH（QoS 大于 0）控制报文**必须**包含一个非零的 16 位报文标识符（Packet Identifier。
[MQTT-2.3.1-2]	客户端每次发送一个新的这些类型的报文时都**必须**分配一个当前未使用的报文标识符。
[MQTT-2.3.1-3]	如果一个客户端要重发这个特殊的控制报文，在随后重发那个报文时，它**必须**使用相同的标识符。当客户端处理完这个报文对应的确认后，这个报文标识符就释放可重用。QoS 1 的 PUBLISH 对应的是 PUBACK，QoS 2 的 PUBLISH 对应的是 PUBCOMP，与 SUBSCRIBE 或 UNSUBSCRIBE 对应的分别是 SUBACK 或 UNSUBACK
[MQTT-2.3.1-4]	发送一个 QoS 0 的 PUBLISH 报文时，相同的条件也适用于服务端。
[MQTT-2.3.1-5]	QoS 设置为 0 的 PUBLISH 报文**不能**包含报文标识符。
[MQTT-2.3.1-6]	PUBACK, PUBREC, PUBREL 报文**必须**包含与最初发送的 PUBLISH 报文相同的报文标识符。
[MQTT-2.3.1-7]	与 [MQTT-2.3.1-6] 类似，SUBACK 和 UNSUBACK **必须**包含在对应的 SUBSCRIBE 和 UNSUBSCRIBE 报文中使用的报文标识符。
[MQTT-3.1.0-1]	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。
[MQTT-3.1.0-2]	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。
[MQTT-3.1.2-1]	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。

[MQTT-3.1.2-2]	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。
[MQTT-3.1.2-3]	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。
[MQTT-3.1.2-4]	如果清理会话（CleanSession）标志被设置为 0，服务端**必须**基于当前会话（使用客户端标识符识别）的状态恢复与客户端的通信。如果没有与这个客户端标识符关联的会话，服务端**必须**创建一个新的会话。在连接断开之后，当连接断开后，客户端和服务端**必须**保存会话信息。
[MQTT-3.1.2-5]	当清理会话标志为 0 的会话连接断开之后，服务端**必须**将之后的 QoS 1 和 QoS 2 级别的消息保存为会话状态的一部分，如果这些消息匹配断开连接时客户端的任何订阅。
[MQTT-3.1.2-6]	如果清理会话（CleanSession）标志被设置为 1，客户端和服务端**必须**丢弃之前的任何会话并开始一个新的会话。会话仅持续和网络连接同样长的时间。与这个会话关联的状态数据**不能**被任何之后的会话重用。
[MQTT-3.1.2.7]	保留消息不是服务端会话状态的一部分，会话终止时**不能**删除保留消息。
[MQTT-3.1.2-8]	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。
[MQTT-3.1.2-9]	如果遗嘱标志被设置为 1，连接标志中的 Will QoS 和 Will Retain 字段会被服务端用到，同时有效载荷中**必须**包含 Will Topic 和 Will Message 字段。
[MQTT-3.1.2-10]	一旦被发布或者服务端收到了客户端发送的 DISCONNECT 报文，遗嘱消息就**必须**从存储的会话状态中移除。
[MQTT-3.1.2-11]	如果遗嘱标志被设置为 0，连接标志中的 Will QoS 和 Will Retain 字段**必须**设置为 0，并且有效载荷中**不能**包含 Will Topic 和 Will Message 字段。
[MQTT-3.1.2-12]	如果遗嘱标志被设置为 0，网络连接断开时，**不能**发送遗嘱消息。.
