MQTT 协议 5.0 中文版

委员会规范草案 02 /
公开评审草案 02

2017 年 10 月 26 日

规范链接

当前版本：

http://docs.oasis-open.org/mqtt/mqtt/v5.0/csprd02/mqtt-v5.0-csprd02.docx (Authoritative)
http://docs.oasis-open.org/mqtt/mqtt/v5.0/csprd02/mqtt-v5.0-csprd02.html
http://docs.oasis-open.org/mqtt/mqtt/v5.0/csprd02/mqtt-v5.0-csprd02.pdf

以前的版本：

http://docs.oasis-open.org/mqtt/mqtt/v5.0/csprd01/mqtt-v5.0-csprd01.docx (Authoritative)
http://docs.oasis-open.org/mqtt/mqtt/v5.0/csprd01/mqtt-v5.0-csprd01.html
http://docs.oasis-open.org/mqtt/mqtt/v5.0/csprd01/mqtt-v5.0-csprd01.pdf

最新版本：

http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.docx (Authoritative)
http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html
http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf

技术委员会：

OASIS Message Queuing Telemetry Transport (MQTT) TC

主席：

Brian Raymor (brian.raymor@microsoft.com), Microsoft
Richard Coppen (coppen@uk.ibm.com), IBM

编辑：

Andrew Banks (andrew_banks@uk.ibm.com), IBM
Ed Briggs (edbriggs@microsoft.com), Microsoft
Ken Borgendale (kwb@us.ibm.com), IBM
Rahul Gupta (rahul.gupta@us.ibm.com), IBM

相关文档：

本规范代替：

- MQTT 协议 3.1.1 版本。编辑是 Andrew Banks 和 Rahul Gupta，发布于 2014 年 10 月 29 日，OASIS 标准：http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.

本规范与此有关：

- MQTT 和 NIST 网络安全框架 1.0 版。 编辑是 Geoff Brown 和 Louis-Philippe Lamoureux。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，或其他提供了有序、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他修订版和其他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/ .

本规范草案的发布基于 OASIS 知识产权政策的 Non-Assertion 模式。关于实现本规范必不可少的任何专利是否已公开，以及其他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).

请注意，为本产品声明的任何机器可读内容（计算机语言定义）单独提供纯文本文件。对于产品叙述文档中出现的任何不一致，以纯文本文档为准。

引用格式：

引用此规范时应该使用下面的引文格式：

[mqtt-v5.0]

MQTT Version 5.0. Edited by Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. 26 October 2017. OASIS Committee Specification Draft 02 / Public Review Draft 02. http://docs.oasis-open.org/mqtt/mqtt/v5.0/csprd02/mqtt-v5.0-csprd02.html. Latest version: http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

文档链接

MQTT v5.0 草案公开审阅版中文翻译项目
MQTT v5.0 草案公开审阅版中文版 PDF

修订记录

版本	日期	发布说明
0.0.1	2018-05-18	发布全部文本，完成初步审校，公开发布第一版

关于译者

GitHub
Email

注意

目录

1 概述 12
1.0 知识产权政策 12
1.1 MQTT 协议的组织结构 12
1.2 术语 12
1.3 规范引用 14
1.4 非规范引用 14
1.5 数据表示 17
1.5.1 二进制位 17
1.5.2 双字节整数 17
1.5.3 四字节整数 17
1.5.4 UTF-8 编码字符串 17
1.5.5 变长字节整数 19
1.5.6 二进制数据 20
1.5.7 UTF-8 字符串对 20
1.6 安全 20
1.7 编辑约定 20
1.8 变更历史 20
1.8.1 MQTT v3.1.1 20
1.8.2 MQTT v5.0 20
2 MQTT 控制报文格式 22
2.1 MQTT 控制报文结构 22
2.1.1 固定报头 22
2.1.2 MQTT 控制报文的类型 22
2.1.3 标志 23
2.1.4 剩余长度 24
2.2 可变报头 24
2.2.1 报文标识符 24
2.2.2 属性 26
2.2.2.1 属性长度 26
2.2.2.2 属性 26
2.3 有效载荷 27
2.4 原因码 28
3 MQTT 控制报文 31
3.1 CONNECT – 连接请求 31
3.1.1 CONNECT 固定报头 31
3.1.2 CONNECT 可变报头 31
3.1.2.1 协议名 31
3.1.2.2 协议版本 32
3.1.2.3 连接标志 32

3.1.2.4 新开始 33
3.1.2.5 遗嘱标志 33
3.1.2.6 遗嘱 QoS 34
3.1.2.7 遗嘱保留 34
3.1.2.8 用户名标志 34
3.1.2.9 密码标志 34
3.1.2.10 保持连接 35
3.1.2.11 CONNECT 属性 35
3.1.2.11.1 属性长度 35
3.1.2.11.2 会话过期间隔 36
3.1.2.11.3 接收最大值 37
3.1.2.11.4 最大报文长度 37
3.1.2.11.5 主题别名最大值 38
3.1.2.11.6 请求响应信息 38
3.1.2.11.7 请求问题信息 38
3.1.2.11.8 用户属性 39
3.1.2.11.9 认证方法 39
3.1.2.11.10 认证数据 39
3.1.2.12 可变报头非规范示例 40
3.1.3 CONNECT 载荷 41
3.1.3.1 客户标识符 41
3.1.3.2 遗嘱属性 41
3.1.3.2.1 属性长度 42
3.1.3.2.2 遗嘱延时间隔 42
3.1.3.2.3 载荷格式指示 42
3.1.3.2.4 消息过期间隔 42
3.1.3.2.5 内容类型 43
3.1.3.2.6 响应主题 43
3.1.3.2.7 对比数据 43
3.1.3.2.8 用户属性 43
3.1.3.3 遗嘱主题 44
3.1.3.4 遗嘱载荷 44
3.1.3.5 用户名 44
3.1.3.6 密码 44
3.1.4 CONNECT 行为 44
3.2 CONNACK – 确认连接请求 45
3.2.1 CONNACK 固定报头 46
3.2.2 CONNACK 可变报头 46
3.2.2.1 连接确认标志 46
3.2.2.1.1 会话存在 46
3.2.2.2 连接原因码 47
3.2.2.3 CONNACK 属性 48
3.2.2.3.1 属性长度 48
3.2.2.3.2 会话过期间隔 48

3.2.2.3.3 接收最大值 48
3.2.2.3.4 最大服务质量 49
3.2.2.3.5 保留可用 49
3.2.2.3.6 最大报文长度 50
3.2.2.3.7 分配客户标识符 50
3.2.2.3.8 主题别名最大值 50
3.2.2.3.9 原因字符串 50
3.2.2.3.10 用户属性 51
3.2.2.3.11 通配符订阅可用 51
3.2.2.3.12 订阅标识符可用 51
3.2.2.3.13 共享订阅可用 52
3.2.2.3.14 服务端保持连接 52
3.2.2.3.15 响应信息 52
3.2.2.3.16 服务端参考 53
3.2.2.3.17 认证方法 53
3.2.2.3.18 认证数据 53
3.2.3 CONNACK 载荷 53
3.3 PUBLISH – 发布消息 53
3.3.1 PUBLISH 固定报头 53
3.3.1.1 重发标志 54
3.3.1.2 服务质量等级 54
3.3.1.3 保留标志 55
3.3.1.4 剩余长度 56
3.3.2 PUBLISH 可变报头 56
3.3.2.1 主题名 56
3.3.2.2 报文标识符 56
3.3.2.3 PUBLISH 属性 56
3.3.2.3.1 属性长度 56
3.3.2.3.2 载荷格式指示 56
3.3.2.3.3 消息过期间隔 57
3.3.2.3.4 主题别名 57
3.3.2.3.5 响应主题 58
3.3.2.3.6 对比数据 58
3.3.2.3.7 用户属性 59
3.3.2.3.8 订阅标识符 59
3.3.2.3.9 内容类型 59
3.3.3 PUBLISH 载荷 60
3.3.4 PUBLISH 行为 60
3.4 PUBACK – 发布确认 62
3.4.1 PUBACK 固定报头 62
3.4.2 PUBACK 可变报头 63
3.4.2.1 PUBACK 原因码 63
3.4.2.2 PUBACK 属性 64
3.4.2.2.1 属性长度 64

3.4.2.2.2 原因字符串 64
3.4.2.2.3 用户属性 64
3.4.3 PUBACK 载荷 64
3.4.4 PUBACK 行为 64
3.5 PUBREC – 发布已接收（QoS 2，第一步） 65
3.5.1 PUBREC 固定报头 65
3.5.2 PUBREC 可变报头 65
3.5.2.1 PUBREC 原因码 65
3.5.2.2 PUBREC 属性 66
3.5.2.2.1 属性长度 66
3.5.2.2.2 原因字符串 66
3.5.2.2.3 用户属性 66
3.5.3 PUBREC 载荷 67
3.5.4 PUBREC 行为 67
3.6 PUBREL – 发布释放（QoS 2，第二步） 67
3.6.1 PUBREL 固定报头 67
3.6.2 PUBREL 可变报头 67
3.6.2.1 PUBREL 原因码 68
3.6.2.2 PUBREL 属性 68
3.6.2.2.1 属性长度 68
3.6.2.2.2 原因字符串 68
3.6.2.2.3 用户属性 68
3.6.3 PUBREL 载荷 68
3.6.4 PUBREL 行为 69
3.7 PUBCOMP – 发布完成（QoS 2，第三步） 69
3.7.1 PUBCOMP 固定报头 69
3.7.2 PUBCOMP 可变报头 69
3.7.2.1 PUBCOMP 原因码 69
3.7.2.2 PUBCOMP 属性 70
3.7.2.2.1 属性长度 70
3.7.2.2.2 原因字符串 70
3.7.2.2.3 用户属性 70
3.7.3 PUBCOMP 载荷 70
3.7.4 PUBCOMP 行为 70
3.8 SUBSCRIBE - 订阅请求 71
3.8.1 SUBSCRIBE 固定报头 71
3.8.2 SUBSCRIBE 可变报头 71
3.8.2.1 SUBSCRIBE 属性 72
3.8.2.1.1 属性长度 72
3.8.2.1.2 订阅标识符 72
3.8.2.1.3 用户属性 72
3.8.3 SUBSCRIBE 载荷 72
3.8.3.1 订阅选项 72

3.8.4 SUBSCRIBE 行为 74
3.9 SUBACK – 订阅确认 76
3.9.1 SUBACK 固定报头 76
3.9.2 SUBACK 可变报头 77
3.9.2.1 SUBACK 属性 77
3.9.2.1.1 属性长度 77
3.9.2.1.2 原因字符串 77
3.9.2.1.3 用户属性 77
3.9.3 SUBACK 载荷 78
3.10 UNSUBSCRIBE – 取消订阅请求 78
3.10.1 UNSUBSCRIBE 固定报头 78
3.10.2 UNSUBSCRIBE 可变报头 79
3.10.2.1 UNSUBSCRIBE 属性 79
3.10.2.1.1 属性长度 79
3.10.2.1.2 用户属性 79
3.10.3 UNSUBSCRIBE 载荷 79
3.10.4 UNSUBSCRIBE 行为 80
3.11 UNSUBACK – 取消订阅确认 81
3.11.1 UNSUBACK 固定报头 81
3.11.2 UNSUBACK 可变报头 81
3.11.2.1 UNSUBACK 属性 81
3.11.2.1.1 属性长度 81
3.11.2.1.2 原因字符串 81
3.11.2.1.3 用户属性 82
3.11.3 UNSUBACK 载荷 82
3.12 PINGREQ – PING 请求 82
3.12.1 PINGREQ 固定报头 83
3.12.2 PINGREQ 可变报头 83
3.12.3 PINGREQ 载荷 83
3.12.4 PINGREQ 行为 83
3.13 PINGRESP – PING 响应 83
3.13.1 PINGRESP 固定报头 83
3.13.2 PINGRESP 可变报头 84
3.13.3 PINGRESP 载荷 84
3.13.4 PINGRESP 行为 84
3.14 DISCONNECT – 断开通知 84
3.14.1 DISCONNECT 固定报头 84
3.14.2 DISCONNECT 可变报头 84
3.14.2.1 断开原因码 85
3.14.2.2 DISCONNECT 属性 86
3.14.2.2.1 属性长度 86
3.14.2.2.2 会话过期间隔 87

3.14.2.2.3 原因字符串 87
3.14.2.2.4 用户属性 87
3.14.2.2.5 服务端参考 87
3.14.3 DISCONNECT 载荷 88
3.14.4 DISCONNECT 行为 88
3.15 AUTH – 认证交换 88
3.15.1 AUTH 固定报头 88
3.15.2 AUTH 可变报头 89
3.15.2.1 认证原因码 89
3.15.2.2 AUTH 属性 89
3.15.2.2.1 属性长度 89
3.15.2.2.2 认证方法 89
3.15.2.2.3 认证数据 90
3.15.2.2.4 原因字符串 90
3.15.2.2.5 用户属性 90
3.15.3 AUTH 载荷 90
3.15.4 AUTH 行为 90
4 操作行为 91
4.1 会话状态 91
4.1.1 存储会话状态 91
4.1.2 会话状态非规范示例 91
4.2 网络连接 92
4.3 服务质量等级和协议流程 92
4.3.1 QoS 0：最多分发一次 92
4.3.2 QoS 1：至少分发一次 93
4.3.3 QoS 2：仅分发一次 94
4.4 消息分发重试 95
4.5 消息收到 96
4.6 消息排序 96
4.7 主题名和主题过滤器 96
4.7.1 主题通配符 96
4.7.1.1 主题层级分隔符 97
4.7.1.2 多层通配符 97
4.7.1.3 单层通配符 97
4.7.2 以$开头的主题 98
4.7.3 主题语义和用法 98
4.8 订阅 99
4.8.1 非共享订阅 99
4.8.2 共享订阅 99
4.9 流控 101
4.10 请求/响应 102
4.10.1 基本请求响应（非规范） 102

4.10.2 确定响应主题值（非规范） 103
4.11 服务端重定向 103
4.12 增强认证 104
4.12.1 重新认证 105
4.13 错误处理 106
4.13.1 无效报文和协议错误 106
4.13.2 其他错误 107
5 安全（非规范） 108
5.1 概述 108
5.2 MQTT 解决方案：安全和认证 108
5.3 轻量级的加密与受限设备 109
5.4 实现注意事项 109
5.4.1 客户端身份认证 109
5.4.2 客户端授权 109
5.4.3 服务端身份认证 110
5.4.4 应用消息和 MQTT 控制报文的完整性 110
5.4.5 应用消息和 MQTT 控制报文的保密性 110
5.4.6 消息传输的不可否认性 110
5.4.7 客户端和服务端盗用检测 111
5.4.8 异常行为检测 111
5.4.9 其它安全注意事项 111
5.4.10 使用 SOCK 代理 112
5.4.11 安全配置文件 112
5.4.11.1 开放通信配置 112
5.4.11.2 安全网络通信配置 112
5.4.11.3 安全传输配置 112
5.4.11.4 工业标准的安全配置 112
6 使用 WebSocket 作为网络层 114
6.1 IANA 注意事项 114
7 一致性 115
7.1 一致性条款 115
7.1.1 MQTT 服务端一致性条款 115
7.1.2 MQTT 客户端一致性条款 115
Appendix A. 致谢 116
Appendix B. 强制性规范声明（非规范） 117
Appendix C. MQTT v5.0 新特性总结（非规范） 131

1 概述

1.0 知识产权政策

此公开评审草案的发布基于 OASIS IPR Policy 的 Non-Assertion 模式。

关于实现本规范必不可少的任何专利是否已公开，以及其他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）。

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
- 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：

- 打开连接到服务端的网络连接
- 发布应用消息给其他相关的客户端
- 订阅以请求接受相关的应用消息
- 取消订阅以移除接受相应消息的请求
- 关闭连接到服务端的网络连接

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
- 关闭来自客户端的网络连接

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。

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。

共享订阅（Shared Subscription）：

一个共享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。一个共享订阅可以与多个订阅会话相关联，便于支持大范围消息交换模式。一条主题匹配的应用消息只发送给关联到此共享订阅的多个会话中的一个会话。一个会话可以包括多个共享订阅，可以同时包含共享订阅与非共享订阅。

通配符订阅（Wildcard Subscription）：

通配符订阅是指主题过滤器（Topic Filter）包含一个或多个通配符的订阅。通配符订阅使得一次订阅匹配多个主题名（Topic Name）。4.7 节 描述了主题过滤器中的通配符。

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。

MQTT 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 MQTT 控制报文，其中一个（PUBLISH 报文）用于传输应用消息。

无效报文（Malformed Packet）：

根据规范不能被正确解析的控制报文。4.13 节 描述了如何进行相应的错误处理。

协议错误（Protocol Error）：

在报文解析之后发现包含协议不允许或与客户端或服务端当前状态不一致的数据的错误。4.13 节 描述了如何进行相应的错误处理。

遗嘱消息（Will Message）：

在网络连接非正常关闭的情况下，由服务端发布的应用消息。3.1.2.5 节 描述了遗嘱消息。

1.3 规范引用

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997,

http://www.rfc-editor.org/info/rfc2119

[RFC3629]

Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003,

http://www.rfc-editor.org/info/rfc3629

[RFC6455]

Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011,

http://www.rfc-editor.org/info/rfc6455

[Unicode]

The Unicode Consortium. The Unicode Standard,

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

1.4 非规范引用

[RFC0793]

Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, http://www.rfc-editor.org/info/rfc793

[RFC5246]

Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008,

http://www.rfc-editor.org/info/rfc5246

[AES]

Advanced Encryption Standard (AES) (FIPS PUB 197).

https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf

[CHACHA20]

ChaCha20 and Poly1305 for IETF Protocols
https://tools.ietf.org/html/rfc7539

[FIPS1402]
Security Requirements for Cryptographic Modules (FIPS PUB 140-2)
https://csrc.nist.gov/csrc/media/publications/fips/140/2/final/documents/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
https://www.iso.org/standard/56425.html

[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

[MQTTV311]
MQTT V3.1.1 Protocol Specification
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[ISO20922]
MQTT V3.1.1 ISO Standard (ISO/IEC 20922:2016)
https://www.iso.org/standard/69466.html

[NISTCSF]
Improving Critical Infrastructure Cybersecurity Executive Order 13636
https://www.nist.gov/sites/default/files/documents/itl/preliminary-cybersecurity-framework.pdf

[NIST7628]
NISTIR 7628 Guidelines for Smart Grid Cyber Security Catalogue
https://www.nist.gov/sites/default/files/documents/smartgrid/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/pci_security/

[RFC1928]

Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, DOI 10.17487/RFC1928, March 1996,

http://www.rfc-editor.org/info/rfc1928

[RFC4511]

Sermersheim, J., Ed., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, DOI 10.17487/RFC4511, June 2006,

http://www.rfc-editor.org/info/rfc4511

[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, DOI 10.17487/RFC5280, May 2008,

http://www.rfc-editor.org/info/rfc5280

[RFC6066]

Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011,

http://www.rfc-editor.org/info/rfc6066

[RFC6749]

Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012,

http://www.rfc-editor.org/info/rfc6749

[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, DOI 10.17487/RFC6960, June 2013,

http://www.rfc-editor.org/info/rfc6960

[SARBANES]

Sarbanes-Oxley Act of 2002.

http://www.gpo.gov/fdsys/pkg/PLAW-107publ204/html/PLAW-107publ204.htm

[USEUPRIVSH]

U.S.-EU Privacy Shield Framework

https://www.privacyshield.gov

[RFC3986]

Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005,

http://www.rfc-editor.org/info/rfc3986

[RFC1035]

Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987,

http://www.rfc-editor.org/info/rfc1035

[RFC2782]

Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000,

http://www.rfc-editor.org/info/rfc2782

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 双字节整数

双字节整数是 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 四字节整数

四字节整数是 32 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 32 位的字在网络上表示为第一个最高有效字节（MSB）后面跟着第一个最低有效字节（LSB），再后面为第二个最高有效字节（MSB）后面跟着第二个最低有效字节（LSB）。

1.5.4 UTF-8 编码字符串

后续描述的 MQTT 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 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在 0 到 65535 字节这个范围内。

图 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 MQTT 控制报文，它**必须**关闭网络连接 [MQTT-1.5.4-1]。

UTF-8 编码的字符串**不能**包含空字符 U+0000 [MQTT-1.5.4-2]。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。

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 MQTT 控制报文，它**可以**把此报文当做无效报文。

- 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)

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-1.5.4-3]。

非规范示例

例如，字符串 A 龘 是一个大写拉丁字母 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 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

1.5.5 变长字节整数

剩余长度字段使用一个变长字节编码方案，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 [MQTT-1.5.5-1]，如表 1-1 所示。

表 1-1 - 变长字节整数大小

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)

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
```

```
    if (multiplier > 128*128*128)
      throw Error(Malformed Variable Byte Integer)
    multiplier *= 128
  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。

1.5.6 二进制数据

二进制数据由一个双字节整数指示其数据长度，因此，二进制数据的长度被限制为 0 到 65,535 字节。

1.5.7 UTF-8 字符串对

UTF-8 字符串对由两个 UTF-8 编码的字符串组成，用来表示名字-值对，第一个字符串表示名字，第二个字符串表示值。

所有的字符串**必须**遵循 UTF-8 字符串编码规范 [MQTT-1.5.7-1]。如果接受者（客户端或者服务端）接受到一个字符串对，然而其编码并不遵循规范，则此报文为无效报文。4.13 节描述了错误处理的信息。

1.6 安全

MQTT 客户端和服务端实现**应该**提供认证、授权和安全通信功能，如 第 5 章 所描述。强烈建议任何关注于个人身份信息或敏感信息的应用使用这些安全设施。

1.7 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]。

1.8 变更历史

1.8.1 MQTT v3.1.1

MQTT v3.1.1 是首个 OASIS 标准版本 MQTT **[MQTTV311]**。

MQTT v3.1.1 也是 ISO/IEC 20922:2016 **[ISO20922]** 标准。

1.8.2 MQTT v5.0

MQTT v5.0 在保持 MQTT 核心不变的基础上添加了大量的新功能。这些功能的主要目标如下：

- 进一步支持大规模可扩展系统
- 改进的错误报告 Improved error reporting
- 规范化包括容量探索和请求响应在内的通用模式

- 包括用户属性在内的可扩展机制
- 改进性能并支持小型客户端

附录 C 对 MQTT v5.0 的改进做出了总结。

2 MQTT 控制报文格式

2.1 MQTT 控制报文结构

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，按照下图描述的顺序。

图 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

2.1.1 固定报头

如下图所示，每个 MQTT 控制报文都包含一个固定报头。

图 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							

2.1.2 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，这些值的定义见下表。

表 2-1 - MQTT 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（保证交付第三步）
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	两个方向都允许	断开连接通知
AUTH	15	两个方向都允许	认证信息交换

2.1.3 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如下表所示。表格中任何标记为“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-2.1.3-1]。如果收到非法的标志，此报文被当做无效报文。有关错误处理的详细信息见 4.13 节。

表格 2-2 - 标志位

MQTT 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 v5.0	DUP	QoS		RETAIN
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
AUTH	Reserved	0	0	0	0

DUP = PUBLISH 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

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

2.1.4 剩余长度

位置：从第 2 个字节开始。

剩余长度（Remaining Length）是一个变长字节整数，用来表示当前控制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。MQTT 控制报文总长度等于固定报头的长度加上剩余长度。

2.2 可变报头

某些 MQTT 控制报文包含一个可变报头部分。它在固定报头和有效载荷之间。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。

2.2.1 报文标识符

部分类型 MQTT 控制报文的可变报头部分包含了 2 个字节的报文标识符字段。这些 MQTT 控制报文类型为：PUBLISH 报文（当 QoS>0 时），PUBACK，PUBREC，PUBREC，PUBREL，PUBCOMP，SUBSCRIBE，SUBACK，UNSUBSCRIBE，UNSUBACK。

需要报文标识符的 MQTT 控制报文如下表所示。

表 2-3 - 包含报文标识符的 MQTT 控制报文

MQTT 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
AUTH	不需要

QoS 设置为 0 的 PUBLISH 报文**不能**包含报文标识符[MQTT-2.2.1-2]。

客户端每次发送一个新的 SUBSCRIBE，UNSUBSCRIBE 或者 PUBLISH（当 QoS>0 时）MQTT 控制报文时都**必须**分配一个当前未使用的非零报文标识符 [MQTT-2.2.1-3]。

服务端每次发送一个新的 PUBLISH（当 QoS>0）MQTT 控制报文时都**必须**分配一个当前未使用的非零报文标识符 [MQTT-2.2.1-4]。

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 对应的是包含原因码 128 以上的 PUBCOMP 或 PUBREC，与 SUBSCRIBE 或 UNSUBSCRIBE 对应的分别是 SUBACK 或 UNSUBACK。

PUBLISH，SUBSCRIBE 和 UNSUBSCRIBE 的报文标识符，在一次会话中对于客户端和服务端来说分属于不同的组。某个报文标识符在某一时刻**不能**被多个命令所使用。

PUBACK，PUBREC 和 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-2.2.1-5]。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-2.2.1-6]。

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。

客户端	服务端
PUBLISH报文标识符 = 0x1234 ---->	
	<---- PUBLISH报文标识符 = 0x1234
PUBACK报文标识符 = 0x1234 ---->	
	<---- PUBACK报文标识符 = 0x1234

2.2.2 属性

CONNECT，CONNACK，PUBLISH，PUBACK，PUBREC，PUBREL，PUBCOMP，SUBSCRIBE，SUBACK，UNSUBACK，DISCONNECT 和 AUTH 报文可变报头的最后一部分是一组属性。CONNECT 报文的遗嘱（Will）属性字段中也包含了一组可选的属性。

属性字段由属性长度和所有属性组成。

2.2.2.1 属性长度

属性长度被编码为变长字节整数。属性长度不包含用于编码属性长度自身的字节数，但包含所有属性的长度。如果没有任何属性，**必须**由属性长度为零的字段来指示 [MQTT-2.2.2-1]。

2.2.2.2 属性

一个属性包含一段数据和一个定义了属性用途和数据类型的标识符。标识符被编码为变长字节整数。任何控制报文，如果包含了：对于该报文类型无效的标识符，或者错误类型的数据，都是无效报文。收到无效报文时，服务端或客户端使用包含原因码 0x81（无效报文）CONNACK 或 DISCONNECT 报文进行错误处理，如 4.13 节 所述。标识符排序不分先后。

表 2-4 - 属性

标识符		属性名 （用途）	数据类型	报文/遗嘱属性
Dec	**Hex**			
1	0x01	载荷格式说明	字节	PUBLISH, Will Properties
2	0x02	消息过期时间	四字节整数	PUBLISH, Will Properties
3	0x03	内容类型	UTF-8 编码字符串	PUBLISH, Will Properties
8	0x08	响应主题	UTF-8 编码字符串	PUBLISH, Will Properties
9	0x09	相关数据	二进制数据	PUBLISH, Will Properties
11	0x0B	定义标识符	变长字节整数	PUBLISH, SUBSCRIBE
17	0x11	会话过期间隔	四字节整数	CONNECT, CONNACK, DISCONNECT
18	0x12	分配客户标识符	UTF-8 编码字符串	CONNACK
19	0x13	服务端保活时间	双字节整数	CONNACK
21	0x15	认证方法	UTF-8 编码字符串	CONNECT, CONNACK, AUTH
22	0x16	认证数据	二进制数据	CONNECT, CONNACK, AUTH
23	0x17	请求问题信息	字节	CONNECT
24	0x18	遗嘱延时间隔	四字节整数	Will Properties
25	0x19	请求响应信息	字节	CONNECT

26	0x1A	请求信息	UTF-8 编码字符串	CONNACK
28	0x1C	服务端参考	UTF-8 编码字符串	CONNACK, DISCONNECT
31	0x1F	原因字符串	UTF-8 编码字符串	CONNACK, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBACK, UNSUBACK, DISCONNECT, AUTH
33	0x21	接收最大数量	双字节整数	CONNECT, CONNACK
34	0x22	主题别名最大长度	双字节整数	CONNECT, CONNACK
35	0x23	主题别名	双字节整数	PUBLISH
36	0x24	最大 QoS	字节	CONNACK
37	0x25	保留属性可用性	字节	CONNACK
38	0x26	用户属性	UTF-8 字符串对	CONNECT, CONNACK, PUBLISH, Will Properties, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK, DISCONNECT, AUTH
39	0x27	最大报文长度	四字节整数	CONNECT, CONNACK
40	0x28	通配符订阅可用性	字节	CONNACK
41	0x29	订阅标识符可用性	字节	CONNACK
42	0x2A	共享订阅可用性	字节	CONNACK

非规范评注

尽管属性标识符用变长字节整数来表示，但在此版本协议中，所有的标识符均由一个字节来表示。

2.3 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 2-5 - 包含有效载荷的 MQTT 控制报文

MQTT 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	需要
PINGREQ	不需要
PINGRESP	不需要
DISCONNECT	不需要
AUTH	不需要

2.4 原因码

原因码是一个单字节无符号数，用来指示一次操作的结果。小于 0x80 的原因码指示某次操作成功完成，通常用 0 来表示。大于等于 0x80 的原因码用来指示操作失败。

CONNACK，PUBACK，PUBREC，PUBREL，PUBCOMP，DISCONNECT 和 AUTH 控制报文的可变报头有一个单字节的原因码。SUBACK 和 UNSUBACK 报文的载荷字段包含一个或多个原因码。

原因码如下表所示。

表 2-6 - 原因码

原因码		名称	报文
Decimal	**Hex**		
0	0x00	成功	CONNACK, PUBACK, PUBREC, PUBREL, PUBCOMP, UNSUBACK, AUTH
0	0x00	正常断开	DISCONNECT
0	0x00	授权的 QoS 0	SUBACK
1	0x01	授权的 QoS 1	SUBACK
2	0x02	授权的 QoS 2	SUBACK
4	0x04	包含遗嘱的断开	DISCONNECT
16	0x10	无匹配订阅	PUBACK, PUBREC
17	0x11	订阅不存在	UNSUBACK
24	0x18	继续认证	AUTH

25	0x19	重新认证	AUTH
128	0x80	未指明的错误	CONNACK, PUBACK, PUBREC, SUBACK, UNSUBACK, DISCONNECT
129	0x81	无效报文	CONNACK, DISCONNECT
130	0x82	协议错误	CONNACK, DISCONNECT
131	0x83	实现错误	CONNACK, PUBACK, PUBREC, SUBACK, UNSUBACK, DISCONNECT
132	0x84	协议版本不支持	CONNACK
133	0x85	客户标识符无效	CONNACK
134	0x86	用户名密码错误	CONNACK
135	0x87	未授权	CONNACK, PUBACK, PUBREC, SUBACK, UNSUBACK, DISCONNECT
136	0x88	服务端不可用	CONNACK
137	0x89	服务端正忙	CONNACK, DISCONNECT
138	0x8A	禁止	CONNACK
139	0x8B	服务端关闭中	DISCONNECT
140	0x8C	无效的认证方法	CONNACK, DISCONNECT
141	0x8D	保活超时	DISCONNECT
142	0x8E	会话被接管	DISCONNECT
143	0x8F	主题过滤器无效	SUBACK, UNSUBACK, DISCONNECT
144	0x90	主题名无效	CONNACK, PUBACK, PUBREC, DISCONNECT
145	0x91	报文标识符已被占用	PUBACK, PUBREC, SUBACK, UNSUBACK
146	0x92	报文标识符无效	PUBREL, PUBCOMP
147	0x93	接收超出最大数量	DISCONNECT
148	0x94	主题别名无效	DISCONNECT
149	0x95	报文过长	CONNACK, DISCONNECT
150	0x96	消息太过频繁	DISCONNECT
151	0x97	超出配额	CONNACK, PUBACK, PUBREC, SUBACK, DISCONNECT
152	0x98	管理行为	DISCONNECT
153	0x99	载荷格式无效	CONNACK, PUBACK, PUBREC, DISCONNECT
154	0x9A	不支持保留	CONNACK, DISCONNECT

155	0x9B	不支持的 QoS 等级	CONNACK, DISCONNECT
156	0x9C	（临时）使用其他服务端	CONNACK, DISCONNECT
157	0x9D	服务端已（永久）移动	CONNACK, DISCONNECT
158	0x9E	不支持共享订阅	SUBACK, DISCONNECT
159	0x9F	超出连接速率限制	CONNACK, DISCONNECT
160	0xA0	最大连接时间	DISCONNECT
161	0xA1	不支持订阅标识符	SUBACK, DISCONNECT
162	0xA2	不支持通配符订阅	SUBACK, DISCONNECT

非规范评注

对于原因码 0x91（报文标识符已被占用）的处理可以为尝试修复会话、以新会话标志为 1 重置会话或者判定客户端或服务端实现有缺陷。

3 MQTT 控制报文

3.1 CONNECT – 连接请求

客户端到服务端的网络连接建立后，客户端发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 4.13 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 CONNECT 固定报头

图 3-1 - CONNECT 报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 报文类型 (1)				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加上有效载荷的长度。编码方式为变长字节整数。

3.1.2 CONNECT 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），保持连接（Keep Alive）和属性（Properties）。2.2.2 节 描述了属性（Properties）编码规则。

3.1.2.1 协议名

图 3-2 - 协议名字节

	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。

支持多种协议的服务端使用协议名字段判断数据是否为 MQTT 报文。协议名**必须**是 UTF-8 字符串“MQTT”。如果服务端不愿意接受 CONNECT 但希望表明其 MQTT 服务端身份，**可以**发送包含原因码为 0x84（不支持的协议版本）的 CONNACK 报文，然后**必须**关闭网络连接 [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 协议版本

图 3-3 - 协议版本字节

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

客户端使用一个字节无符号数表示协议修订级别。MQTT v5.0 的协议版本字段为 5（0x05）。

支持多版本 MQTT 协议的服务端使用 *协议版本* 字段判定客户端正使用的 MQTT 协议版本。如果协议版本不是 5 且服务端不愿意接受此 CONNECT 报文，**可以**发送包含原因码 0x84（不支持的协议版本）的 CONNACK 报文，然后**必须**关闭网络连接 [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 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 Start	Reserved
byte 8	X	X	X	X	X	X	X	0

服务端**必须**验证 CONNECT 报文的保留标志位（第 0 位）是否为 0 [MQTT-3.1.2-3]，如果不为 0 则此报文为无效报文。4.13 节 给出了错误处理信息。

3.1.2.4 新开始

位置：连接标志字节的第 1 位

这个二进制位表明此次连接是一个新的会话还是一个已存在的会话的延续。4.1 节 定义了会话状态。

如果收到新开始（Clean Start）为 1 的 CONNECT 报文，客户端和服务端**必须**丢弃任何已存在的会话，并开始一个新的会话 [MQTT-3.1.2-4]。相应的，CONNACK 报文中的会话存在标志设置为 0。

如果收到新开始（Clean Start）为 0 的 CONNECT 报文，并且存在一个关联此客户标识符的会话，服务端**必须**基于此会话的状态恢复与客户端的通信 [MQTT-3.1.2-5]。如果收到新开始（Clean Start）为 0 的 CONNECT 报文，并且不存在任何关联此客户标识符的会话，服务端**必须**创建一个新的会话 [MQTT-3.1.2-6]。

3.1.2.5 遗嘱标志

位置：连接标志字节的第 2 位

如果遗嘱标志（Will Flag）被设置为 1，表示遗嘱消息**必须**已存储在服务端与此客户标识符相关的会话中 [MQTT-3.1.2-7]。遗嘱消息（Will Message）包含遗嘱属性，遗嘱主题和遗嘱载荷字段。遗嘱**必须**在网络连接被关闭、遗嘱延时间隔到期或者会话结束之后被发布，除非服务端收到包含原因码为 0x00（正常关闭）的 DISCONNECT 报文之后删除了遗嘱消息（Will Message），或者一个关于此客户标识符的新的网络连接在遗嘱迟发时间（Will Delay Interval）超时之前被创建 [MQTT-3.1.2-8]。

遗嘱发布的条件，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
- 客户端在没有发送包含原因码 0x00（正常关闭）的情况下关闭了网络连接
- 服务端在没有收到包含原因码 0x00（正常关闭）的情况下关闭了网络连接

如果遗嘱标志（Will Flag）被设置为 1，遗嘱属性（Will Property）、遗嘱主题（Will Topic）和遗嘱载荷（Will Payload）字段**必须**存在于报文有效载荷中 [MQTT-3.1.2-9]。一旦遗嘱消息（Will Message）被发布或者服务端收到包含原因码为 0x00（正常关闭）的 DISCONNECT 报文，遗嘱消息（Will Message）**必须**从服务端的会话中删除 [MQTT-3.1.2-10]。

服务端**应该**在网络连接断开并且遗嘱迟发时间（Will Delay Interval）到期，或者会话结束之后立即发布遗嘱消息。服务端关闭或出错的情况下，**可以**在服务重新启动之后发布遗嘱消息（Will Message）。这种情况下从服务端出错到遗嘱发布之间存在一定的延迟。

关于遗嘱延时间隔（Will Delay Interval）的详细信息，请参考 3.1.3.2 节。

非规范评注

通过设置晚于会话过期间隔（Session Expiry Interval）的遗嘱迟发时间（Will Delay Interval）并发送包含原因码 0x04（包含遗嘱的断开连接），客户端得以发出会话过期（Session Expiry）通告。

3.1.2.6 遗嘱 QoS

位置：连接标志字节的第 3、4 位

这两个比特指定了发布遗嘱消息（Will Message）时的服务质量（QoS）。

如果遗嘱标志（Will Flag）设置为 0，遗嘱服务质量（Will QoS）**必须**也设置为 0（0x00） [MQTT-3.1.2-11]。

如果遗嘱标志设置为 1，遗嘱服务质量**可以**被设置为 0（0x00），1（0x01）或 2（0x02） [MQTT-3.1.2-12]。设置为 3（0x03）的报文是无效报文。4.13 节 描述了错误处理信息。

3.1.2.7 遗嘱保留

位置：连接标志字节的第 5 位

此位指定遗嘱消息（Will Message）在发布时是否会被保留。

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-3.1.2-13]。如果遗嘱标志被设置为 1 时，如果遗嘱保留被设置为 0，则服务端**必须**将遗嘱消息当做非保留消息发布 [MQTT-3.1.2-14]。如果遗嘱保留被设置为 1，则服务端**必须**将遗嘱消息当做保留消息发布 [MQTT-3.1.2-15]。

3.1.2.8 用户名标志

位置：连接标志字节的第 7 位

如果用户名标志（User Name Flag）被设置为 0，有效载荷中**不能**包含用户名字段 [MQTT-3.1.2-16]。如果用户名标志被设置为 0，有效载荷中**必须**包含用户名字段 [MQTT-3.1.2-17]。

3.1.2.9 密码标志

位置：连接标志字节的第 6 位

如果密码标志（Password Flag）被设置为 0，有效载荷中不能包含密码字段 [MQTT-3.1.2-18]。如果密码标志被设置为 1，有效载荷中**必须**包含密码字段 [MQTT-3.1.2-19]。

非规范评注

相比 MQTT v3.1.1，此版本协议允许在没有用户名的情况下发送密码。这表明密码除了作为口令之外还可以有其他用途。

3.1.2.10 保持连接

图 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）使用双字节整数来表示以秒为单位的时间间隔。它是指在客户端传输完成一个 MQTT 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 MQTT 控制报文可以发送，客户端**必须**发送一个 PINGREQ 报文 [MQTT-3.1.2-20]。

如果服务端返回的 CONNACK 报文中包含服务端保持连接（Server Keep Alive），客户端**必须**使用此值代替其发送的保持连接（Keep Alive） [MQTT-3.1.2-21]。

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 1.5 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，并判定网络连接已断开 [MQTT-3.1.2-22]。

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（Keep Alive）值为零的结果是关闭保持连接（Keep Alive）机制。如果保持连接（Keep Alive）值为零，客户端不必按照任何特定的时间发送 MQTT 控制报文。