[MQTT-3.1.2-13]	如果遗嘱标志被设置为 0，遗嘱 QoS 也**必须**设置为 0(0x00)。
[MQTT-3.1.2-14]	如果遗嘱标志被设置为 1，遗嘱 QoS 的值可以等于 0(0x00)，1(0x01)，2(0x02)。它的值**不能**等于 3。
[MQTT-3.1.2-15]	如果遗嘱标志被设置为 0，遗嘱保留（Will Retain）标志也**必须**设置为 0。
[MQTT-3.1.2-16]	如果遗嘱保留被设置为 0，服务端**必须**将遗嘱消息当作非保留消息发布。
[MQTT-3.1.2-17]	如果遗嘱保留被设置为 1，服务端**必须**将遗嘱消息当作保留消息发布。
[MQTT-3.1.2-18]	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。
[MQTT-3.1.2-19]	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。
[MQTT-3.1.2-20]	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。
[MQTT-3.1.2-21]	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

[MQTT-3.1.2-22]	如果用户名标志被设置为 0，密码标志也**必须**设置为 0。
[MQTT-3.1.2-23]	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。
[MQTT-3.1.2-24]	如果保持连接的值非零，并且服务端在一点五倍的保持连接时间内没有收到客户端的控制报文，它**必须**断开客户端的网络连接，认为网络连接已断开。
[MQTT-3.1.3-1]	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。
[MQTT-3.1.3-2]	服务端使用客户端标识符 (ClientId) 识别客户端。连接服务端的每个客户端都有唯一的客户端标识符（ClientId）。客户端和服务端都必须使用 ClientId 识别两者之间的 MQTT 会话相关的状态。
[MQTT-3.1.3-3]	客户端标识符 (ClientId) **必须**存在而且**必须**是 CONNECT 报文有效载荷的第一个字段。
[MQTT-3.1.3-4]	客户端标识符**必须**是 1.5.3 节定义的 UTF-8 编码字符串。
[MQTT-3.1.3-5]	服务端**必须**允许 1 到 23 个字节长的 UTF-8 编码的客户端标识符，客户端标识符只能包含这些字符：“0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ”（大写字母，小写字母和数字）。
[MQTT-3.1.3-6]	服务端**可以**允许客户端提供一个零字节的客户端标识符 (ClientId)，如果这样做了，服务端**必须**将这看作特殊情况并分配唯一的客户端标识符给那个客户端。然后它**必须**假设客户端提供了那个唯一的客户端标识符，正常处理这个 CONNECT 报文。
[MQTT-3.1.3-7]	如果客户端提供了一个零字节的客户端标识符，它**必须**同时将清理会话标志设置为 1。
[MQTT-3.1.3-8]	如果客户端提供的 ClientId 为零字节且清理会话标志为 0，服务端**必须**发送返回码为 0x02（表示标识符不合格）的 CONNACK 报文响应客户端的 CONNECT 报文，然后关闭网络连接。
[MQTT-3.1.3-9]	如果服务端拒绝了这个 ClientId，它**必须**发送返回码为 0x02（表示标识符不合格）的 CONNACK 报文响应客户端的 CONNECT 报文，然后关闭网络连接。
[MQTT-3.1.3-10]	遗嘱主题**必须**是 1.5.3 节定义的 UTF-8 编码字符串。
[MQTT-3.1.3-11]	用户名**必须**是 1.5.3 节定义的 UTF-8 编码字符串。
[MQTT-3.1.4-1]	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-3.1.4-2]	如果 ClientId 表明客户端已经连接到这个服务端，那么服务端**必须**断开原有的客户端连接。
[MQTT-3.1.4-3]	服务端**必须**按照 3.1.2.4 节的描述执行清理会话的过程。
[MQTT-3.1.4-4]	服务端**必须**发送返回码为零的 CONNACK 报文作为 CONNECT 报文的确认响应。
[MQTT-3.1.4-5]	如果服务端拒绝了 CONNECT，它**不能**处理客户端在 CONNECT 报文之后发送的任何数据。

[MQTT-3.2.0-1]	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。
[MQTT-3.2.2-1]	如果服务端收到清理会话（CleanSession）标志为 1 的连接，除了将 CONNACK 报文中的返回码设置为 0 之外，还**必须**将 CONNACK 报文中的当前会话设置（Session Present）标志为 0。
[MQTT-3.2.2-2]	如果服务端收到一个 CleanSession 为 0 的连接，当前会话标志的值取决于服务端是否已经保存了 ClientId 对应客户端的会话状态。如果服务端已经保存了会话状态，它**必须**将 CONNACK 报文中的当前会话标志设置为 1。
[MQTT-3.2.2-3]	如果服务端没有已保存的会话状态，它**必须**将 CONNACK 报文中的当前会话设置为 0。还需要将 CONNACK 报文中的返回码设置为 0。
[MQTT-3.2.2-4]	如果服务端发送了一个包含非零返回码的 CONNACK 报文，它**必须**将当前会话标志设置为 0。
[MQTT-3.2.2-5]	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。.