非规范评注

服务端可能因为其他原因断开客户端连接，比如服务端将要关闭服务。设置保持连接（Keep Alive）不保证客户端将一直保持连接状态。

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 CONNECT 属性

3.1.2.11.1 属性长度

CONNECT 报文可变报头中的属性（Properties）长度被编码为变长字节整数。

3.1.2.11.2 会话过期间隔

17 (0x11)，会话过期间隔（Session Expiry Interval）标识符。

跟随其后的是用四字节整数表示的以秒为单位的会话过期间隔（Session Expiry Interval）。包含多个会话过期间隔（Session Expiry Interval）将造成协议错误（Protocol Error）。

如果会话过期间隔（Session Expiry Interval）值未指定，则使用 0。如果设置为 0 或者未指定，会话将在网络连接（Network Connection）关闭时结束。

如果会话过期间隔（Session Expiry Interval）为 0xFFFFFFFF (UINT_MAX)，则会话永不过期。

如果网络连接关闭时会话过期间隔（Session Expiry Interval）大于 0，则客户端与服务端**必须**存储会话状态 [MQTT-3.1.2-23]。

非规范评注

客户端或服务端可能会因为中断运行导致会话时钟某些时间未运行。这将导致会话的删除被延迟。

更多关于会话的信息参考 4.1 节。关于会话存储的状态的详细和限制参考 4.1.1 节。

当会话过期时，客户端和服务端无需以原子操作的方式删除会话状态。

非规范评注

把新开始（Clean Start）设置为 1 且会话过期间隔（Session Expiry Interval）设置为 0，等同于在 MQTT v3.1.1 中把清理会话（CleanSession）设置为 1。把新开始（Clean Start）设置为 0 且不设置会话过期间隔（Session Expiry Interval），等同于在 MQTT v3.1.1 中把清理会话标志设置为 0。

非规范评注

当希望只处理连接上服务端之后才发布的消息，客户端应该把新开始（Clean Start）设置为 1 且会话过期间隔（Session Expiry Interval）设置为 0，这样客户端就不会收到它连接之前被服务端所发布的消息，并且需要每次连接上服务端时重新订阅其感兴趣的主题。

非规范评注

某些客户端使用的网络可能只能提供断断续续的连接，这种客户端可以使用较短的会话过期间隔（Session Expiry Interval）以便在网络再次可用后重新连接到服务端时获得持续的消息交付。如果客户端不再重新连接，且允许会话过期，应用消息将会丢失。

非规范评注

某个客户端设置较长的会话过期间隔（Session Expiry Interval）或设置会话不过期，即要求服务端为其保持会话到其下一次连接上服务端之后。只有打算在一段时间之后将会重连服务端时，客户端才应该设置较长的会话过期间隔（Session Expiry Interval）。当客户端认定其将来不会使用本次会话时，应该在断开时把会话过期间隔（Session Expiry Interval）设置为 0。

非规范评注

客户端应当使用 CONNACK 报文中的会话存在（Session Present）来判定服务端是否存储了其会话。

非规范评注

客户端应当以服务端返回的会话存在（Session Present）标志来判定会话是否已过期，而不是客户端自己实现的会话过期状态。如果客户端自己实现会话过期状态，则需要将会话应当被删除的时间作为会话状态的一部分而存储。

3.1.2.11.3 接收最大值

33 (0x21)，接收最大值（Receive Maximum）标识符。

跟随其后的是由双字节整数表示的最大接收值。包含多个接收最大值或接收最大值为 0 将造成协议错误（Protocol Error）。

客户端使用此值限制客户端愿意同时处理的 QoS 等级 1 和 QoS 等级 2 的发布消息最大数量。没有机制可以限制服务端试图发送的 QoS 为 0 的发布消息。

接收最大值只将被应用在当前网络连接。如果没有设置最大接收值，将使用默认值 65535。

关于接收最大值的详细使用，参考 4.9 节 流控。

3.1.2.11.4 最大报文长度

39 (0x27)，最大报文长度（Maximum Packet Size）标识符。

跟随其后的是由四字节整数表示的客户端愿意接收的最大报文长度（Maximum Packet Size），如果没有设置最大报文长度（Maximum Packet Size），则按照协议由固定报头中的剩余长度可编码最大值和协议报头对数据包的大小做限制。

包含多个最大报文长度（Maximum Packet Size）或者最大报文长度（Maximum Packet Size）值为 0 将造成协议错误。

非规范评注

客户端如果选择了限制最大报文长度，应该为最大报文长度设置一个合理的值。

如 2.1.4 节 所述，最大报文长度是 MQTT 控制报文的总长度。客户端使用最大报文长度通知服务端其所能处理的单个报文长度限制。

服务端**不能**发送超过最大报文长度（Maximum Packet Size）的报文给客户端 [MQTT-3.1.2-24]。收到长度超过限制的报文将导致协议错误，客户端发送包含原因码 0x95（报文过大）的 DISCONNECT 报文给服务端，详见 4.13 节。

当报文过大而不能发送时，服务端**必须**丢弃这些报文，然后当做应用消息发送已完成处理 [MQTT-3.1.2-25]。

共享订阅的情况下，如果一条消息对于部分客户端来说太长而不能发送，服务端可以选择丢弃此消息或者把消息发送给剩余能够接收此消息的客户端。

非规范评注

服务端可以把那些没有发送就被丢弃的报文放在*死信队列*上，或者执行其他诊断操作。具体的操作超出了本规范的范围。

3.1.2.11.5 主题别名最大值

34 (0x22)，主题别名最大值（Topic Alias Maximum）标识符。

跟随其后的是用双字节整数表示的主题别名最大值（Topic Alias Maximum）。包含多个主题别名最大值（Topic Alias Maximum）将造成协议错误（Protocol Error）。没有设置主题别名最大值属性的情况下，主题别名最大值默认为零。

此值指示了客户端能够接收的来自服务端的主题别名（Topic Alias）最大数量。客户端使用此值来限制本次连接可以拥有的主题别名的数量。服务端在一个 PUBLISH 报文中发送的主题别名**不能**超过客户端设置的主题别名最大值（Topic Alias Maximum） [MQTT-3.1.2-26]。值为零表示本次连接客户端不接受任何主题别名（Topic Alias）。如果主题别名最大值（Topic Alias）没有设置，或者设置为零，则服务端**不能**向此客户端发送任何主题别名（Topic Alias） [MQTT-3.1.2-27]。

3.1.2.11.6 请求响应信息

25 (0x19)，请求响应信息（Request Response Information）标识符。

跟随其后的是用一个字节表示的 0 或 1。包含多个请求响应信息（Request Response Information），或者请求响应信息（Request Response Information）的值既不为 0 也不为 1 会造成协议错误（Protocol Error）。如果没有请求响应信息（Request Response Information），则请求响应默认值为 0。

客户端使用此值向服务端请求 CONNACK 报文中的响应信息（Response Information）。值为 0，表示服务端**不能**返回响应信息 [MQTT-3.1.2-28]。值为 1，表示服务端**可以**在 CONNACK 报文中返回响应信息。

非规范评注

即使客户端请求响应信息（Response Information），服务端也可以选择不发送响应信息（Response Information）。

更多关于请求/响应信息的内容，请参考 4.10 节。

3.1.2.11.7 请求问题信息

23 (0x17)，请求问题信息（Request Problem Information）标识符。

跟随其后的是用一个字节表示的 0 或 1。包含多个请求问题信息（Request Problem Information），或者请求问题信息（Request Problem Information）的值既不为 0 也不为 1 会造成协议错误（Protocol Error）。如果没有请求问题信息（Request Problem Information），则请求问题默认值为 1。

客户端使用此值指示遇到错误时是否发送原因字符串（Reason String）或用户属性（User Properties）。

如果请求问题信息的值为 0，服务端**可以**选择在 CONNACK 或 DISCONNECT 报文中返回原因字符串（Reason String）或用户属性（User Properties），但**不能**在除 PUBLISH，CONNACK 或 DISCONNECT 之外的报文中发送原因字符串（Reason String）或用户属性（User Properties） [MQTT-3.1.2-29]。如果此值为 0，并且在除 PUBLISH，CONNACK 或 DISCONNECT 之外的报文中收到了原因字符串（Reason String）或用户属性（User Properties），客户端将发送一个包含原因码 0x82（协议错误）的 DISCONNECT 报文给服务端，如 4.13 节 所述。

如果此值为 1，服务端**可以**在任何被允许的报文中返回原因字符串（Reason String）或用户属性（User Properties）。

3.1.2.11.8 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是 UTF-8 字符串对。

用户属性（User Property）可以出现多次，表示多个名字/值对。相同的名字可以出现多次。

> **非规范评注**
>
> CONNECT 报文中的用户属性可以被用来发送客户端到服务端的连接相关的属性。这些属性的意义本规范不做定义。

3.1.2.11.9 认证方法

21 (0x15)，认证方法（Authentication Method）标识符。

跟随其后的是一个 UTF-8 编码的字符串，包含了扩展认证的认证方法（Authentication Method）名称。包含多个认证方法将造成协议错误（协议错误）。

如果没有认证方法，则不进行扩展验证。参考 4.12 节。

如果客户端在 CONNECT 报文中设置了认证方法，则客户端在收到 CONNACK 报文之前**不能**发送除 AUTH 或 DISCONNECT 之外的报文 [MQTT-3.1.2-30]。

3.1.2.11.10 认证数据

22 (0x16)，认证数据（Authentication Data）标识符。

跟随其后的是二进制的认证数据。没有认证方法却包含了认证数据（Authentication Data），或者包含多个认证数据（Authentication Data）将造成协议错误（Protocol Error）。

认证数据的内容由认证方法定义，关于扩展认证的更多信息，请参考 4.12 节。

3.1.2.12 可变报头非规范示例

图 3-6 - 可变报头示例

	说明	7	6	5	4	3	2	1	0
协议名 Protocol Name									
byte 1	长度 Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	长度 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
协议版本 Protocol Version									
	说明	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 7	版本 Version (5)	0	0	0	0	0	1	0	1
连接标志 Connect Flags									
byte 8	用户名标志 User Name Flag (1) 密码标志 Password Flag (1) 遗嘱保留标志 Will Retain (0) 遗嘱服务质量 Will QoS (01) 遗嘱标志 Will Flag (1) 新开始 Clean Start(1) *保留 Reserved* (0)	1	1	0	0	1	1	1	0
保持连接 Keep Alive									
byte 9	保持连接 Keep Alive MSB (0)	0	0	0	0	0	0	0	0
byte 10	保持连接 Keep Alive LSB (10)	0	0	0	0	1	0	1	0
属性 Properties									
byte 11	长度 Length (5)	0	0	0	0	0	1	0	1
byte 12	会话过期间隔标识符 (17)	0	0	0	1	0	0	0	1
byte 13	会话过期间隔 Session Expiry Interval (10)	0	0	0	0	0	0	0	0
byte 14		0	0	0	0	0	0	0	0
byte 15		0	0	0	0	0	0	0	0

byte 16		0	0	0	0	1	0	1	0

3.1.3 CONNECT 载荷

CONNECT 报文的载荷中包含由可变报头（Variable Header）中的标志确定的一个或多个以长度为前缀的字段。这些字段若存在，**必须**按照客户标识符（Client Identifier）、遗嘱属性（Will Properties）、遗嘱主题（Will Topic）、遗嘱载荷（Will Payload）、用户名（User Name）、密码（Password）的顺序出现 [MQTT-3.1.3-1]。

3.1.3.1 客户标识符

服务端使用客户标识符（ClientID）识别客户端。连接服务端的每个客户端都有唯一的客户标识符（ClientID）。客户端和服务端都**必须**使用客户标识符（ClientID）识别两者之间的 MQTT 会话相关的状态 [MQTT-3.1.3-2]。更多关于会话状态的信息请参考 4.1 节。

客户标识符**必须**存在，且作为 CONNECT 报文载荷的第一个字段出现 [MQTT-3.1.3-3]。

客户标识符**必须**被编码为 1.5.4 节 中所定义的 UTF-8 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 编码的客户标识符，客户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 个字节的客户标识符 (ClientID)。服务端**可以**允许包含不是上面列表字符的客户标识符 (ClientID)。

服务端**可以**允许客户端提供一个零字节的客户标识符 (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标识符给那个客户端 [MQTT-3.1.3-6]。然后它**必须**假设客户端提供了那个唯一的客户标识符，正常处理这个 CONNECT 报文 [MQTT-3.1.3-7]。

如果服务端拒绝了某个客户标识符（ClientID），它**可以**发送包含原因码 0x85（客户标识符无效）的 CONNACK 报文作为对客户端的 CONNECT 报文的回应，如 4.13 节 所述。之后**必须**关闭网络连接 [MQTT-3.1.3-8]。

> **非规范评注**
>
> 客户端在实现时可以提供一个便于生成随机客户标识符的算法。使用此算法时，客户端需要注意避免创建长期孤儿会话。

3.1.3.2 遗嘱属性

如果遗嘱标志（Will Flag）被设置为 1，有效载荷的下一个字段是遗嘱属性（Will Properties）。遗嘱属性字段定义了遗嘱消息（Will Message）将何时被发布，以及被发布时的应用消息（Application Message）属性。遗嘱属性包括属性长度和属性。

3.1.3.2.1 属性长度

遗嘱属性（Will Properties）中的属性长度被编码为可变长字节整数。

3.1.3.2.2 遗嘱延时间隔

24 (0x18)，遗嘱延时间隔（Will Delay Interval）标识符。

跟随其后的是由四字节整数表示的以秒为单位的遗嘱延时间隔（Will Delay Interval）。包含多个遗嘱延时间隔将造成协议错误（Protocol Error）。如果没有设置遗嘱延时间隔，遗嘱延时间隔默认值将为 0，即不用延时发布遗嘱消息（Will Message）。

服务端将在遗嘱延时间隔（Will Delay Interval）到期或者会话（Session）结束时发布客户端的遗嘱消息（Will Message），取决于两者谁先发生。如果某个会话在遗嘱延时间隔到期之前创建了新的网络连接，则服务端**不能**发送遗嘱消息 [MQTT-3.1.3-9]。

非规范评注

遗嘱时间间隔的一个用途是避免在频繁的网络连接临时断开时发布遗嘱消息，因为客户端往往会很快重新连上网络并继续之前的会话。

非规范评注

如果某个连接到服务端的网络连接使用已存在的客户标识符，此已存在的网络连接的遗嘱消息将会被发布，除非新的网络连接设置了新开始（Clean Start）为 0 并且遗嘱延时大于 0。如果遗嘱延时为 0，遗嘱消息将在网络连接断开时发布。如果新开始为 1，遗嘱消息也将被发布，因为此会话已结束。

3.1.3.2.3 载荷格式指示

1 (0x01)，载荷格式指示（Payload Format Indicator）标识符。

跟随载荷格式指示（Payload Format Indicator ）之后的可能是：

- 0 (0x00)，表示遗嘱消息（Will Message）是未指定的字节，等同于不发送载荷格式指示。
- 1 (0x01)，表示遗嘱消息（Will Message）是 UTF-8 编码的字符数据。载荷中的 UTF-8 数据**必须**按照 Unicode 规范[Unicode] 和 RFC 3629 [RFC3629]中的申明进行编码。

包含多个载荷格式指示（Payload Format Indicator）将造成协议错误（Protocol Error）。服务端**可以**按照格式指示对遗嘱消息（Will Message）进行验证，如果验证失败发送一条包含原因码 0x99（载荷格式无效）的 CONNACK 报文。如 4.13 节 所述。

3.1.3.2.4 消息过期间隔

2 (0x02)，消息过期间隔（Message Expiry Interval）标识符。

跟随其后的是表示消息过期间隔（Message Expiry Interval）的四字节整数。包含多个消息过期间隔将导致协议错误（Protocol Error）。

如果设定了消息过期间隔（Message Expiry Interval），四字节整数描述了遗嘱消息的生命周期（秒），并在服务端发布遗嘱消息时被当做发布过期间隔（Publication Expiry Interval）。

如果没有设定消息过期间隔，服务端发布遗嘱消息时将不发送消息过期间隔（Message Expiry Interval）。

3.1.3.2.5 内容类型

3 (0x03)，内容类型（Content Type）标识符。

跟随其后的是一个以 UTF-8 格式编码的字符串，用来描述遗嘱消息（Will Message）的内容。包含多个内容类型（Content Type）将造成协议错误（Protocol Error）。内容类型的值由发送应用程序和接收应用程序确定。

3.1.3.2.6 响应主题

8 (0x08)，响应主题（Response Topic）标识符。

跟随其后的是一个以 UTF-8 格式编码的字符串，用来表示响应消息的主题名（Topic Name）。包含多个响应主题（Response Topic）将造成协议错误。响应主题的存在将遗嘱消息（Will Message）标识为一个请求报文。

更多关于请求/响应的内容，参考 4.10 节。

3.1.3.2.7 对比数据

9 (0x09)，对比数据（Correlation Data）标识符。

跟随其后的是二进制数据。对比数据被请求消息发送端在收到响应消息时用来标识相应的请求。包含多个对比数据将造成协议错误（Protocol Error）。如果没有设置对比数据，则请求方（Requester）不需要任何对比数据。

对比数据只对请求消息（Request Message）的发送端和响应消息（Response Message）的接收端有意义。

更多关于请求/响应的内容，参考 4.10 节。

3.1.3.2.8 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是一个 UTF-8 字符串对。用户属性（User Property）可以出现多次，表示多个名字/值对。相同的名字可以出现多次。

服务端在发布遗嘱消息（Will Message）时**必须**维护用户属性（User Properties）的顺序 [MQTT-3.1.3-10]。

非规范评注

此属性旨在提供一种传递应用层名称-值标签的方法，其含义和解释仅由负责发送和接收它们的应用程序所有。

3.1.3.3 遗嘱主题

如果遗嘱标志（Will Flag）被设置为 1，遗嘱主题（Will Topic）为载荷中下一个字段。遗嘱主题（Will Topic）**必须**为 UTF-8 编码的字符串，如 1.5.4 节 所定义 [MQTT-3.1.3-11]。

3.1.3.4 遗嘱载荷

如果遗嘱标志（Will Flag）被设置为 1，遗嘱载荷（Will Payload）为载荷中下一个字段。遗嘱载荷定义了将要发布到遗嘱主题（Will Topic）的应用消息载荷，如 3.1.2.5 节 所定义。此字段为二进制数据。

3.1.3.5 用户名

如果用户名标志（User Name Flag）被设置为 1，用户名（User Name）为载荷中下一个字段。用户名**必须是** 1.5.4 节 定义的 UTF-8 编码字符串 [MQTT-3.1.3-12]。服务端可以将它用于身份验证和授权。

3.1.3.6 密码

如果密码标志（Password Flag）被设置为 1，密码（Password）为载荷中下一个字段。密码字段是二进制数据，尽管被称为密码，但可以被用来承载任何认证信息。

3.1.4 CONNECT 行为

注意：服务端**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 MQTT 协议的早期版本）。如果服务端确定协议是 MQTT v5.0，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，服务端关闭网络连接 [MQTT-3.1.4-1]。服务端**可以**在关闭网络连接之前发送包含 3.1.2.4 节所述的 0x80 及以上原因码的 CONNACK 报文。
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，**应该**执行身份验证和授权检查。如果任何一项检查没通过，服务端**必须**关闭网络连接 [MQTT-3.1.4-2]。在关闭网络连接之前，服务端**可以**发送一个合适的包含如 3.2 节 和 4.13 节 所述的 0x80 及以上原因码的 CONNACK 报文。

如果验证成功，服务端会执行下列步骤。

1. 如果客户标识符（ClientID）所代表的客户端已经连接到此服务端，那么向原有的客户端发送一个包含原因码为 0x8E（会话被接管）的 DISCONNECT 报文，并且**必须**关闭原有的网络连接 [MQTT-3.1.4-3]。如果原有客户端存在遗嘱消息（Will Message），遗嘱消息按照 3.1.2.5 节 所描述的方式发布。

非规范评注

如果原有网络连接包含遗嘱消息，且遗嘱延时间隔为 0，则遗嘱消息会在此网络连接被关闭时发送。如果原有网络连接会话过期间隔为 0，或者新网络连接新开始标志设置为 1 且原有网络连接包含遗嘱消息，则遗嘱消息会被发送，因为原有会话已结束。

2. 服务端**必须**按照 3.1.2.4 节 所描述的方式对新开始标志进行处理[MQTT-3.1.4-4]。

3. 服务端**必须**使用包含原因码为 0x00（成功）的 CONNACK 报文对客户端的 CONNECT 报文进行确认 [MQTT-3.1.4-5]。

非规范评注

如果服务端被用来处理商业关键数据，推荐对网络连接进行认证和授权。如果认证和授权成功，服务端可通过发送包含原因码为 0x00（成功）的 CONNACK 报文进行响应，否则建议服务端根本不要发送 CONNACK 报文，因为这是一种潜在的对 MQTT 服务端的攻击，可以被用来进行拒绝服务攻击或密码猜测攻击。

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 MQTT 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 报文，它**不能**处理客户端在 CONNECT 报文之后发送的任何除 AUTH 以外的报文 [MQTT-3.1.4-6]。

非规范评注

客户端通常会等待 CONNACK 报文。然而，如果在收到 CONNACK 报文之前就自由的发送其它 MQTT 控制报文将会简化客户端的实现，因为它不必监督连接的状态。如果连接被拒绝了，客户端在接收 CONNACK 报文之前发送的任何数据将不会被服务端所处理。

非规范评注

选择在收到 CONNACK 报文之前就发送 MQTT 控制报文的客户端将不知道服务端所存在的约束以及会话是否被使用。

非规范评注

服务端在对某个客户端完成认证之前，可以选择限制读取该客户端的网络数据或者关闭该客户端的网络连接。这是一种避免拒绝服务攻击的方法。

3.2 CONNACK – 确认连接请求

CONNACK 报文由服务端所发送，作为对来自客户端的 CONNECT 报文的响应。服务端在发送任何除 AUTH 以外的报文之前**必须**先发送包含原因码为 0x00（成功）的 CONNACK 报文 [MQTT-3.2.0-1]。服务端在一次网络连接中**不能**发送多个 CONNACK 报文 [MQTT-3.2.0-2]。

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 CONNACK 固定报头

固定报头的格式见图 3-7 的描述。

图 3-7 – 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							

剩余长度字段

用变长字节整数来编码，表示可变报头的长度。

3.2.2 CONNACK 可变报头

CONNACK 报文的可变报头按顺序包含以下字段：连接确认标志（Connect Acknowledge Flags），连接原因码（Reason Code），属性（Properties）。属性的编码规则如 2.2.2 节 所描述。

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 [MQTT-3.2.2-1] 。

第 0（SP）位是会话存在标志（Session Present Flag）。

3.2.2.1.1 会话存在

位置：连接确认标志（Connect Acknowledge Flags）的第 0 位。

会话存在（Session Present）标志通知客户端，服务端是否正在使用此客户标识符之前连接的会话状态（Session State）。会话存在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。

如果服务端接受一个新开始（Clean Start）为 1 的连接，服务端在 CONNACK 报文中除了把原因码设置为 0x00（成功）之外，还**必须**把会话存在标志设置为 0 [MQTT-3.2.2-2]。

如果服务端接受一个新开始（Clean Start）为 0 的连接，并且服务端已经保存了此客户标识符（ClientID）的会话状态（Session State），服务端在 CONNACK 报文中**必须**把会话存在标志设置为 1。否则，服务端**必须**把会话存在标志设置为 0。无论如何，服务端在 CONNACK 报文中**必须**把原因码设置为 0x00（成功） [MQTT-3.2.2-3]。

如果客户端从服务端接收到的会话存在标志值与预期的不同，客户端做如下处理：

- 如果客户端没有保存的会话状态，但收到会话存在标志为 1，客户端**必须**关闭网络连接 [MQTT-3.2.2-4]。 如果希望重新开始一个新的会话，客户端可以使用新开始（Clean Start）为 1 并重新连接服务端。
- 如果客户端保存了会话状态，但收到的会话存在标志为 0，客户端若要继续此网络连接，它**必须**丢弃其保存的会话状态 [MQTT-3.2.2-5]。

如果服务端发送的 CONNACK 报文中原因码非 0，它**必须**把会话存在标志设置为 0 [MQTT-3.2.2-6]。

3.2.2.2 连接原因码

可变报头中第 2 个字节是连接原因码（Reason Code）。

连接原因码（Reason Code）的值如下所示。如果服务端收到一个格式正确的 CONNECT 报文，但服务端无法完成连接的创建，服务端**可以**发送一个包含适当的连接原因码的 CONNACK 报文。如果服务端发送了一个包含原因码大于等于 128 的 CONNACK 报文，它随后**必须**关闭网络连接 [MQTT-3.2.2-7]。

表 3-1 - 连接原因码

值	16 进制	原因码名称	说明
0	0x00	成功	连接被接受。
128	0x80	未指明的错误	服务端不愿透露的错误，或者没有适用的原因码。
129	0x81	无效报文	CONNECT 报文内容不能被正确的解析。
130	0x82	协议错误	CONNECT 报文内容不符合本规范。
131	0x83	实现特定错误	CONNECT 有效，但不被服务端所接受。
132	0x84	协议版本不支持	服务端不支持客户端所请求的 MQTT 协议版本。
133	0x85	客户标识符无效	客户标识符有效，但未被服务端所接受。
134	0x86	用户名密码错误	客户端指定的用户名密码未被服务端所接受。
135	0x87	未授权	客户端未被授权连接。
136	0x88	服务端不可用	MQTT 服务端不可用。
137	0x89	服务端正忙	服务端正忙，请重试。
138	0x8A	禁止	客户端被禁止，请联系服务端管理员。
140	0x8C	无效的认证方法	认证方法未被支持，或者不匹配当前使用的认证方法。
144	0x90	主题名无效	遗嘱主题格式正确，但未被服务端所接受。
149	0x95	报文过长	CONNECT 报文超过最大允许长度。
151	0x97	超出配额	已超出实现限制或管理限制。

153	0x99	载荷格式无效	遗嘱载荷数据与载荷格式指示符不匹配。
154	0x9A	不支持保留	遗嘱保留标志被设置为 1，但服务端不支持保留消息。
155	0x9B	不支持的 QoS 等级	服务端不支持遗嘱中设置的 QoS 等级。
156	0x9C	（临时）使用其他服务端	客户端应该临时使用其他服务端。
157	0x9D	服务端已（永久）移动	客户端应该永久使用其他服务端
159	0x9F	超出连接速率限制	超出了所能接受的连接速率限制。

服务端发送的 CONNACK 报文**必须**设置一种原因码 [MQTT-3.2.2-8]。

非规范评注

原因码 0x80（未指明的错误）可以被用作：服务器知道失败的原因但是并不希望透露给客户端，或者没有其他适用的原因码。

出于安全考虑，发现 CONNECT 出错时服务端可以选择不发送 CONNACK 报文而关闭网络连接。例如，在公网中向未被授权的网络连接告知自身 MQTT 服务端身份并不明智。

3.2.2.3 CONNACK 属性

3.2.2.3.1 属性长度

CONNACK 报文可变报头中的属性长度，编码为变长字节整数。

3.2.2.3.2 会话过期间隔

17 (0x11)，会话过期间隔（Session Expiry Interval）标识符。

跟随其后的是用四字节整数表示的以秒为单位的会话过期间隔（Session Expiry Interval）。包含多个会话过期间隔（Session Expiry Interval）将造成协议错误（Protocol Error）。

如果会话过期间隔（Session Expiry Interval）值未指定，则使用 CONNECT 报文中指定的会话过期时间间隔。服务端使用此属性通知客户端它使用的会话过期时间间隔与客户端在 CONNECT 中发送的值不同。更详细的关于会话过期时间的描述，请参考 3.1.2.11.2 节 。

3.2.2.3.3 接收最大值

33 (0x21)，接收最大值（Receive Maximum）描述符。

跟随其后的是由双字节整数表示的最大接收值。包含多个接收最大值或接收最大值为 0 将造成协议错误（Protocol Error）。

服务端使用此值限制服务端愿意为该客户端同时处理的 QoS 为 1 和 QoS 为 2 的发布消息最大数量。没有机制可以限制客户端试图发送的 QoS 为 0 的发布消息。

如果没有设置最大接收值，将使用默认值 65535。

关于接收最大值的详细使用，参考 4.9 节 流控部分。

3.2.2.3.4 最大服务质量

36 (0x24)，最大服务质量（Maximum QoS）标识符。

跟随其后的是用一个字节表示的 0 或 1。包含多个最大服务质量（Maximum QoS）或最大服务质量既不为 0 也不为 1 将造成协议错误。如果没有设置最大服务质量，客户端可使用最大 QoS 为 2。

如果服务端不支持 Qos 为 1 或 2 的 PUBLISH 报文，服务端**必须**在 CONNACK 报文中发送最大服务质量以指定其支持的最大 QoS 值 [MQTT-3.2.2-9]。即使不支持 QoS 为 1 或 2 的 PUBLISH 报文，服务端也**必须**接受请求 QoS 为 0、1 或 2 的 SUBSCRIBE 报文 [MQTT-3.2.2-10]。

如果从服务端接收到了最大 QoS 等级，则客户端**不能**发送超过最大 QoS 等级所指定的 QoS 等级的 PUBLISH 报文 [MQTT-3.2.2-11]。服务端接收到超过其指定的最大服务质量的 PUBLISH 报文将造成协议错误（Protocol Error）。这种情况下应使用包含原因码为 0x9B（不支持的 QoS 等级）的 DISCONNECT 报文进行处理，如 4.13 节 所述。

如果服务端收到包含遗嘱的 QoS 超过服务端处理能力的 CONNECT 报文，服务端**必须**拒绝此连接。服务端**应该**使用包含原因码为 0x9B（不支持的 QoS 等级）的 CONNACK 报文进行错误处理，随后**必须**关闭网络连接。4.13 节 所述 [MQTT-3.2.2-12]。

> **非规范评注**
>
> 客户端不必支持 QoS 为 1 和 2 的 PUBLISH 报文。客户端只需将其发送的任何 SUBSCRIBE 报文中的 QoS 字段限制在其支持的最大服务质量以内即可。

3.2.2.3.5 保留可用

37 (0x25)，保留可用（Retain Available）标识符。

跟随其后的是一个单字节字段，用来声明服务端是否支持保留消息。值为 0 表示不支持保留消息，为 1 表示支持保留消息。如果没有设置保留可用字段，表示支持保留消息。包含多个保留可用字段或保留可用字段值不为 0 也不为 1 将造成协议错误（Protocol Error）。

如果服务端收到一个包含保留标志位 1 的遗嘱消息的 CONNECT 报文且服务端不支持保留消息，服务端**必须**拒绝此连接请求，且**应该**发送包含原因码为 0x9A（不支持保留）的 CONNACK 报文，随后**必须**关闭网络连接 [MQTT-3.2.2-13]。

从服务端接收到的保留可用标志为 0 时，客户端**不能**发送保留标志设置为 1 的 PUBLISH 报文 [MQTT-3.2.2-14]。如果服务端收到这种 PUBLISH 报文，将造成协议错误（Protocol Error），此时服务端**应该**发送包含原因码为 0x9A（不支持保留）的 DISCONNECT 报文，如 4.13 节 所述。

3.2.2.3.6 最大报文长度

39 (0x27)，最大报文长度（Maximum Packet Size）标识符。

跟随其后的是由四字节整数表示的服务端愿意接收的最大报文长度（Maximum Packet Size）。如果没有设置最大报文长度，则按照协议由固定报头中的剩余长度可编码最大值和协议报头对数据包的大小做限制。

包含多个最大报文长度（Maximum Packet Size），或最大报文长度为 0 将造成协议错误（Protocol Error）。

如 2.1.4 节 所述，最大报文长度是 MQTT 控制报文的总长度。服务端使用最大报文长度通知客户端其所能处理的单个报文长度限制。

客户端不能发送超过最大报文长度（Maximum Packet Size）的报文给服务端 [MQTT-3.2.2-15]。收到长度超过限制的报文将导致协议错误，此时服务端应该发送包含原因码 0x95（报文过长）的 DISCONNECT 报文给客户端，详见 4.13 节。

3.2.2.3.7 分配客户标识符

18 (0x12)，分配客户标识符（Assigned Client Identifier）标识符。

跟随其后的是 UTF-8 编码的分配客户标识符（Assigned Client Identifier）字符串。包含多个分配客户标识符将造成协议错误（Protocol Error）。

服务端分配客户标识符的原因是 CONNECT 报文中的客户标识符长度为 0。

如果客户端使用长度为 0 的客户标识符（ClientID），服务端**必须**回复包含分配客户标识符（Assigned Client Identifier）的 CONNACK 报文。分配客户标识符**必须**是没有被服务端的其他会话所使用的新客户标识符 [MQTT-3.2.2-16]。

3.2.2.3.8 主题别名最大值

34 (0x22)，主题别名最大值（Topic Alias Maximum）标识符。

跟随其后的是用双字节整数表示的主题别名最大值（Topic Alias Maximum）。包含多个主题别名最大值（Topic Alias Maximum）将造成协议错误（Protocol Error）。没有设置主题别名最大值属性的情况下，主题别名最大值默认为零。

此值指示了服务端能够接收的来自客户端的主题别名（Topic Alias）最大值。服务端使用此值来限制本次连接可以拥有的主题别名的值。客户端在一个 PUBLISH 报文中发送的主题别名值**不能**超过服务端设置的主题别名最大值（Topic Alias Maximum） [MQTT-3.2.2-17]。值为 0 表示本次连接服务端不接受任何主题别名（Topic Alias）。如果主题别名最大值（Topic Alias）没有设置，或者设置为 0，则客户端**不能**向此服务端发送任何主题别名（Topic Alias） [MQTT-3.2.2-18]。

3.2.2.3.9 原因字符串

31 (0x1F)，原因字符串（Reason String）标识符。

跟随其后的是 UTF-8 编码的字符串，表示此次响应相关的原因。此原因字符串（Reason String）是为诊断而设计的可读字符串，**不应该**被客户端所解析。

服务端使用此值向客户端提供附加信息。如果加上原因字符串之后的 CONNACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此原因字符串 [MQTT-3.2.2-19]。包含多个原因字符串将造成协议错误（Protocol Error）。

非规范评注

客户端对原因字符串的恰当使用包括：抛出异常时使用此字符串，或者将此字符串写入日志。

3.2.2.3.10 用户属性

38 (0x26)，用户属性（User Property）标识符。
跟随其后的是 UTF-8 字符串对。此属性可用于向客户端提供包括诊断信息在内的附加信息。如果加上用户属性之后的 CONNACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此属性 [MQTT-3.2.2-20]。用户属性（User Property）允许出现多次，以表示多个名字/值对，且相同的名字可以多次出现。

用户属性的内容和意义本规范不做定义。CONNACK 报文的接收端**可以**选择忽略此属性。

3.2.2.3.11 通配符订阅可用

40 (0x28)，通配符订阅可用（Wildcard Subscription Available）标识符。

跟随其后的是一个单字节字段，用来声明服务器是否支持通配符订阅（Wildcard Subscriptions）。值为 0 表示不支持通配符订阅，值为 1 表示支持通配符订阅。如果没有设置此值，则表示支持通配符订阅。包含多个通配符订阅可用属性，或通配符订阅可用属性值不为 0 也不为 1 将造成协议错误（Protocol Error）。

如果服务端在不支持通配符订阅（Wildcard Subscription）的情况下收到了包含通配符订阅的 SUBSCRIBE 报文，将造成协议错误（Protocol Error）。此时服务端将发送包含原因码为 0xA2（通配符订阅不支持）的 DISCONNECT 报文，如 4.13 节 所述。

服务端在支持通配符订阅的情况下仍然可以拒绝特定的包含通配符订阅的订阅请求。这种情况下，服务端**可以**发送一个包含原因码为 0xA2（通配符订阅不支持）的 SUBACK 报文。

3.2.2.3.12 订阅标识符可用

41 (0x29)，订阅标识符可用（Subscription Identifier Available）标识符。

跟随其后的是一个单字节字段，用来声明服务端是否支持订阅标识符（Subscription Identifiers）。值为 0 表示不支持订阅标识符，值为 1 表示支持订阅标识符。如果没有设置此值，则表示支持订阅标识符。包含多个订阅标识符可用属性，或订阅标识符可用属性值不为 0 也不为 1 将造成协议错误（Protocol Error）。

如果服务端在不支持订阅标识符（Subscription Identifier）的情况下收到了包含订阅标识符的 SUBSCRIBE 报文，将造成协议错误（Protocol Error）。此时服务端将发送包含原因码为 0xA1（订阅标识符不支持）的 DISCONNECT 报文，如 4.13 节 所述。

3.2.2.3.13 共享订阅可用

42 (0x2A)，共享订阅可用（Shared Subscription Available）标识符。

跟随其后的是一个单字节字段，用来声明服务端是否支持共享订阅（Shared Subscription）。值为 0 表示不支持共享订阅，值为 1 表示支持共享订阅。如果没有设置此值，则表示支持共享订阅。包含多个共享订阅可用（Shared Subscription Available），或共享订阅可用属性值不为 0 也不为 1 将造成协议错误（Protocol Error）。

如果服务端在不支持共享订阅（Shared Subscription）的情况下收到了包含共享订阅的 SUBSCRIBE 报文，将造成协议错误（Protocol Error）。此时服务端将发送包含原因码为 0x9E（共享订阅不支持）的 DISCONNECT 报文，如 4.13 节 所述。

3.2.2.3.14 服务端保持连接

19 (0x13)，服务端保持连接（Server Keep Alive）标识符。

跟随其后的是由服务端分配的双字节整数表示的保持连接（Keep Alive）时间。如果服务端发送了服务端保持连接（Server Keep Alive）属性，客户端**必须**使用此值代替其在 CONNECT 报文中发送的保持连接时间值 [MQTT-3.2.2-21]。如果服务端没有发送服务端保持连接属性，服务端**必须**使用客户端在 CONNECT 报文中设置的保持连接时间值 [MQTT-3.2.2-22]。包含多个服务端保持连接属性将造成协议错误（Protocol Error）。

> **非规范评注**
>
> 服务端保持连接属性的主要作用是通知客户端它将会比客户端指定的保持连接更快的断开非活动的客户端。

3.2.2.3.15 响应信息

26 (0x1A)，响应信息（Response Information）标识符。

跟随其后的是一个以 UTF-8 编码的字符串，作为创建响应主题（Response Topic）的基本信息。关于客户端如何根据响应信息（Response Information）创建响应主题不在本规范的定义范围内。包含多个响应信息将造成协议错误（Protocol Error）。

如果客户端发送的请求响应信息（Request Response Information）值为 1，则服务端在 CONNACK 报文中发送响应信息（Response Information）为**可选**项。

> **非规范评注**
>
> 响应信息通常被用来传递主题订阅树的一个全局唯一分支，此分支至少在该客户端的会话生命周期内为该客户端所保留。请求客户端和响应客户端的授权需要使用它，所以它通常不能仅仅是一个随机字符串。一般把此分支作为特定客户端的订阅树根节点。通常此信息需要正确配置，以使得服务器能返回信息。使用此机制时，具体的信息一般由服务端来进行统一配置，而非由各个客户端自己配置。

更多关于请求/响应的信息，请参考 4.10 节 。

3.2.2.3.16 服务端参考

28 (0x1C)，服务端参考（Server Reference）标识符。

跟随其后的是一个以 UTF-8 编码的字符串，可以被客户端用来标识其他可用的服务端。包含多个服务端参考（Server Reference）将造成协议错误（Protocol Error）。

服务端在包含了原因码为 0x9C（（临时）使用其他服务端）或 0x9D（服务端已（永久）移动）的 CONNACK 报文或 DISCONNECT 报文中设置服务端参考，如 4.13 节 所述。

关于如何使用服务端参考，请参考 4.11 节 服务端重定向信息。

3.2.2.3.17 认证方法

21 (0x15)，认证方法（Authentication Method）标识符。

跟随其后的是一个以 UTF-8 编码的字符串，包含了认证方法（Authentication Method）名。包含多个认证方法将造成协议错误（Protocol Error）。更多关于扩展认证的信息，请参考 4.12 节 。