[MQTT-3.2.2-6]	如果认为上表格 3.1 中的所有连接返回码都不太合适，那么服务端**必须**关闭网络连接，不需要发送 CONNACK 报文。
[MQTT-3.3.1-1]	客户端或服务端请求重发一个 PUBLISH 报文时，**必须**将 DUP 标志设置为 1。
[MQTT-3.3.1-2]	对于 QoS 0 的消息，DUP 标志**必须**设置为 0
[MQTT-3.3.1-3]	服务端发送 PUBLISH 报文给订阅者时，收到（入站）的 PUBLISH 报文的 DUP 标志的值不会被传播。发送（出站）的 PUBLISH 报文与收到（入站）的 PUBLISH 报文中的 DUP 标志是独立设置的，它的值**必须**单独的根据发送（出站）的 PUBLISH 报文是否是一个重发来确定。
[MQTT-3.3.1-4]	PUBLISH 报文**不能**将 QoS 所有的位设置为 1。如果服务端或客户端收到 QoS 所有位都为 1 的 PUBLISH 报文，它**必须**关闭网络连接。
[MQTT-3.3.1-5]	如果客户端发给服务端的 PUBLISH 报文的保留（RETAIN）标志被设置为 1，服务端**必须**存储这个应用消息和它的服务质量等级（QoS），以便它可以被分发给未来的主题名匹配的订阅者。
[MQTT-3.3.1-6]	一个新的订阅建立时，对每个匹配的主题名，如果存在最近保留的消息，它**必须**被发送给这个订阅者。
[MQTT-3.3.1-7]	如果服务端收到一条保留（RETAIN）标志为 1 的 QoS 0 消息，它**必须**丢弃之前为那个主题保留的任何消息。它**应该**将这个新的 QoS 0 消息当作那个主题的新保留消息，但是任何时候都**可以**选择丢弃它 — 如果这种情况发生了，那个主题将没有保留消息。
[MQTT-3.3.1-8]	服务端发送 PUBLISH 报文给客户端时，如果消息是作为客户端一个新订阅的结果发送，它**必须**将报文的保留标志设为 1。
[MQTT-3.3.1-9]	当一个 PUBLISH 报文发送给客户端是因为匹配一个已建立的订阅时，服务端**必须**将保留标志设为 0，不管它收到的这个消息中保留标志的值是多少。

[MQTT-3.3.1-10]	保留标志为 1 且有效载荷为零字节的 PUBLISH 报文会被服务端当作正常消息处理，它会被发送给订阅主题匹配的客户端。此外，同一个主题下任何现存的保留消息必须被移除，因此这个主题之后的任何订阅者都不会收到一个保留消息。
[MQTT-3.3.1-11]	服务端**不能**存储零字节的保留消息。
[MQTT-3.3.1-12]	如果客户端发给服务端的 PUBLISH 报文的保留标志位 0，服务端**不能**存储这个消息也**不能**移除或替换任何现存的保留消息。
[MQTT-3.3.2-1]	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。
[MQTT-3.3.2-2]	PUBLISH 报文中的主题名**不能**包含通配符。
[MQTT-3.3.2-3]	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）。
[MQTT-3.3.4-1]	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，见表格 3.4 的描述。
[MQTT-3.3.5-1]	服务端**必须**将消息分发给所有订阅匹配的 QoS 等级最高的客户端。
[MQTT-3.3.5-2]	如果服务端实现不授权某个客户端发布 PUBLISH 报文，它没有办法通知那个客户端。它**必须**按照正常的 QoS 规则发送一个正面的确认，或者关闭网络连接。
[MQTT-3.6.1-1]	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。
[MQTT-3.8.1-1]	SUBSCRIBE 控制报固定报头的第 3,2,1,0 位是保留位，**必须**分别设置为 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。