3.2.2.3.18 认证数据

22 (0x16)，认证数据（Authentication Data）标识符。

跟随其后的是包含认证数据（Authentication Data）的二进制数据。此数据的内容由认证方法和已交换的认证数据状态定义。包含多个认证数据将造成协议错误（Protocol Error）。更多关于扩展认证的信息，请参考 4.12 节 。

3.2.3 CONNACK 载荷

CONNACK 报文没有有效载荷。

3.3 PUBLISH – 发布消息

PUBLISH 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 PUBLISH 固定报头

图 3-8 – 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 为 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 标志位 1 的 MQTT 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 MQTT 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。 2.2.1 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（QoS）等级保证。服务质量等级在下表中列出。

表 3-2 - QoS 定义

QoS 值	Bit 2	Bit 1	说明
0	0	0	最多分发一次
1	0	1	至少分发一次
2	1	0	仅分发一次
-	1	1	保留 – 不能使用

如果服务端在对客户端响应的 CONNACK 报文中包含了最大服务质量（Maximum QoS）且服务端收到的 PUBLISH 报文的 QoS 大于此最大服务质量，服务端发送包含原因码为 0x9B（不支持的 QoS 等级）的 DISCONNECT 报文，如 4.13 节 所述。

PUBLISH 报文的 2 个 QoS 比特位不能同时设置为 1 [MQTT-3.3.1-4]。如果服务端或客户端收到 QoS 2 个比特位都为 1 的无效 PUBLISH 报文，使用包含原因码为 0x81（无效报文）的 DISCONNECT 报文关闭网络连接，如 4.13 节 所述。

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**存储此应用消息，并用其替换此话题下任何已存在的消息 [MQTT-3.3.1-5]，以便它可以被分发给未来的匹配此主题名（Topic Name）的订阅者。如果载荷为空，消息可以正常被服务端所处理，但是此话题下的任何保留消息**必须**被丢弃，并且此话题未来的订阅者将不会收到保留消息 [MQTT-3.3.1-6]。载荷为空的保留消息将**不能**被存储在服务端 [MQTT-3.3.1-7]。

如果客户端发给服务端的 PUBLISH 报文的保留标志位为 0，服务器**不能**把此消息存储为保留消息，也不能丢弃或替换任何已存在的保留消息 [MQTT-3.3.1-8]。

如果服务端发送给客户端的 CONNACK 报文中包含保留可用属性，且属性值为 0，但收到的 PUBLISH 报文中保留标志位为 1，服务端使用包含原因码为 0x9A（保留不支持）的 DISCONNECT 报文断开网络连接，如 4.13 节 所述。

当一个新的非共享订阅（Non-shared Subscription）被创建时，每个匹配的话题下的最新保留消息如果存在，将根据保留消息订阅选项（Retain Handling Subscription Option）发送给客户端。这些消息在发送时保留标志被设置为 1。保留消息的发送由保留消息处理订阅选项控制，收到订阅时：

- 如果保留消息处理属性被设置为 0，服务端**必须**发送主题与客户端订阅的主题过滤器（Topic Filter）相匹配的所有保留消息 [MQTT-3.3.1-9]。
- 如果保留消息处理属性被设置为 1，如果尚不存在匹配的订阅，服务端**必须**发送主题与客户端订阅的主题过滤器相匹配的所有保留消息。如果已存在相匹配的订阅，服务器**不能**发送这些保留消息 [MQTT-3.3.1-10]。
- 如果保留消息处理属性被设置为 2，服务器**不能**发送这些保留消息 [MQTT-3.3.1-11]。

订阅选项（Subscription Options）的定义，参考 3.8.3.1 节 。

如果服务端收到保留标志设置为 1 且 QoS 设置为 0 的 PUBLISH 报文，服务端**应该**把此 QoS 为 0 的消息存储为其主题下最新的保留消息，但服务端**可以**选择在任何时间丢弃此消息。如果发生丢弃，该主题下将不存在任何保留消息。

如果某个主题当前的保留消息过期，该主题下将不存在任何保留消息。

服务端转发应用消息时，保留标志位的设置由发布保留（Retain As Published）订阅选项决定。订阅选项的定义，请参考 3.8.3.1 节 。

- 如果发布保留（Retain As Published）订阅选项被设置为 0，服务端在转发应用消息时**必须**将保留标志设置为 0，而不管收到的 PUBLISH 报文中保留标志位如何设置的 [MQTT-3.3.1-12]。
- 如果发布保留（Retain As Published）订阅选项被设置为 1，服务端在转发应用消息时**必须**将保留标志设置为与收到的 PUBLISH 消息中的保留标志位相同 [MQTT-3.3.1-13]。

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。新的非共享订阅者将会收到最近的状态。

3.3.1.4 剩余长度

等于可变报头的长度加上有效载荷的长度，被编码为变长字节整数。

3.3.2 PUBLISH 可变报头

PUBLISH 报文可变报头按顺序包含：主题名（Topic Name），报文标识符（Packet Identifier），属性（Properties）。属性的编码规则如 2.2.2 节 所述。

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 1.5.4 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 报文中的主题名**必须**匹配该订阅的主题过滤器（Topic Filter），如 4.7 节 所定义的匹配过程 [MQTT-3.3.2-3]。然而，由于服务端允许将主题名映射为其他名字，主题名可能与原始 PUBLISH 报文中的主题名不同。

发送端可以使用主题别名（Topic Alias）以便减少 PUBLISH 报文的长度。主题别名如 3.3.2.3.4 节 所述。主题名长度为 0 且没有主题别名，将造成协议错误（Protocol Error）。

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。2.2.1 节 提供了有关报文标识符的更多信息。

3.3.2.3 PUBLISH 属性

3.3.2.3.1 属性长度

PUBLISH 报文可变报头中的属性长度被编码为变长字节整数。

3.3.2.3.2 载荷格式指示

1 (0x01)，载荷格式指示（Payload Format Indicator）标识符。

跟随其后的是单字节的载荷格式指示值，可以是：

- 0 (0x00)，说明载荷是未指定格式的字节，相当于没有发送载荷格式指示。

- 1 (0x01)，说明载荷是 UTF-8 编码的字符数据。载荷中的 UTF-8 数据**必须**是按照 Unicode [Unicode] 的规范和 RFC 3629 [RFC3629] 的重申进行编码。

服务端**必须**把接收到的应用消息中的载荷格式指示原封不动的发给所有的订阅者 [MQTT-3.3.2-4]。接收者**可以**验证载荷数据与所指示的格式一致，如果不一致，发送包含原因码为 0x99（载荷格式无效）的 PUBACK，PUBREC 或 DISCONNECT 报文，如 4.13 节 所述。

3.3.2.3.3 消息过期间隔

2 (0x02)，消息过期间隔（Message Expiry Interval）标识符。

跟随其后的是四字节整数表示的消息过期间隔（Message Expiry Interval）。

如果消息过期间隔存在，四字节整数表示以秒为单位的应用消息（Application Message）生命周期。如果消息过期间隔（Message Expiry Interval）已过期，服务端还没开始向匹配的订阅者交付该消息，则服务端**必须**删除该订阅者的消息副本 [MQTT-3.3.2-5]。

如果消息过期间隔不存在，应用消息不会过期。

服务端发送给客户端的 PUBLISH 报文中**必须**包含消息过期间隔，值为接收时间减去消息在服务端的等待时间 [MQTT-3.3.2-6]。关于状态存储的细节和限制，参考 4.1 节。

3.3.2.3.4 主题别名

35 (0x23)，主题别名（Topic Alias）标识符。

跟随其后的是表示主题别名（Topic Alias）值的双字节整数。包含多个主题别名值将造成协议错误（Protocol Error）。

主题别名是一个整数，用来代替主题名对主题进行识别。主题别名可以减小 PUBLISH 报文的长度，这对某个网络连接中发送的很长且反复使用的主题名来说很有用。

发送端决定是否使用主题别名及别名值如何选取。发送端通过在 PUBLISH 报文中包含的非 0 长度主题名和主题别名来设置主题别名映射。接收端正常处理该 PUBLISH 报文，但同样将指定的主题别名映射到主题名。

如果接收端已经设置了某个主题别名映射，发送端可以发送包含主题别名和长度为 0 的主题名的 PUBLISH 报文。接收端把此 PUBLISH 报文的主题名当做其包含的主题别名所映射的主题名。

发送端可以通过在同一个网络连接中发送另一个包含同样主题别名和不同非 0 长度主题名的 PUBLISH 报文来修改主题别名映射关系。

主题别名映射仅作用于某个网络连接及其生命周期内。接收端**不能**将任何主题别名映射从一个网络连接转发到另一个网络连接 [MQTT-3.3.2-7]。

主题别名不允许为 0。发送端**不能**发送包含主题别名值为 0 的 PUBLISH 报文 [MQTT-3.3.2-8]。

客户端**不能**发送主题别名值大于服务端的 CONNACK 报文中指定的主题别名最大值（Topic Alias Maximum）的 PUBLISH 报文 [MQTT-3.3.2-9]。客户端**必须**接受所有值大于 0 且小于等于其发送的 CONNECT 报文中的主题别名最大值的主题别名 [MQTT-3.3.2-10]。

服务端**不能**发送包含主题别名值大于客户端在 CONNECT 报文中指定的主题别名最大值（Topic Alias Maximum）的 PUBLISH 报文 [MQTT-3.3.2-11]。服务端**必须**接受所有值大于 0 且小于等于其发送的 CONNACK 报文中的主题别名最大值的主题别名 [MQTT-3.3.2-12]。

客户端和服务端使用的主题别名映射相互独立。因此一般来说，客户端发送给服务端的主题别名值为 1 的 PUBLISH 报文和服务端发送给客户端的主题别名值为 1 的 PUBLISH 报文，将被映射到不同的主题。

3.3.2.3.5 响应主题

8 (0x08)，响应主题（Response Topic）标识符。

跟随其后的是一个 UTF-8 编码的字符串，用作响应消息的主题名。响应主题**必须**是按照 1.5.4 节 所定义的 UTF-8 编码的字符串 [MQTT-3.3.2-13]。响应主题不能包含通配符 [MQTT-3.3.2-14]。包含多个响应主题将造成协议错误（Protocol Error）。响应主题的存在将消息标识为请求报文。

更多关于请求/响应的信息，参考 4.10 节 。

服务端在收到应用消息时**必须**将响应主题原封不动的发送给所有的订阅者 [MQTT-3.3.2-15]。

非规范评注

包含响应主题的应用消息接收端使用响应主题作为主题名，发送作为响应消息的 PUBLISH 报文。如果请求消息中包含对比数据，接收端应当在发送作为对此请求消息进行响应的 PUBLISH 报文中包含此对比数据。

3.3.2.3.6 对比数据

9 (0x09)，对比数据（Correlation Data）标识符。

跟随其后的是二进制数据。对比数据被请求消息发送端在收到响应消息时用来标识相应的请求。包含多个对比数据将造成协议错误（Protocol Error）。如果没有设置对比数据，则请求方（Requester）不需要任何对比数据。

服务端在收到应用消息时**必须**原封不动的把对比数据发送给所有的订阅者 [MQTT-3.3.2-16]。对比数据只对请求消息（Request Message）的发送端和响应消息（Response Message）的接收端有意义。

非规范评注

接收端收到包含响应主题和对比数据的应用消息时，发送以响应主题为主题名的 PUBLISH 报文作为响应消息。客户端在响应消息中应将对比数据作为 PUBLISH 报文的一部分原封不动的发送出去。

非规范评注

如果对客户端响应消息中的对比数据所做的任何更改会造成应用程序错误，则应当对对比数据进行加密/哈希，以便接收端能检测到对比数据是否被更改。

更多关于请求/响应的信息，请参考 4.10 节。

3.3.2.3.7 用户属性

38 (0x26)，用户属性（User Property）。

跟随其后的是 UTF-8 字符串对。用户属性（User Property）允许出现多次，以表示多个名字/值对，且相同的名字可以多次出现。

服务端在转发应用消息到客户端时**必须**原封不动的把所有的用户属性放在 PUBLISH 报文中 [MQTT-3.3.2-17]。服务端在转发应用消息时**必须**保持所有用户属性的先后顺序 [MQTT-3.3.2-18]。

非规范评注

此属性旨在提供一种传递应用层名称-值标签的方法，其含义和解释仅由负责发送和接收它们的应用程序所有。

3.3.2.3.8 订阅标识符

11 (0x0B)，订阅标识符（Subscription Identifier）标识符。
跟随其后的是一个变长字节整数表示的订阅标识符。

订阅标识符取值范围从 1 到 268,435,455。订阅标识符的值为 0 将造成协议错误。如果某条发布消息匹配了多个订阅，则将包含多个订阅标识符。这种情况下他们的顺序并不重要。

3.3.2.3.9 内容类型

3 (0x03)， 内容类型（Content Type）标识符。
跟随其后的是一个以 UTF-8 格式编码的字符串，用来描述应用消息的内容。内容类型**必须**是 UTF-8 编码的字符串，如 1.5.4 节 所定义 [MQTT-3.3.2-19]。
包含多个内容类型将造成协议错误（Protocol Error）。内容类型的值由发送应用程序和接收应用程序确定。

服务端**必须**把收到的应用消息中的内容类型原封不动的发送给所有的订阅者 [MQTT-3.3.2-20]。

非规范评注

UTF-8 编码字符串可以使用一个 MIME 内容类型字符串来描述应用消息的内容。由于发送程序和接收程序负责内容类型字符串的定义和解释，因此 MQTT 服务端只确保内容类型是有效的 UTF-8 编码的字符串，不会做其他方面的验证。

非规范评注

图 3-9 是一个 PUBLISH 示例报文，其中主题名为 *a/b*，报文标识符为 10，没有属性。

图 3-9 - PUBLISH 报文可变报头非规范示例

	说明	7	6	5	4	3	2	1	0
主题名									
byte 1	长度 MSB (0)	0	0	0	0	0	0	0	0
byte 2	长度 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
属性长度									
byte 8	无属性	0	0	0	0	0	0	0	0

3.3.3 PUBLISH 载荷

载荷包含将被发布的应用消息。载荷的内容和格式由应用程序指定。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 PUBLISH 行为

PUBLISH 报文的接收端**必须**按照 PUBLISH 报文中的 QoS 等级发送响应报文 [MQTT-3.3.4-1]。

表 3-3 - 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

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

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。PUBLISH 报文包含 SUBSCRIBE 报文中承载的订阅标识符--如果存在的话。

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叠，因此发布的消息可能会匹配多个主题过滤器。这种情况下，服务端**必须**按照所有匹配的订阅中最大的 QoS 等级把消息发送给客户端

[MQTT-3.3.4-2]。此外，服务端**可以**为每一个匹配的订阅按照订阅时的 QoS 等级，把消息副本分发给客户端。

如果客户端收到一个未经请求的应用消息（没有匹配任何订阅），且 QoS 大于客户端指定的最大服务质量（Maximum QoS），客户端使用包含原因码为 0x9B（不支持的 QoS 等级）的 DISCONNECT 报文断开连接，如 4.13 节 所述。

如果客户端在这些重叠的订阅中指定了订阅标识符，服务端在发布这些订阅相匹配的消息时**必须**包含这些订阅标识符 [MQTT-3.3.4-3]。如果服务端对这些重叠的订阅只发送一条相匹配的消息，服务端**必须**在 PUBLISH 报文中包含所有的相匹配的订阅标识符（如果存在），但没有顺序要求 [MQTT-3.3.4-4]。如果服务端对这些重叠的订阅**必须**分别发送相匹配的消息，则每个 PUBLISH 报文中含与订阅相匹配的订阅标识符（如果存在） [MQTT-3.3.4-5]。

可能存在客户端对同一个发布消息做了多次订阅，并且这些订阅中有多个订阅使用了相同的订阅标识符，这种情况下 PUBLISH 报文将携带多个相同的订阅标识符。

PUBLISH 报文中若包含服务端收到的 SUBSCRIBE 报文以外的订阅标识符，将造成协议错误（Protocol Error）。从客户端发送给服务端的 PUBLISH 报文**不能**包含订阅标识符 [MQTT-3.3.4-6]。

对于共享订阅，发送给某个客户端的 PUBLISH 报文中将只包含该客户端的 SUBSCRIBE 报文中发送的订阅标识符。

收到 PUBLISH 报文时，接收端的行为取决于报文的 QoS 等级，如 4.3 节 所述。

如果 PUBLISH 报文包含主题别名，接收端按照以下方式进行处理：

1) 主题别名为 0 或大于最大主题别名（Maximum Topic Alias），将造成协议错误（Protocol Error），接收端使用包含原因码为 0x94（主题别名无效）的 DISCONNECT 报文断开网络连接，如 4.13 节 所述。

2) 如果接收端已创建此主题别名的映射，
   a) 如果报文包含的主题名长度为 0，接收端使用主题别名对应的主题名处理此报文
   b) 如果报文包含的主题名长度不为 0，接收端使用此主题名处理此报文，并更新此主题别名映射到此主题名

3) 如果接收端还没有创建此主题别名的映射，
   a) 如果报文包含的主题名长度为 0，将造成协议错误，接收端使用包含原因码为 0x82（协议错误）的 DISCONNECT 报文断开网络连接，如 4.13 节 所述。
   b) 如果报文包含的主题名长度不为 0，接收端使用此主题名处理此报文，并为此报文中的主题别名和主题名创建映射关系

非规范评注

如果服务端向客户端分发应用消息时使用了不同的协议级别（比如 MQTT v3.1.1）-- 不支持属性或本规范提供的其他功能，应用消息中的某些信息将丢失，依赖于这些信息的应用程序可能无法正常工作。

客户端在收到服务端的 PUBACK，PUBCOMP 或包含原因码大于等于 128 的 PUBREC 报文之前，**不能**发送数量超过服务端的接收最大值（Receive Maximum）的 QoS 为 1 和 2 的 PUBLISH 报文 [MQTT-3.3.4-7]。服务端在发送 PUBACK 或 PUBCOMP 响应之前，如果收到数量超过客户端的接收最大值的 QoS 为 1 和 2 的 PUBLISH 报文，服务端使用包含原因码为 0x93（超出接收最大值）的 DISCONNECT 报文断开网络连接，如 4.13 节 所述。更多关于流量控制的信息，参考 4.9 节。

客户端**不能**延迟发送任何报文，除了 PUBLISH 报文--如果已发送且没有收到确认的 PUBLISH 报文数量已达到服务端的接收最大值（Receive Maximum） [MQTT-3.3.4-8]。接收最大值只应用于当前网络连接。

非规范评注

客户端可以选择发送少于服务端接收最大值的未经确认的 PUBLISH 报文，尽管它可以发送更多数量的报文。

非规范评注

客户端可以选择暂停发送 QoS 为 0 的报文，当其暂停发送了 QoS 为 1 和 2 的 PUBLISH 报文。

非规范评注

如果客户端在收到 CONNACK 之前发送 QoS 为 1 或 QoS 为 2 的 PUBLISH 报文，客户端有可能被服务器断开连接，因为它发送了超过服务端接收最大值数量的发布报文。

服务端在接收到客户端的 PUBACK，PUBCOMP 或包含原因码大于等于 128 的 PUBREC 报文之前，**不能**发送数量超过客户端的接收最大值（Receive Maximum）的 QoS 为 1 和 2 的 PUBLISH 报文 [MQTT-3.3.4-9]。客户端在发送 PUBACK 或 PUBCOMP 响应之前，如果收到数量超过服务端的接收最大值的 QoS 为 1 和 2 的 PUBLISH 报文，客户端使用包含原因码为 0x93（超出接收最大值）的 DISCONNECT 报文断开网络连接，如 4.13 节 所述。更多关于流量控制的信息，参考 4.9 节 。

服务端不能延迟发送任何报文，除了 PUBLISH 报文--如果已发送且没有收到确认的 PUBLISH 报文数量已到达客户端的接收最大值（Receive Maximum） [MQTT-3.3.4-10]。

非规范评注

服务端可以选择发送少于客户端接收最大值的未经确认的 PUBLISH 报文，尽管它可以发送更多数量的报文。

非规范评注

服务端可以选择暂停发送 QoS 为 0 的报文，当其暂停发送了 QoS 为 1 和 2 的 PUBLISH 报文。

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 PUBACK 固定报头

图 3-10 - 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							

剩余长度字段

表示可变报头的长度，用变长字节整数编码。

3.4.2 PUBACK 可变报头

PUBACK 可变报头按顺序包含以下字段：所确认的 PUBLISH 报文标识符，PUBACK 原因码，属性长度，属性（Properties）。属性编码规则如 2.22 节 所述。

图 3-11 – 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							
byte 3	PUBACK 原因码							
byte 4	属性长度							

3.4.2.1 PUBACK 原因码

PUBACK 可变报头第 3 字节是原因码（ Reason Code）。剩余长度为 2，则表示使用原因码 0x00（成功）。

表 3-4 - PUBACK 原因码

值	16 进制	原因码名称	说明
0	0x00	成功	消息被接收。QoS 为 1 的消息已发布。
16	0x10	无匹配的订阅者	消息被接收，但没有订阅者。只有服务端会发送此原因码。如果服务端得知没有匹配的订阅者，服务端**可以**使用此原因码代替 0x00（成功）。
128	0x80	未指明的错误	接收端不接受此消息，且不愿意透露错误原因或没有适用的原因码。
131	0x83	实现特定错误	PUBLISH 报文有效，但不被接收端所接受。
135	0x87	未授权	PUBLISH 报文未授权。
144	0x90	主题名无效	主题名格式正确，但未被客户端或服务端所接受。

145	0x91	报文标识符被占用	报文标识符正被占用。可能表明客户端和服务端之间的会话状态不匹配。
151	0x97	超出配额	已超出实现限制或管理限制。
153	0x99	载荷格式无效	载荷格式与载荷格式指示符不匹配。

服务端或客户端发送 PUBACK 报文时**必须**设置其中一种 PUBACK 原因码 [MQTT-3.4.2-1]。当原因码为 0x00（成功）且没有属性（Properties）时，原因码和属性长度可以被省略。在这种情况下，PUBACK 剩余长度为 2。

3.4.2.2 PUBACK 属性

3.4.2.2.1 属性长度

PUBACK 可变报头中属性长度被编码为变长字节整数。如果剩余长度小于 4 字节，则没有属性长度。

3.4.2.2.2 原因字符串

31 (0x1F)，原因字符串（Reason String）标识符。

跟随其后的是 UTF-8 编码的字符串，表示此次响应相关的原因。此原因字符串（Reason String）是为诊断而设计的可读字符串，**不能**被接收端所解析。

发送端使用此值向接收端提供附加信息。如果加上原因字符串之后的 PUBACK 报文长度超出了接收端指定的最大报文长度（Maximum Packet Size），则发送端**不能**发送此原因字符串 [MQTT-3.4.2-2]。包含多个原因字符串将造成协议错误（Protocol Error）。

3.4.2.2.3 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是 UTF-8 字符串对。此属性可用于提供包括诊断信息在内的附加信息。如果加上用户属性之后的 PUBACK 报文长度超出了接收端指定的最大报文长度（Maximum Packet Size），则发送端**不能**发送此属性 [MQTT-3.4.2-3]。用户属性（User Property）允许出现多次，以表示多个名字/值对，且相同的名字可以多次出现。

3.4.3 PUBACK 载荷

PUBACK 报文没有有效载荷。

3.4.4 PUBACK 行为

描述见 4.3.2 节。

3.5 PUBREC – 发布已接收（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 PUBREC 固定报头

图 3-12 - 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 (5)				保留			
	0	1	0	1	0	0	0	0
byte 2	剩余长度							

剩余长度字段

表示可变报头的长度，用变长字节整数编码。

3.5.2 PUBREC 可变报头

PUBREC 可变报头按顺序包含以下字段：所确认的 PUBLISH 报文标识符（Packet Identifier），PUBREC 原因码（Reason Code），属性（Properties）。属性的编码规则，如 2.2.2 节 所述。

图 3-13 - 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							
byte 3	PUBREC 原因码							
byte 4	属性长度							

3.5.2.1 PUBREC 原因码

PUBREC 可变报头第 3 字节是原因码（Reason Code）。如果剩余长度为 2，则表示使用原因码 0x00（成功）。

表 3-5 – PUBREC 原因码

值	16 进制	原因码名称	说明
0	0x00	成功	消息被接收。QoS 为 2 的消息已发布。
16	0x10	无匹配的订阅者	消息被接收，但没有订阅者。只有服务端会发送此原因码。 如果服务端得知没有匹配的订阅者，服务端**可以**使用此原因

			码代替 0x00（成功）。
128	0x80	未指明的错误	接收端不接受此消息，且不愿意透露错误原因或没有适用的原因码。
131	0x83	实现特定错误	PUBLISH 报文有效，但不被接收端所接受。
135	0x87	未授权	PUBLISH 报文未授权。
144	0x90	主题名无效	主题名格式正确，但未被客户端或服务端所接受。
145	0x91	报文标识符被占用	报文标识符正被占用。可能表明客户端和服务端之间的会话状态不匹配。
151	0x97	超出配额	已超出实现限制或管理限制。
153	0x99	载荷格式无效	载荷格式与载荷格式指示符不匹配。

服务端或客户端发送 PUBREC 报文时**必须**设置其中一种原因码 [MQTT-3.5.2-1]。当原因码为 0x00（成功）且没有属性（Properties）时，原因码和属性长度可以被省略。在这种情况下，PUBREC 剩余长度为 2。

3.5.2.2 PUBREC 属性

3.5.2.2.1 属性长度

PUBREC 可变报头的属性长度被编码为变长字节整数。如果剩余长度小于 4，则表示没有属性长度字段。

3.5.2.2.2 原因字符串

31 (0x1F)，原因字符串（Reason String）标识符。

跟随其后的是 UTF-8 编码的字符串，表示此次响应相关的原因。此原因字符串（Reason String）是为诊断而设计的可读字符串，**不应该**被接收端所解析。

发送端使用此值向接收端提供附加信息。如果加上原因字符串之后的 PUBREC 报文长度超出了接收端指定的最大报文长度（Maximum Packet Size），则发送端**不能**发送此属性 [MQTT-3.5.2-2]。包含多个原因字符串将造成协议错误（Protocol Error）。

3.5.2.2.3 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是 UTF-8 字符串对。此属性可用于提供包括诊断信息在内的附加信息。如果加上用户属性之后的 PUBREC 报文长度超出了接收端指定的最大报文长度（Maximum Packet Size），则发送端**不能**发送此属性 [MQTT-3.5.2-3]。用户属性（User Property）允许出现多次，以表示多个名字/值对，且相同的名字可以多次出现。

3.5.3 PUBREC 载荷

PUBREC 报文没有有效载荷。

3.5.4 PUBREC 行为

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 PUBREL 固定报头

图 3-14 – 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							

PUBREL 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，被编码为变长字节整数。

3.6.2 PUBREL 可变报头

PUBREL 报文的可变报头按顺序包含以下字段：所确认的 PUBREC 报文标识符，PUBREL 原因码，属性（Properties）。属性的编码规则如 2.2.2 节 所述。

图 3-15 – 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							
byte 3	PUBREL 原因码							
byte 4	属性长度							

3.6.2.1 PUBREL 原因码

可变报头第 3 字节是 PUBREL 原因码。如果剩余长度为 2，则表示使用原因码 0x00 （成功）。

表 3-6 - PUBREL 原因码

值	16 进制	原因码名称	说明
0	0x00	成功	消息已释放。
146	0x92	报文标识符未发现	未知的报文标识符。会话恢复阶段这并非错误，但其他时间这表明服务端和客户端的会话状态不匹配。

客户端或服务端发送 PUBREL 报文时**必须**设置其中一种 PUBREL 原因码 [MQTT-3.6.2-1]。当原因码为 0x00（成功）且没有属性（Properties）时，原因码和属性长度可以被省略。在这种情况下，PUBREL 剩余长度为 2。

3.6.2.2 PUBREL 属性

3.6.2.2.1 属性长度

PUBREL 报文可变报头中的属性长度被编码为变长字节整数。如果剩余长度小于 4，则表示没有属性长度字段。

3.6.2.2.2 原因字符串

31 (0x1F)，原因字符串（Reason String）标识符。

跟随其后的是 UTF-8 编码的字符串，表示此次响应相关的原因。此原因字符串（Reason String）是为诊断而设计的可读字符串，**不应该**被接收端所解析。

发送端使用此值向接收端提供附加信息。如果加上原因字符串之后的 PUBREL 报文长度超出了接收端指定的最大报文长度（Maximum Packet Size），则发送端**不能**发送此原因字符串 [MQTT-3.6.2-2]。包含多个原因字符串将造成协议错误（Protocol Error）。

3.6.2.2.3 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是 UTF-8 字符串对。此属性可用于提供包括诊断信息或关于 PUBREL 的信息。如果加上用户属性之后的 PUBREL 报文长度超出了接收端指定的最大报文长度（Maximum Packet Size），则发送端**不能**发送此属性 [MQTT-3.6.2-3]。用户属性（User Property）允许出现多次，以表示多个名字/值对，且相同的名字可以多次出现。

3.6.3 PUBREL 载荷

PUBREL 报文没有有效载荷。

3.6.4 PUBREL 行为

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 PUBCOMP 固定报头

图 3-16 – 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							

剩余长度字段

表示可变报头的长度，编码为变长字节整数。

3.7.2 PUBCOMP 可变报头

PUBCOMP 报文可变报头按顺序包含以下字段：所确认的 PUBREL 报文标识符，PUBCOMP 原因码，属性。属性（Properties）编码规则如 2.2.2 节 所述。

图 3-17 - 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							
byte 3	PUBCOMP 原因码							
byte 4	属性长度							

3.7.2.1 PUBCOMP 原因码

可变报头第 3 字节是 PUBCOMP 原因码。如果剩余长度为 2，则表示使用原因码 0x00（成功）。

表 3-7 – PUBCOMP 原因码

值	16 进制	原因码名称	说明

0	0x00	成功	报文标识符已释放。QoS 2 消息已完成发布。
146	0x92	报文标识符未发现	未知的报文标识符。会话恢复阶段这并非错误，但其他时间这表明服务端和客户端的会话状态不匹配。

服务端或客户端发送 PUBCOMP 报文时**必须**设置一种 PUBCOMP 原因码 [MQTT-3.7.2-1]。当原因码为 0x00（成功）且没有属性（Properties）时，原因码和属性长度可以被省略。在这种情况下，PUBCOMP 剩余长度为 2。

3.7.2.2 PUBCOMP 属性

3.7.2.2.1 属性长度

PUBCOMP 报文可变报头中的属性长度被编码为变长字节整数。如果剩余长度小于 4，则表示没有属性长度字段。

3.7.2.2.2 原因字符串

31 (0x1F)，原因字符串（Reason String）标识符。

跟随其后的是 UTF-8 编码的字符串，表示此次响应相关的原因。此原因字符串（Reason String）是为诊断而设计的可读字符串，**不应该**被接收端所解析。

发送端使用此值向接收端提供附加信息。如果加上原因字符串之后的 PUBCOMP 报文长度超出了接收端指定的最大报文长度（Maximum Packet Size），则发送端**不能**发送此原因字符串 [MQTT-3.7.2-2]。包含多个原因字符串将造成协议错误（Protocol Error）。

3.7.2.2.3 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是 UTF-8 字符串对。此属性可用于提供诊断信息或关于其他信息。如果加上用户属性之后的 PUBCOMP 报文长度超出了接收端指定的最大报文长度（Maximum Packet Size），则发送端**不能**发送此属性 [MQTT-3.7.2-3]。用户属性（User Property）允许出现多次，以表示多个名字/值对，且相同的名字可以多次出现。

3.7.3 PUBCOMP 载荷

PUBCOMP 报文没有有效载荷。

3.7.4 PUBCOMP 行为

描述见 4.3.3 节。

3.8 SUBSCRIBE - 订阅请求

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（Subscription）注册客户端所感兴趣的一个或多个主题。服务端向客户端发送 PUBLISH 报文以转发被发布到符合这些订阅主题的应用消息。SUBSCRIBE 报文同样（为每个订阅）指定了服务端可以向其发送的应用消息最大 QoS 等级。

3.8.1 SUBSCRIBE 固定报头

图 3-18 - 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 报文固定报头第 3，2，1，0 比特位是保留位，**必须**被设置为 0，0，1，0。服务端**必须**将其他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

表示可变报头的长度加上有效载荷的长度，被编码为变长字节整数。

3.8.2 SUBSCRIBE 可变报头

SUBSCRIBE 报文可变报头按顺序包含以下字段：报文标识符（Packet Identifier），属性（Properties）。2.2.1 节 提供了更多关于报文标识符的信息。属性的编码规则如 2.2.2 节 所述。

非规范示例

图 3-19 展示了一个包含报文标识符为 10，且没有属性的 SUBSCRIBE 可变报头。

图 3-19 – SUBSCRIBE 可变报头示例

	说明	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
byte 3	属性长度 (0)	0	0	0	0	0	0	0	0

3.8.2.1 SUBSCRIBE 属性

3.8.2.1.1 属性长度

SUBSCRIBE 报文可变报头中的属性长度被编码为变长字节整数。

3.8.2.1.2 订阅标识符

11 (0x0B)，订阅标识符（Subscription Identifier）标识符。

跟随其后的是一个变长字节整数表示订阅标识符。订阅标识符取值范围从 1 到 268,435,455。订阅标识符的值为 0 或包含多个订阅标识符将造成协议错误（Protocol Error）。

订阅标识符与 SUBSCRIBE 报文所创建或修改的订阅（Subscription）相关联。如果包含订阅标识符，它将与订阅一起被存储。如果未指定此属性，则订阅被存储时将不包含订阅标识符。

更多关于订阅标识符的处理信息，参考 3.8.3.1 节 。

3.8.2.1.3 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是 UTF-8 字符串对。

用户属性允许出现多次，以表示多个名字/值对。同样的名字允许出现多次。

> **非规范评注**
>
> SUBSCRIBE 报文的用户属性可以被客户端用来向服务端发送订阅相关的属性。本规范不定义这些属性的意义。

3.8.3 SUBSCRIBE 载荷

SUBSCRIBE 报文的载荷包含一列主题过滤器，指明客户端希望订阅的主题。主题过滤器**必须**为 UTF-8 编码的字符串 [MQTT-3.8.3-1]。每个主题过滤器之后跟着一个订阅选项（Subscription Options）字节。

载荷**必须**包含至少一个主题过滤器/订阅选项对 [MQTT-3.8.3-2]。不包含载荷的 SUBSCRIBE 报文将造成协议错误（Protocol Error）。错误处理信息，参考 4.13 节。

3.8.3.1 订阅选项

订阅选项的第 0 和 1 比特代表最大服务质量字段。此字段给出服务端可以向此客户端发送的应用消息的最大 QoS 等级。最大服务质量字段为 3 将造成协议错误（Protocol Error）。

订阅选项的第 2 比特表示非本地（No Local）选项。值为 1，表示应用消息不能被转发给发布此消息的客户标识符 [MQTT-3.8.3-3]。共享订阅时把非本地选项设为 1 将造成协议错误（Protocol Error） [MQTT-3.8.3-4]。

订阅选项的第 3 比特表示发布保留（Retain As Published）选项。值为 1，表示向此订阅转发应用消息时保持消息被发布时设置的保留（RETAIN）标志。值为 0，表示向此订阅转发应用消息时把保留标志设置为 0。当订阅建立之后，发送保留消息时保留标志设置为 1。

订阅选项的第 4 和 5 比特表示保留操作（Retain Handling）选项。此选项指示当订阅建立时，是否发送保留消息。此选项不影响之后的任何保留消息的发送。如果没有匹配主题过滤器的保留消息，则此选项所有值的行为都一样。值可以设置为：

0 = 订阅建立时发送保留消息

1 = 订阅建立时，若该订阅当前不存在则发送保留消息

2 = 订阅建立时不要发送保留消息

保留操作的值设置为 3 将造成协议错误（Protocol Error）。

订阅选项的第 6 和 7 比特为将来所保留。服务端**必须**把此保留位非 0 的 SUBSCRIBE 报文当做无效报文 [MQTT-3.8.3-5]。

非规范评注

非本地（No Local）和发布保留（Retain As Published）订阅选项在客户端把消息发送给其他服务端的情况下，可以被用来实现桥接。

非规范评注

已存在订阅的情况下不发送保留消息是很有用的，比如重连完成时客户端不确定订阅是否在之前的会话连接中被创建。

非规范评注

不发送保存的保留消息给新创建的订阅是很有用的，比如客户端希望接收变更通知且不需要知道最初的状态。

非规范评注

对于某个指示其不支持保留消息的服务端，发布保留和保留处理选项的所有有效值都将得到同样的结果：订阅时不发送任何保留消息，且所有消息的保留标志都会被设置为 0。

图 3-20 – SUBSCRIBE 报文载荷格式

说明	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							
订阅选项								
	保留位		保留处理		RAP	NL	QoS	
byte N+1	0	0	X	X	X	X	X	X

RAP 指发布保留（Retain as Published）。

NL 指非本地（No Local）。

非规范示例

图 3-21 展示了 SUBSCRIBE 载荷示例，包含 2 个主题过滤器：第一个为“a/b”，QoS 为 1；第二个为“c/d” QoS 为 2。

图 3-21 - 载荷字节格式非规范示例

	说明	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
主题过滤器									
byte 1	长度 MSB (0)	0	0	0	0	0	0	0	0
byte 2	长度 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
订阅选项									
byte 6	订阅选项 (1)	0	0	0	0	0	0	0	1
主题过滤器									
byte 7	长度 MSB (0)	0	0	0	0	0	0	0	0
byte 8	长度 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
订阅选项									
byte 12	订阅选项 (2)	0	0	0	0	0	0	1	0

3.8.4 SUBSCRIBE 行为

当服务端收到来自客户端的 SUBSCRIBE 报文时，**必须**使用 SUBACK 报文作为相应 [MQTT-3.8.4-1]。
SUBACK 报文**必须**和待确认的 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相匹配的 PUBLISH 报文。

如果服务端收到的 SUBSCRIBE 报文中的一个主题过滤器与当前会话的一个非共享订阅（Non-shared Subscription）相同，那么**必须**使用新的订阅替换现存的订阅 [MQTT-3.8.4-3]。新订阅的主题过滤器与之前的订阅相同，但其订阅选项可能不同。如果保留处理选项为 0，任何匹配该主题过滤器的保留消息**必须**被重发，但替换订阅不能造成应用消息的丢失 [MQTT-3.8.4-4]。

如果服务端收到的非共享主题过滤器（Non-shared Topic Filter）不同于当前会话的任何主题过滤器，一个新的非共享订阅将被创建。如果保留处理选项不为 2，所有相匹配的保留消息将发送给客户端。

如果服务端收到的主题过滤器与服务端已存在的某个共享订阅（Shared Subscription）主题过滤器相同，则将此会话添加到该共享订阅中。不发送任何保留消息。

如果服务端收到的共享订阅主题过滤器（Shared Subscription Topic Filter）与任何已存在的共享订阅主题过滤器都不同，一个新的共享订阅将被创建。将此会话作为订阅者添加到该共享订阅。不发送任何保留消息。

更多关于共享订阅的细节，参考 4.8 节。

如果服务端收到的 SUBSCRIBE 报文包含多个主题过滤器，服务端**必须**当做收到一系列多个 SUBSCRIBE 报文来处理--除了将它们的响应组合为单个 SUBACK 响应 [MQTT-3.8.4-5]。

服务端发送给客户端的 SUBACK 报文**必须**为每一个主题过滤器/订阅选项对包含一个原因码 [MQTT-3.8.4-6]。 此原因码**必须**说明为该订阅授予的最大 QoS 等级，或指示订阅失败 [MQTT-3.8.4-7]。服务端可能授予了低于订阅者所请求的最大 QoS 等级。响应该订阅的应用消息 QoS 等级**必须**为该消息发布时的 QoS 等级和服务端授予的最大 QoS 等级二者最小值 [MQTT-3.8.4-8]。在原始消息发布的 QoS 等级为 1，且授予的最大 QoS 等级为 0 的情况下，服务端允许发送重复的消息副本给订阅者（？）。