[MQTT-3.8.3-1]	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-3.8.3-2]	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-3.8.3-3]	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-3-8.3-4]	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。
[MQTT-3.8.4-1]	服务端收到客户端发送的一个 SUBSCRIBE 报文时，**必须**使用 SUBACK 报文响应。
[MQTT-3.8.4-2]	SUBACK 报文**必须**和等待确认的 SUBSCRIBE 报文有相同的报文标识符。
[MQTT-3.8.4-3]	如果服务端收到一个 SUBSCRIBE 报文，报文的主题过滤器与一个现存订阅的主题过滤器相同，那么**必须**使用新的订阅彻底替换现存的订阅。新订阅的主题过滤器和之前订阅的相同，但是它的最大 QoS 值可以不同。与这个主题过滤器匹配的任何现存的保留消息**必须**被重发，但是发布流程**不能**中断。
[MQTT-3.8.4-4]	如果服务端收到包含多个主题过滤器的 SUBSCRIBE 报文，它**必须**如同收到了一系列的多个 SUBSCRIBE 报文一样处理那个，除了需要将它们的响应合并到一个单独的 SUBACK 报文发送。

[MQTT-3.8.4-5]	服务端发送给客户端的 SUBACK 报文对每一对主题过滤器 和 QoS 等级都**必须**包含一个返回码。这个返回码**必须**表示那个订阅被授予的最大 QoS 等级，或者表示这个订阅失败。
[MQTT-3.8.4-6]	服务端可以授予比订阅者要求的低一些的 QoS 等级。为响应订阅而发出的消息的有效载荷的 QoS **必须**是原始发布消息的 QoS 和服务端授予的 QoS 两者中的最小值。如果原始消息的 QoS 是 1 而被授予的最大 QoS 是 0，允许服务端重复发送一个消息的副本给订阅者。
[MQTT-3.9.3-1]	返回码的顺序**必须**和 SUBSCRIBE 报文中主题过滤器的顺序相同。
[MQTT-3.9.3-2]	0x00, 0x01, 0x02, 0x80 之外的 SUBACK 返回码是保留的，**不能**使用。
[MQTT-3.10.1-1]	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。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。
[MQTT-3.10.3-1]	UNSUBSCRIBE 报文中的主题过滤器**必须**是连续打包的、按照 1.5.3 节定义的 UTF-8 编码字符串。
[MQTT-3.10.3-2]	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。
[MQTT-3.10.4-1]	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。
[MQTT-3.10.4-2]	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。
[MQTT-3.10.4-3]	如果服务端删除了一个订阅，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。
[MQTT-3.10.4-4]	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。
[MQTT-3.10.4-5]	即使没有删除任何主题订阅，服务端也**必须**发送一个 SUBACK 响应。
[MQTT-3.10.4-6]	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。
[MQTT-3.12.4-1]	服务端**必须**发送 PINGRESP 报文响应客户端的 PINGREQ 报文。
[MQTT-3.14.1-1]	服务端**必须**验证所有的保留位都被设置为 0，如果它们不为 0 **必须**断开连接。