非规范评注

如果订阅客户端的某个主题过滤器已被授予的最大 QoS 等级为 1，那么匹配此过滤器的 QoS 等级为 0 的应用消息按照 QoS 等级为 0 分发给此客户端。这意味着客户端最多只能收到该消息的一个副本。另一方面，发布到相同主题的 QoS 等级为 2 的消息，其 QoS 等级被服务端降级为 1 以便分发给该客户端。因此该客户端可能收到此消息的多个副本。

非规范评注

如果订阅客户端被授予的最大 QoS 等级为 0，那么按照 QoS 等级为 2 发布的应用消息在繁忙时可能会丢失，但服务端不应该发送重复的消息副本。发布到相同主题的 QoS 等级为 1 的消息，分发给该客户端时可能会丢失或重复。

非规范评注

使用 QoS 等级 2 订阅某个主题过滤器，等于是说：*我想要按照消息被发布时的 QoS 等级接收匹配此过滤器的消息*。这意味着发布者负责决定消息可以被发布的最大 QoS 等级，但订阅端可以要求服务端降低该消息的 QoS 到更适合它的等级。

订阅标识符是服务端的会话状态的一部分，并将在收到 PUBLISH 报文时返回给客户端。当服务端收到客户端的 UNSUBSCRIBE 报文时，服务端将此会话标识符从服务端的会话状态中移除：当服务端收到客户端的 UNSUBSCRIBE 报文，当服务端收到客户端对同样主题过滤器的 SUBSCRIBE 报文但订阅标识符不同或没有订阅标识符，或者当服务端在 CONNACK 报文中将会话存在标志设置为 0。

订阅标识符不构成客户端的会话状态的一部分。在一个有用的实现中，客户端将订阅标识符与其他客户端状态相关联，此客户端状态将被移除：当客户端取消订阅，当客户端以不同的订阅标识符或没有订阅标识符订阅同样的主题过滤器，或者当客户端收到的 CONNACK 报文中会话存在标志被设置为 0。

服务端在重传的 PUBLISH 报文中无需使用同一组订阅标识符。客户端可以通过发送包含与当前会话已存在的主题过滤器的 SUBSCRIBE 报文进行重新订阅。如果客户端在 PUBLISH 报文初传之后重新订阅并使用了不同的订阅标识符，允许服务端在任何重传中使用初传所包含的订阅标识符，或者在重传中使用此新的订阅标识符。不允许服务端在发送了包含新的订阅标识符的 PUBLISH 报文之后再次使用旧的订阅标识符。

非规范评注

使用场景，用以阐述订阅标识符：

- 客户端实现指示某条发布消息匹配多个订阅的编程接口，客户端实现每次订阅时生成新的订阅标识符。如果返回的发布消息包含多个订阅标识符，则该发布消息匹配多个订阅。
- 客户端实现允许订阅者将消息定向到其相关联的订阅的回调，客户端实现生成映射到唯一回调的订阅标识符。收到某条发布消息时，使用订阅标识符决定触发哪一个回调。
- 客户端实现在发布消息时返回程序用于订阅的主题字符串，为此客户端生成一个唯一标识了该主题过滤器的标识符。收到某条发布消息时，客户端实现使用此标识符查找原始主题过滤器，并将主题过滤器返回给其应用程序。
- 网关（Gateway）将从服务端收到的发布消息转发给向该网关做了订阅的客户端，网关实现维护其收到的每个唯一的订阅过滤器到其收到的一组客户标识符--订阅标识符对的映射，网关对它转发给服务端的每个主题过滤器生成一个唯一的标识符。收到某条发布消息时，网关使用从服务端收到的订阅标识符查找对应的客户标识符--订阅标识符对，并把它们加入发送给客户端的 PUBLISH 报文中。如果上游服务端因为消息匹配了多个订阅而发送了多个 PUBLISH 报文，则此行为将反映到客户端。

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 报文包含一个原因码列表，用于指定授予的最大 QoS 等级或 SUBSCRIBE 报文所请求的每个订阅发生的错误。

3.9.1 SUBACK 固定报头

图 3-22 - 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

可变报头长度加上有效载荷长度，编码为变长字节整数。

3.9.2 SUBACK 可变报头

SUBACK 报文可变报头按顺序包含以下字段：所确认的 SUBSCRIBE 报文标识符，属性（Properties）。

3.9.2.1 SUBACK 属性

3.9.2.1.1 属性长度

SUBACK 可变报头中的属性长度被编码为变长字节整数。

3.9.2.1.2 原因字符串

31 (0x1F)，原因字符串（Reason String）标识符。

跟随其后的是 UTF-8 编码的字符串，表示此次响应相关的原因。此原因字符串（Reason String）是为诊断而设计的可读字符串，**不应该**被客户端所解析。

服务端使用此值向客户端提供附加信息。如果加上原因字符串之后的 SUBACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此原因字符串 [MQTT-3.9.2-1]。包含多个原因字符串将造成协议错误（Protocol Error）。

3.9.2.1.3 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是 UTF-8 字符串对。此属性可用于向客户端提供包括诊断信息在内的附加信息。如果加上用户属性之后的 SUBACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此属性 [MQTT-3.9.2-2]。用户属性（User Property）允许出现多次，以表示多个名字/值对，且相同的名字可以多次出现。

图 3-23 - 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 SUBACK 载荷

有效载荷包含一个原因码列表。每个原因码对应 SUBSCRIBE 报文中的一个被确认的主题过滤器。
SUBACK 报文中的原因码顺序**必须**与 SUBSCRIBE 报文中的主题过滤器顺序相匹配 [MQTT-3.9.3-1]。

表 3-8 - 订阅原因码

值	16 进制	原因码名称	说明
0	0x00	授予 QoS 等级 0	订阅被接受且最大 QoS 等级为 0。可能低于所请求的 QoS 等级。
1	0x01	授予 QoS 等级 1	订阅被接受且最大 QoS 等级为 1。可能低于所请求的 QoS 等级。
2	0x02	授予 QoS 等级 2	订阅被接受且任何 QoS 等级都将被发送给此订阅。
128	0x80	未指明错误	订阅未被接受，且服务端不愿意透露原因或没有适用的原因码。
131	0x83	实现特定错误	SUBSCRIBE 有效但不被服务端所接受。
135	0x87	未授权	客户端未被授权做此订阅。
143	0x8F	主题过滤器无效	主题过滤器格式正确，但不被允许。
145	0x91	报文标识符已占用	指定的报文标识符正在被使用中。
151	0x97	超出配额	已超出实现限制或管理限制。
158	0x9E	共享订阅不支持	服务端不支持此客户端进行共享订阅。
161	0xA1	订阅标识符不支持	服务端不支持订阅标识符；订阅标识符不被接受。
162	0xA2	通配符订阅不支持	服务端不支持通配符订阅；订阅未被接受。

服务端发送 SUBACK 报文时**必须**对收到的每一个主题过滤器设置一种原因码 [MQTT-3.9.3-2]。

非规范评注

对于 SUBSCRIBE 报文中的每个主题过滤器，总有一个对应的原因码。如果原因码不是针对某个特定的主题过滤器（比如 0x91（报文标识符已占用）），则对每个主题过滤器都使用此原因码。

3.10 UNSUBSCRIBE – 取消订阅请求

客户端发送 UNSUBSCRIBE 报文给服务端，用于取消订阅主题。

3.10.1 UNSUBSCRIBE 固定报头

图 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 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

等于可变报头长度（2 字节）加上有效载荷长度，编码为变长字节整数。

3.10.2 UNSUBSCRIBE 可变报头

UNSUBSCRIBE 报文可变报头按顺序包含以下字段：报文标识符和属性（Properties）。2.2.1 节 提供了有关报文标识符的更多信息。属性的编码规则，如 2.2.2 节 所述。

3.10.2.1 UNSUBSCRIBE 属性

3.10.2.1.1 属性长度

SUBSCRIBE 可变报头中属性的长度被编码为变长字节整数。

3.10.2.1.2 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是一个 UTF-8 字符串对。

用户属性允许出现多次，以表示多个名字/值对。相同的名字可以出现多次。

> **非规范评注**
>
> UNSUBSCRIBE 报文中的用户属性可以被客户端用来向服务端发送订阅相关的属性。本规范不定义这些属性的意义。

3.10.3 UNSUBSCRIBE 载荷

UNSUBSCRIBE 报文有效载荷包含一列客户端希望取消订阅的主题过滤器。UNSUBSCRIBE 报文中的主题过滤器**必须**为 1.5.4 节 所述的 UTF-8 编码字符串 [MQTT-3.10.3-1]，且连续填充。

UNSUBSCRIBE 报文有效载荷**必须**包含至少一个主题过滤器 [MQTT-3.10.3-2]。不包含有效载荷的 UNSUBSCRIBE 报文将造成协议错误（Protocol Error）。错误处理信息，参考 4.13 节。

> **非规范示例**
>
> 图 3-30 展示了 UNSUBSCRIBE 报文的载荷示例，包括两个主题过滤器 “a/b”和“c/d”。

图 3-30 - 载荷字节格式非规范示例

	说明	7	6	5	4	3	2	1	0
主题过滤器									
byte 1	长度 MSB (0)	0	0	0	0	0	0	0	0
byte 2	长度 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	长度 MSB (0)	0	0	0	0	0	0	0	0
byte 7	长度 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 UNSUBSCRIBE 行为

服务端**必须**对客户端的 UNSUBSCRIBE 报文中提供的主题过滤器（不管是否包含通配符）逐个字符与当前持有的主题过滤器集进行比较。如果任何过滤器完全匹配，则**必须**删除其拥有的订阅 [MQTT-3.10.4-1]，否则不会进行额外的处理。

当服务端收到 UNSUBSCRIBE 报文：

- 它**必须**停止添加为了交付给客户端的与主题过滤器相匹配的任何新消息 [MQTT-3.10.4-2]。
- 它**必须**完成任何已经开始发送给客户端的、与主题过滤器相匹配的、QoS 等级为 1 或 2 的消息 [MQTT-3.10.4-3]。
- 它**可以**继续交付任何为交付给客户端而缓存的消息。

服务端**必须**发送 UNSUBACK 报文以响应客户端的 UNSUBSCRIBE 请求 [MQTT-3.10.4-4]。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 [MQTT-3.10.4-5]。

如果服务端收到的 UNSUBSCRIBE 报文包含多个主题过滤器，服务端**必须**当做收到一系列多个 UNSUBSCRIBE 报文来处理--除了将它们的响应组合为单个 SUBACK 响应 [MQTT-3.10.4-6]。

如果某个主题过滤器代表一个共享订阅，此会话将被从该共享订阅中删除。如果此会话是该共享订阅所关联的唯一会话，该共享订阅被删除。共享订阅的处理，参考 4.8.2 节。

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 UNSUBACK 固定报头

图 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							

剩余长度字段

等于可变报头的长度加上有效载荷的长度，编码为变长字节整数。

3.11.2 UNSUBACK 可变报头

UNSUBACK 报文可变报头按顺序包含以下字段：所确认的 UNSUBSCRIBE 报文标识符和属性（ Properties）。属性的编码规则如 2.2.2 节 所述。

图 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							

3.11.2.1 UNSUBACK 属性

3.11.2.1.1 属性长度

UNSUBACK 报文可变报头中的属性的长度被编码为变长字节整数。

3.11.2.1.2 原因字符串

31 (0x1F)，原因字符串（Reason String）标识符。

跟随其后的是 UTF-8 编码的字符串，表示此次响应相关的原因。此原因字符串（Reason String）是为诊断而设计的可读字符串，**不应该**被客户端所解析。

服务端使用此值向客户端提供附加信息。如果加上原因字符串之后的 UNSUBACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此原因字符串 [MQTT-3.11.2-1]。包含多个原因字符串将造成协议错误（Protocol Error）。

3.11.2.1.3 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是 UTF-8 字符串对。此属性可用于向客户端提供包括诊断信息在内的附加信息。如果加上用户属性之后的 UNSUBACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此属性 [MQTT-3.11.2-2]。用户属性（User Property）允许出现多次，以表示多个名字/值对，且相同的名字可以多次出现。

3.11.3 UNSUBACK 载荷

有效载荷包含一个原因码列表。每个原因码对应 UNSUBSCRIBE 报文中的一个被确认的主题过滤器。UNSUBACK 报文中的原因码顺序**必须**与 UNSUBSCRIBE 报文中的主题过滤器顺序相匹配 [MQTT-3.11.3-1]。

单字节无符号取消订阅原因码的值如下所示。服务端发送 UNSUBACK 报文时对于每个收到的主题过滤器，**必须**使用一个取消订阅原因码 [MQTT-3.11.3-2]。

表 3-9 - 取消订阅原因码

值	16 进制	原因码名称	说明
0	0x00	成功	订阅已被删除。
17	0x11	订阅未发现	没有该客户端匹配的主题过滤器被使用。
128	0x80	未指定错误	取消订阅不能被完成且服务端不愿意透露原因或没有其他适用的原因码。
131	0x83	实现指定错误	UNSUBSCRIBE 报文有效，但服务端不接受。
135	0x87	未授权	客户端未被授权进行取消订阅。
143	0x8F	主题过滤器无效	主题过滤器格式正确，但不被允许。
145	0x91	报文标识符已占用	指定的报文标识符正在被使用中。

非规范评注

对于 UNSUBSCRIBE 报文中的每个主题过滤器，总有一个对应的原因码。如果原因码不是针对某个特定的主题过滤器（比如 0x91（报文标识符已占用）），则对每个主题过滤器都使用此原因码。

3.12 PINGREQ – PING 请求

客户端发送 PINGREQ 报文给服务端，可被用于：

- 在没有任何其他 MQTT 控制报文从客户端发给服务端时，告知服务端客户端还活着。
- 请求服务端发送响应以确认服务端还活着。
- 使用网络已确认网络连接没有断开。

此报文被用在保持连接（Keep Alive）的处理中。详细信息，参考 3.1.2.10 节。

3.12.1 PINGREQ 固定报头

图 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 PINGREQ 可变报头

PINGREQ 报文没有可变报头。

3.12.3 PINGREQ 载荷

PINGREQ 报文没有有效载荷

3.12.4 PINGREQ 行为

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 – PING 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。

此报文被用在保持连接（Keep Alive）的处理中。详细信息，参考 3.1.2.10 节。

3.13.1 PINGRESP 固定报头

图 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 PINGRESP 可变报头

PINGRESP 报文没有可变报头。

3.13.3 PINGRESP 载荷

PINGRESP 报文没有有效载荷。

3.13.4 PINGRESP 行为

客户端收到此报文时不做任何处理。

3.14 DISCONNECT – 断开通知

DISCONNECT 报文是客户端发给服务端的最后一个 MQTT 控制报文。表示客户端为什么断开网络连接的原因。客户端和服务端在关闭网络连接之前**可以**发送一个 DISCONNECT 报文。如果在客户端没有首先发送包含原因码为 0x00（正常断开）DISCONNECT 报文并且连接包含遗嘱消息的情况下，遗嘱消息会被发布。更多细节，参考 3.1.2.5 节。

服务端**不能**发送 DISCONNECT 报文，直到它发送了包含原因码小于 0x80 的 CONNACK 报文之后 [MQTT-3.14.0-1]。

3.14.1 DISCONNECT 固定报头

图 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)				保留字段			
	1	1	1	0	0	0	0	0
byte 2	剩余长度							

服务端或客户端**必须**验证所有的保留位都被设置为 0，如果他们不为 0，发送包含原因码为 0x81（无效报文）的 DISCONNECT 报文，如 4.13 节所述 [MQTT-3.14.1-1]。

剩余长度字段

等于可变报头的长度，编码为变长字节整数。

3.14.2 DISCONNECT 可变报头

DISCONNECT 报文的可变报头按顺序包含以下字段：断开原因码，属性（Properties）。属性的编码规则如 2.2.2 节 所述。

3.14.2.1 断开原因码

可变报头的第 1 个字节是断开原因码。如果剩余长度小于 1，则表示使用原因码 0x00（正常断开）。

单字节无符号断开原因码字段如下所示。

表 3-10 – 断开原因值

值	16 进制	原因码名称	发送端	说明
0	0x00	正常断开	客户端或服务端	正常关闭连接。不发送遗嘱。
4	0x04	包含遗嘱消息的断开	客户端	客户端希望断开但也需要服务端发布它的遗嘱消息。
128	0x80	未指定错误	客户端或服务端	连接被关闭，但发送端不愿意透露原因，或者没有其他适用的原因码。
129	0x81	无效的报文	客户端或服务端	收到的报文不符合本规范。
130	0x82	协议错误	客户端或服务端	收到意外的或无序的报文。
131	0x83	实现指定错误	客户端或服务端	收到的报文有效，但根据实现无法进行处理。
135	0x87	未授权	服务端	请求没有被授权
137	0x89	服务端正忙	服务端	服务端正忙且不能继续处理此客户端的请求。
139	0x8B	服务正关闭	服务端	服务正在关闭。
141	0x8D	保持连接超时	服务端	连接因为在超过 1.5 倍的保持连接时间内没有收到任何报文而关闭。
142	0x8E	会话被接管	服务端	另一个使用了相同的客户标识符的连接已建立，导致此连接关闭。
143	0x8F	主题过滤器无效	服务端	主题过滤器格式正确，但不被服务端所接受。
144	0x90	主题名无效	客户端或服务端	主题名格式正确，但不被客户端或服务端所接受。
147	0x93	超出接收最大值	客户端或服务端	客户端或服务端收到了数量超过接收最大值的未发送 PUBACK 或 PUBCOMP 的发布消息。
148	0x94	主题别名无效	客户端或服务端	客户端或服务端收到的 PUBLISH 报文包含的主题别名大于其在 CONNECT 或 CONNACK 中发送的主题别名最大值。
149	0x95	报文过大	客户端或服务端	报文长度大于此客户端或服务端的最大报文长度。

150	0x96	消息速率过高	客户端或服务端	收到的数据速率太高。
151	0x97	超出配额	客户端或服务端	已超出实现限制或管理限制。
152	0x98	管理操作	客户端或服务端	连接因为管理操作被关闭。
153	0x99	载荷格式无效	客户端或服务端	载荷格式与指定的载荷格式指示符不匹配。
154	0x9A	不支持保留	服务端	服务端不支持保留消息。
155	0x9B	不支持的 QoS 等级	服务端	客户端指定的 QoS 等级大于 CONNACK 报文中指定的最大 QoS 等级。
156	0x9C	（临时）使用其他服务端	服务端	客户端应该临时使用其他服务端。
157	0x9D	服务端已（永久）移动	服务端	服务端已移动且客户端应该永久使用其他服务端。
158	0x9E	不支持共享订阅	服务端	服务端不支持共享订阅。
159	0x9F	超出连接速率限制	服务端	此连接因为连接速率过高而被关闭。
160	0xA0	最大连接时间	服务端	超出为此连接授予的最大连接时间。
161	0xA1	不支持订阅标识符	服务端	服务端不支持订阅标识符；订阅未被接受。
162	0xA2	不支持通配符订阅	服务端	服务端不支持通配符订阅；订阅未被接受。

客户端或服务端发送 DISCONNECT 报文时**必须**使用一种 DISCONNECT 原因码 [MQTT-3.14.2-1]。如果原因码为 0x00（正常断开）且没有属性，原因码和属性长度可以被省略。这种情况下 DISCONNECT 报文剩余长度为 0。

非规范评注

DISCONNECT 报文用于指示断开的原因，例如没有确认报文（比如 QoS 等级 0 的发布消息）或当客户端或服务端不能继续处理连接。

非规范评注

客户端可以使用这些信息来决定是否重新连接，以及在重新尝试之前应该等待多长时间。

3.14.2.2 DISCONNECT 属性

3.14.2.2.1 属性长度

DISCONNECT 报文可变报头中的属性（Properties）的长度被编码为变长字节整数。如果剩余长度小于 2，属性长度使用 0。

3.14.2.2.2 会话过期间隔

17 (0x11)，会话过期间隔（Session Expiry Interval）标识符。

跟随其后的是用四字节整数表示的以秒为单位的会话过期间隔（Session Expiry Interval）。包含多个会话过期间隔将造成协议错误（Protocol Error）。