[MQTT-3.14.4-1]	客户端发送 DISCONNECT 报文之后，**必须**关闭网络连接。
[MQTT-3.14.4-2]	客户端发送 DISCONNECT 报文之后，**不能**通过那个网络连接再发送任何控制报文。
[MQTT-3.14.4-3]	服务端收到 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，具体描述见 3.1.2.5 节。
[MQTT-4.1.0-1]	在整个会话期间，客户端和服务端都**必须**存储会话状态。
[MQTT-4.1.0-2]	会话**必须**至少持续和它的活跃网络连接同样长的时间。

[MQTT-4.3.1-1]	对于 QoS 0 的分发协议，发送者**必须**发送 QoS 等于 0，DUP 等于 0 的 PUBLISH 报文。
[MQTT-4.3.2-1]	对于 QoS 1 的分发协议，发送者 • 每次发送新的应用消息都**必须**分配一个未使用的报文标识符。 • MUST send a PUBLISH Packet containing this Packet Identifier with QoS=1, DUP=0. • 发送的 PUBLISH 报文**必须**包含报文标识符且 QoS 等于 1，DUP 等于 0。 • **必须**将这个 PUBLISH 报文看作是 *未确认的*，直到从接收者那收到对应的 PUBACK 报文。4.4 节有一个关于未确认消息的讨论。
[MQTT-4.3.2-2]	对于 QoS 1 的分发协议，接收者 • 响应的 PUBACK 报文**必须**包含一个报文标识符，这个标识符来自接收到的、已经接受所有权的 PUBLISH 报文。 • 发送了 PUBACK 报文之后，接收者必须将任何包含相同报文标识符的入站 PUBLISH 报文当作一个新的消息，并忽略它的 DUP 标志的值。
[MQTT-4.3.3-1]	对于 QoS 2 的分发协议，发送者 • 必须给要发送的新应用消息分配一个未使用的报文标识符。 • MUST send a PUBLISH packet containing this Packet Identifier with QoS=2, DUP=0. • 发送的 PUBLISH 报文**必须**包含报文标识符且报文的 QoS 等于 2,，DUP 等于 0。 • **必须**将这个 PUBLISH 报文看作是 *未确认的*，直到从接收者那收到对应的 PUBREC 报文。4.4 节有一个关于未确认消息的讨论。 • 收到 PUBREC 报文后**必须**发送一个 PUBREL 报文。PUBREL 报文必须包含与原始 PUBLISH 报文相同的报文标识符。 • MUST treat the PUBREL packet as “unacknowledged” until it has received the corresponding PUBCOMP packet from the receiver. • **必须**将这个 PUBREL 报文看作是 *未确认的*，直到从接收者那收到对应的 PUBCOMP 报文。 • 一旦发送了对应的 PUBREL 报文就**不能**重发这个 PUBLISH 报文。
[MQTT-4.3.3-2]	对于QoS 2的分发协议，接收者 • 响应的 PUBREC 报文**必须**包含报文标识符，这个标识符来自接收到的、已经接受所有权的 PUBLISH 报文。 • 在收到对应的 PUBREL 报文之前，接收者**必须**发送 PUBREC 报文确认任何后续的具有相同标识符的 PUBLISH 报文。 在这种情况下，它**不能**重复分发消息给任何后续的接收者。 • 响应 PUBREL 报文的 PUBCOMP 报文**必须**包含与 PUBREL 报文相同的标识符。 • 发送 PUBCOMP 报文之后，接收者必须将包含相同报文标识符的任何后续 PUBLISH 报文当作一个新的发布。

[MQTT-4.4.0-1]	客户端设置清理会话（CleanSession）标志为 0 重连时，客户端和服务端**必须**使用原始的报文标识符重发任何未确认的 PUBLISH 报文（如果 QoS>0）和 PUBREL 报文。
[MQTT-4.5.0-1]	服务端接管入站应用消息的所有权时，它**必须**将消息添加到订阅匹配的客户端的会话状态中。匹配规则定义见 4.7 节。
[MQTT-4.5.0-2]	客户端**必须**按照可用的服务质量（QoS）规则确认它收到的任何 PUBLISH 报文，不管它选择是否处理报文包含的应用消息。
[MQTT-4.6.0-1]	重发任何之前的 PUBLISH 报文时，**必须**按原始 PUBLISH 报文的发送顺序重发（适用于 QoS 1 和 QoS 2 消息）。
[MQTT-4.6.0-2]	**必须**按照对应的 PUBLISH 报文的顺序发送 PUBACK 报文（QoS 1 消息）。
[MQTT-4.6.0-3]	**必须**按照对应的 PUBLISH 报文的顺序发送 PUBREC 报文（QoS 2 消息）。
[MQTT-4.6.0-4]	**必须**按照对应的 PUBREC 报文的顺序发送 PUBREL 报文（QoS 2 消息）。
[MQTT-4.6.0-5]	服务端**必须**默认认为每个主题都是有序的。它**可以**提供一个管理功能或其它机制，以允许将一个或多个主题当作是**无序的**。
[MQTT-4.6.0-6]	服务端处理发送给有序主题的消息时，**必须**按照上面的规则将消息分发给每个订阅者。此外，它**必须**按照从客户端收到的顺序发送 PUBLISH 报文给消费者（对相同的主题和 QoS）。
[MQTT-4.7.1-1]	主题过滤器中可以使用通配符，但是主题名**不能**使用通配符。
[MQTT-4.7.1-2]	多层通配符必须位于它自己的层级或者跟在主题层级分隔符后面。不管哪种情况，它都**必须**是主题过滤器的最后一个字符。
[MQTT-4.7.1-3]	在主题过滤器的任意层级都可以使用单层通配符，包括第一个和最后一个层级。然而它**必须**占据过滤器的整个层级。
[MQTT-4.7.2-1]	服务端**不能**将 $ 字符开头的主题名匹配通配符 (#或+) 开头的主题过滤器。
[MQTT-4.7.3-1]	所有的主题名和主题过滤器**必须**至少包含一个字符。
[MQTT-4.7.3-2]	主题名和主题过滤器**不能**包含空字符 (Unicode U+0000) [Unicode] 。
[MQTT-4.7.3-3]	主题名和主题过滤器是 UTF-8 编码字符串，它们**不能**超过 65535 字节。
[MQTT-4.7.3-4]	匹配订阅时，服务端**不能**对主题名或主题过滤器执行任何规范化（normalization）处理，不能修改或替换任何未识别的字符。
[MQTT-4.8.0-1]	除非另有说明，如果服务端或客户端遇到了协议违规的行为，它**必须**关闭传输这个协议违规控制报文的网络连接。
[MQTT-4.8.0-2]	如果客户端或服务端处理入站控制报文时遇到了瞬时错误，它**必须**关闭传输那个控制报文的网络连接。
[MQTT-6.0.0-1]	MQTT 控制报文**必须**使用 WebSocket 二进制数据帧发送。如果收到任何其它类型的数据帧，接收者**必须**关闭网络连接。
[MQTT-6.0.0-2]	单个 WebSocket 数据帧可以包含多个或者部分 MQTT 报文。接收者**不能**假设 MQTT 控制报文按 WebSocket 帧边界对齐。

[MQTT-6.0.0-3]	客户端**必须**将字符串 **mqtt** 包含在它提供的 WebSocket 子协议列表里。
[MQTT-6.0.0-4]	服务端选择和返回的 WebSocket 子协议名**必须**是 **mqtt**
[MQTT-7.0.0-1]	MQTT 实现可以同时是 MQTT 客户端和 MQTT 服务端。接受入站连接和建立到其它服务端的出站连接的服务端必须同时符合 MQTT 客户端和 MQTT 服务端的要求。
[MQTT-7.0.0-2]	为了与任何其它的一致性实现交互操作，一致性实现不能要求使用在本规范之外定义的任何扩展。
[MQTT-7.1.1-1]	满足一致性要求的服务端**必须**支持使用一个或多个底层传输协议，只要它提供有序的、可靠的、双向字节流（从客户端到服务端和从服务端到客户端）
[MQTT-7.1.2-1]	满足一致性要求的客户端**必须**支持使用一个或多个底层传输协议，只要它提供有序的、可靠的、双向字节流（从客户端到服务端和从服务端到客户端）