如果没有设置会话过期间隔，则使用 CONNECT 报文中的会话过期间隔。

会话过期间隔**不能**由服务端的 DISCONNECT 报文发送 [MQTT-3.14.2-2]。

如果 CONNECT 报文中的会话过期间隔为 0，则客户端在 DISCONNECT 报文中设置非 0 会话过期间隔将造成协议错误（Protocol Error）。如果服务端收到这种非 0 会话过期间隔，则不会将其视为有效的 DISCONNECT 报文。服务端使用包含原因码为 0x82（协议错误）的 DISCONNECT 报文，如 4.13 节 所述。

3.14.2.2.3 原因字符串

31 (0x1F)，原因字符串（Reason String）标识符。

跟随其后的是 UTF-8 编码字符串表示断开原因。此原因字符串是为诊断而设计的可读字符串，**不应该**被接收端所解析。

如果此属性使得 DISCONNECT 报文的长度超出了接收端指定的最大报文长度，则发送端**不能**发送此属性 [MQTT-3.14.2-3]。包含多个原因字符串将造成协议错误（Protocol Error）。

3.14.2.2.4 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是 UTF-8 字符串对。此属性可用于向客户端提供包括诊断信息在内的附加信息。如果加上用户属性之后的 DISCONNECT 报文长度超出了接收端指定的最大报文长度，则发送端**不能**发送此属性 [MQTT-3.14.2-4]。用户属性允许出现多次，以表示多个名字/值对，且相同的名字可以多次出现。

3.14.2.2.5 服务端参考

28 (0x1C)，服务端参考（Server Reference）标识符。

跟随其后的是一个 UTF-8 编码字符串，客户端可以使用它来识别其他要使用的服务端。包含多个服务端参考将造成协议错误（Protocol Error）。

服务端发送包含一个服务端参考和原因码 0x9C（（临时）使用其他服务端）或 0x9D（服务端已（永久）移动）的 DISCONNECT 报文，如 4.13 节 所述。

关于如何使用服务端参考，参考 4.11 节 服务端重定向。

图 3-24 - DISCONNECT 报文可变报头非规范示例

	说明	7	6	5	4	3	2	1	0
断开原因码									
byte 1		0	0	0	0	0	0	0	0
属性									
byte 2	长度 (5)	0	0	0	0	0	1	0	1
byte 3	会话过期间隔标识符 (17)	0	0	0	1	0	0	0	1
byte 4	会话过期间隔 (0)	0	0	0	0	0	0	0	0
byte 5		0	0	0	0	0	0	0	0
byte 6		0	0	0	0	0	0	0	0
byte 7		0	0	0	0	0	0	0	0

3.14.3 DISCONNECT 载荷

DISCONNECT 报文没有有效载荷。

3.14.4 DISCONNECT 行为

发送端发送完 DISCONNECT 报文之后：

- **不能**再在此网络连接上发送任何 MQTT 控制报文 [MQTT-3.14.4-1]。
- **必须**关闭网络连接 [MQTT-3.14.4-2]。

接收到包含原因码为 0x00（成功）的 DISCONNECT 时，服务端：

- **必须**丢弃任何与当前连接相关的遗嘱消息，而不发布它 [MQTT-3.14.4-3]，如 3.1.2.5 节 所述。

接收到 DISCONNECT 报文时，接收端：

- **应该**关闭网络连接

3.15 AUTH – 认证交换

AUTH 报文被从客户端发送给服务端，或从服务端发送给客户端，作为扩展认证交换的一部分，比如质询/响应认证。如果 CONNECT 报文不包含相同的认证方法，则客户端或服务端发送 AUTH 报文将造成协议错误（Protocol Error）。

3.15.1 AUTH 固定报头

图 3-35 – AUTH 报文固定报头

Bit	7	6	5	4	3	2	1	0

byte 1	MQTT 控制报文类型 (15)				保留位			
	1	1	1	1	0	0	0	0
byte 2	剩余长度							

AUTH 报文固定报头第 3，2，1，0 位是保留位，**必须**全设置为 0。客户端或服务端**必须**把其他值当做无效值并关闭网络连接 [MQTT-3.15.1-1]。

剩余长度字段

等于可变报头的长度，编码为变长字节整数。

3.15.2 AUTH 可变报头

AUTH 报文可变报头按顺序包含以下字段：认证原因码（Authentication Reason Code），属性（Properties）。属性的编码规则，如 2.2.2 节 所述。

3.15.2.1 认证原因码

可变报头第 0 字节是认证原因码（Authenticate Reason Code）。单字节无符号认证原因码字段的值如下所示。AUTH 报文的发送端**必须**使用一种认证原因码 [MQTT-3.15.2-1]。

表 3-11 - 认证原因码

值	16 进制	原因码名称	发送端	说明
0	0x00	成功	服务端	认证成功。
24	0x18	继续认证	服务端或客户端	继续下一步认证。
25	0x19	重新认证	客户端	开始重新认证。

如果原因码为 0x00（成功）并且没有属性字段，则可以省略原因码和属性长度。这种情况下，AUTH 报文剩余长度为 0。

3.15.2.2 AUTH 属性

3.15.2.2.1 属性长度

AUTH 报文可变报头中的属性的长度被编码为变长字节整数。

3.15.2.2.2 认证方法

21 (0x15)，认证方法（Authentication Method）标识符。

跟随其后的是一个 UTF-8 编码字符串，包含认证方法名称。省略认证方法或者包含多个认证方法都将造成协议错误（Protocol Error）。更多关于扩展认证的信息，参考 4.12 节。

3.15.2.2.3 认证数据

22 (0x16)，认证数据（Authentication Data）标识符。

跟随其后的是二进制数据，包含认证数据。包含多个认证数据将造成协议错误（Protocol Error）。此数据的内容由认证方法定义。更多关于扩展认证的信息，参考 4.12 节。

3.15.2.2.4 原因字符串

31 (0x1F)，原因字符串（Reason String）标识符。

跟随其后的是 UTF-8 编码字符串，表示断开原因。此原因字符串是为诊断而设计的可读字符串，**不应该**被接收端所解析。

如果加上原因字符串之后的 AUTH 报文长度超出了接收端所指定的最大报文长度，则发送端不能发送此属性 [MQTT-3.15.2-2]。包含多个原因字符串将造成协议错误（Protocol Error）。

3.15.2.2.5 用户属性

38 (0x26)，用户属性（User Property）标识符。

跟随其后的是 UTF-8 字符串对。此属性可用于向客户端提供包括诊断信息在内的附加信息。如果加上用户属性之后的 AUTH 报文长度超出了接收端指定的最大报文长度，则服务端**不能**发送此属性 [MQTT-3.15.2-3]。用户属性（User Property）允许出现多次，以表示多个名字/值对，且相同的名字可以多次出现。

3.15.3 AUTH 载荷

AUTH 报文没有有效载荷。

3.15.4 AUTH 行为

更多关于扩展认证的信息，参考 4.12 节。

4 操作行为

4.1 会话状态

为实现 QoS 等级 1 和 QoS 等级 2 协议流，客户端和服务端需要将状态与客户标识符相关联，这被称为会话状态。服务端还将订阅信息存储为会话状态的一部分。

会话可以跨越一系列的网络连接。它持续到最新的网络连接（Network Connections）加上会话过期间隔（Session Expiry Interval）。

客户端的会话状态包括：

- 已发送给服务端，但是还没有完成确认的 QoS 等级 1 和 QoS 等级 2 的消息。
- 从服务端收到的，但是还没有完成确认的 QoS 等级 2 消息。

服务端的会话状态包括：

- 会话是否存在，即使会话状态其余部分为空。
- 客户端订阅信息，包括任何订阅标识符。
- 已发送给客户端，但是还没有完成确认的 QoS 等级 1 和 QoS 等级 2 的消息。
- 等待传输给客户端的 QoS 等级 0（可选），QoS 等级 1 和 QoS 等级 2 的消息。
- 从客户端收到的，但是还没有完成确认的 QoS 等级 2 消息。遗嘱小子和遗嘱延时间隔。
- 如果会话当前未连接，会话结束时间和会话状态将被丢弃。

保留消息不是会话状态的一部分，会话结束时不被删除。

4.1.1 存储会话状态

当网络连接打开时，客户端和服务端**不能**丢弃会话状态 [MQTT-4.1.0-1]。当网络连接被关闭并且会话过期间隔已过时，服务端**必须**丢弃会话状态 [MQTT-4.1.0-2]。

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。已存储的会话状态可能因为管理操作（比如某个预定义条件的自动响应）而被丢弃。它造成的后果就是会话终止。这些操作可能是因为资源受限或其他操作原因引发的。硬件或软件故障可能导致客户端或服务端存储的会话状态丢失或损坏。需要谨慎的评估客户端和服务端的存储能力，以确保存储空间充足。

4.1.2 会话状态非规范示例

例如，想要收集电表读数的用户可能会决定使用 QoS 等级 1 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将所有的数据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）字节流。此规范不要求任何指定的传输协议。客户端或服务端**可以**支持这里列出的任何传输协议，或者满足 本节 要求的任何其他传输协议。

客户端或服务端**必须**支持使用一个或多个提供有序的、可靠的、双向传输（从客户端到服务端和从服务端到客户端）字节流传输的底层传输协议 [MQTT-4.2-1]。

非规范评注

MQTT v5.0 使用的传输层协议是 [RFC0793] 定义的 TCP/IP 协议。下面的协议也支持：

- TLS [RFC5246]
- WebSocket [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。

非规范评注

无连接的网络传输，如用户数据包协议 (UDP) 本身不适合，因为它们可能丢失或重新排列数据。

4.3 服务质量等级和协议流程

MQTT 按照后面章节定义的服务质量（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端也可以是接收端。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。

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。接收端不会发送响应，发送端也不会重试。消息可能送达一次也可能根本没送达。

对于 QoS 等级 0 的分发协议，发送端

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

对于 QoS 等级 0 的分发协议，接收端

- 接受 PUBLISH 报文时同时接受消息的所有权。

图 4-1 – QoS 等级 0 协议流程图，非规范示例

发送端动作	控制报文	接收端动作
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

服务质量等级 1 确保消息至少送达一次。QoS 等级 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。 2.2.1 节 提供了有关报文标识符的更多信息。

对于 QoS 等级 1 的分发协议，发送端

- 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 [MQTT-4.3.2-1]。
- 发送的 PUBLISH 报文**必须**包含报文标识符且 QoS 等于 1，DUP 等于 0 [MQTT-4.3.2-2]。
- **必须**将这个 PUBLISH 报文看作是*未确认的*，直到从接收端那收到对应的 PUBACK 报文。 4.4 节 有一个关于未确认消息的讨论 [MQTT-4.3.2-3]。

一旦发送端收到 PUBACK 报文，这个报文标识符就可以重用。

注意：允许发送端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 等级 1 的分发协议，接收端

- 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 [MQTT-4.3.2-4]。
- 发送了 PUBACK 报文之后，接收端**必须**将任何包含相同报文标识符的入站 PUBLISH 报文当做一个新的消息，并忽略它的 DUP 标志的值 [MQTT-4.3.2-5]。

图 4-2 – QoS 等级 1 协议流程图，非规范示例

发送端动作	控制报文	接收端动作
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] 不要求接收端在发送 PUBACK 之前完整分发应用消息。原来的发送端收到 PUBACK 报文之后，应用消息的所有权就会转移给这个接收端。

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 等 2 消息可变报头中有报文标识符。2.2.1 节 提供了有关报文标识符的更多信息。QoS 等级 2 的 PUBLISH 报文的接收端使用一个两部确认过程来确认收到。

对于 QoS 等级 2 的分发协议，发送端

- **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 [MQTT-4.3.3-1]。
- 发送端 PUBLISH 报文**必须**包含报文标识符且报文的 QoS 等于 2，DUP 等于 0 [MQTT-4.3.3-2]。
- **必须**将这个 PUBLISH 报文看作是*未确认的*，直到从接收端那收到对应的 PUBREC 报文 [MQTT-4.3.3-3]。4.4 节 有一个关于未确认消息的讨论。
- 收到发送端发送的包含原因码小于 0x80 的 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 [MQTT-4.3.3-4]。
- **必须**将这个 PUBREL 报文看作是*未确认的*，直到从接收端那收到对应的 PUBCOMP 报文 [MQTT-4.3.3-5]。
- 一旦发送了对应的 PUBREL 报文就**不能**重发这个 PUBLISH 报文 [MQTT-4.3.3-6]。
- 如果 PUBLISH 报文已发送，**不能**应用消息过期属性 [MQTT-4.3.3-7]。

一旦发送端收到包含原因码大于 0x80 的 PUBCOMP 报文，这个报文标识符就可以重用。

注意：允许发送端在等待确认时使用不同的报文标识符发送后续的 PUBLISH 报文，受制于 4.9 节 描述的流量控制。

对于 QoS 等级 2 的分发协议，接收端

- 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 [MQTT-4.3.3-8]。
- 如果接收端发送了包含原因码大于等于 0x80 的 PUBREC 报文，它**必须**将后续包含相同报文标识符的 PUBLISH 报文当做是新的应用消息 [MQTT-4.3.3-9]。
- 在收到对应的 PUBREL 报文之前，接收端**必须**发送 PUBREC 报文确认任何后续的具有相同报文标识符的 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 [MQTT-4.3.3-10]。
- **必须**发送包含与 PUBREL 相同报文标识符的 PUBCOMP 报文作为对 PUBREL 报文的响应 [MQTT-4.3.3-11]。
- 发送 PUBCOMP 报文之后，接收端**必须**将后续包含相同报文标识符的 PUBLISH 报文当做是新的应用消息 [MQTT-4.3.3-12]。
- **必须**继续 QoS 等级 2 确认序列，即使它已经应用了消息过期属性 [MQTT-4.3.3-13]。

4.4 消息分发重试

客户端以新开始（Clean Start）标志为 0 且会话存在的情况下重连时，客户端和服务端都**必须**使用原始报文标识符重新发送任何未被确认的 PUBLISH 报文（当 QoS > 0）和 PUBREL 报文。这是唯一**要求**客户端或服务端重发消息的情况。客户端和服务端**不能**在其他任何时间重发消息 [MQTT-4.4.0-1]。

如果收到包含原因码大于等于 0x80 的 PUBACK 或 PUBREC，则对应的 PUBLISH 报文被看作已确认，且**不能**被重传 [MQTT-4.4.0-2]。

图 4-3 – QoS 等级 2 协议流程图，非规范示例

发送端动作	控制报文	接收端行为
存储消息		
发送 PUBLISH 报文 QoS=2，DUP=0，带报文标识符		
	---------->	
		存储报文标识符，然后启动应用消息的向前分发 [1]
		发送 PUBREC 报文，带报文标识符和原因码
	<----------	
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		
	---------->	
		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] 不要求接收端在发送 PUBREC 和 PUBCOMP 之前完整分发应用消息。原始发送端收到 PUBREC 报文之后，应用消息的所有权就会转移给这个接收端。然而，接收端需要在接受所有权之前执行对所有可能导致转发失败（例如超出配额、权限等）的条件的检查。接收端在 PUBREC 中使用适当的原因码指示所有权接受成功或失败。

4.5 消息收到

当服务端接受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 [MQTT-4.5.0-1]。匹配规则定义见 4.7 节 。

正常情况下，客户端收到的消息是对他们创建的订阅的响应。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。 UNSUBSCRIBE 操作正在被处理时也可能收到消息。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，不管它是否选择处理其包含的应用消息 [MQTT-4.5.0-2]。

4.6 消息排序

实现 4.3 节 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 等级 1 和 QoS 等级 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 等级 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 等级 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 等级 2 消息） [MQTT-4.6.0-4]。

一个有序主题（Ordered Topic）是一个主题，在这个主题中，客户端可以确定从同一个客户端接收的相同 QoS 等级的消息的顺序与他们发布的顺序一致。当服务端处理发布到有序主题的消息时，它**必须**按照消息从任何给定客户端接收的顺序发送 PUBLISH 报文给消费端（对于同一主题和 QoS 等级） [MQTT-4.6.0-5]。这是上面列出的规则的补充。

默认情况下，服务端转发非共享订阅的消息时，**必须**将每个主题都视为有序主题 [MQTT-4.6.0-6]。服务端**可以**提供管理或其他机制来允许一个或多个主题不被当作有序主题。

> **非规范评注**
>
> 上面列出的规则确保，使用 QoS 等级 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 等级 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。关于如何使用 Receive Maximum 的详细信息，参考 4.9 节 流控。

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，允许客户端一次订阅多个主题。

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-4.7.0-1]。

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

数字符号（'#' 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**单独指定，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-4.7.1-1]。

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"主题名，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。在使用它时，它**必须**占据过滤器的整个层级 [MQTT-4.7.1-2]。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/"。

- "+"是有效的。
- "+/tennis/#"是有效的。

- “sport+”是无效的。
- “sport/+/player1”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$字符开头的主题名匹配通配符（#或+）开头的主题过滤器 [MQTT-4.7.2-1]。服务端**应该**阻止客户端使用这种主题名与其他客户端交换消息。服务端实现**可以**将$开头的主题名用作其他目的。

非规范评注

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

下列规则应用于主题名和主题过滤器：

- 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大小写敏感的。
- 主题名和主题过滤器可以包含空格字符。
- 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 65,535 字节 [MQTT-4.7.3-3]。见 1.5.4 节 。

除了不能超过 UTF-8 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”是不同的主题名。

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。主题资源**可以**是管理员在服务端预先定义好的，也**可以**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 订阅

MQTT 提供两种订阅方式，共享和非共享。

非规范评注

在早期的 MQTT 版本中，所有的订阅都是非共享的。

4.8.1 非共享订阅

非共享订阅只与创建它的会话相关联。每个订阅（Subscription）包含一个指示用于在此会话上分发消息的主题过滤器和订阅选项。服务端负责收集与过滤器相匹配的消息，并在此会话的连接上发送这些消息。

一个会话不能有多个包含相同主题过滤器的非共享订阅，因此主题过滤器可以用作标识此会话的订阅的关键词。

如果有多个客户端，每个客户端都拥有对某个相同主题的非共享订阅，则每个客户端都将获得在该主题上发布的应用消息的副本。这意味着非共享订阅不能被用于多个消费客户端的应用消息负载均衡，因为在这种情况下，每条消息都将被传递给每一个订阅的客户端。

4.8.2 共享订阅

共享订阅可以与多个订阅会话相关联。与非共享订阅一样，它包含一个主题过滤器和订阅选项。但是，与此主题过滤器相匹配的发布消息仅被发布到其中一个订阅会话。共享订阅在多个消费客户端并行共享处理发布消息时是很有用的。

使用特殊样式的主题过滤器来表示共享订阅。过滤器格式如下：

$share/{ShareName}/{filter}

- $share 是字符串字面量，用来把主题过滤器标记为共享订阅主题过滤器。
- {ShareName}是字符串，不包含"/"，"+"或"#"。
- {filter}该字符串的剩余部分与非共享订阅中的主题过滤器具有相同的语法和语义。参考 4.7 节。

共享订阅主题过滤器**必须**以$share/开始，且**必须**包含至少一个字符长度的共享名（ShareName） [MQTT-4.8.2-1]。共享名不能包含字符"/"，"+"或"#"，但**必须**跟在"/"字符后面。此"/"字符后面**必须**跟随一个主题过滤器 [MQTT-4.8.2-2] ，如 4.7 节 所述。

非规范评注

共享订阅在 MQTT 服务端的范围内定义，而不是在会话中定义。共享订阅的主题过滤器包含共享名，因此服务端可以有多个包含相同{过滤器}组件的共享订阅。通常，应用程序使用共享名表示共享同一个订阅的一组订阅会话。

示例：

- 共享订阅"$share/consumer1/sport/tennis/+"和"$share/consumer2/sport/tennis/+"是不同的共享订阅，因此可以被关联到不同的会话组。它们都与非共享订阅主题"sport/tennis/+"相匹配。

  如果一条消息被发布到匹配主题"sport/tennis/+"，则消息的副本仅发送给所有订阅"$share/consumer1/sport/tennis/+"的会话中的一个会话，也仅发送给所有订阅"$share/consumer2/sport/tennis/+"的会话中的一个会话。更多的副本将发送给所有对"sport/tennis/+"进行非共享订阅的客户端。

- 共享订阅"$share/consumer1//finance"匹配非共享订阅主题"/finance"。

  注意，"$share/consumer1//finance"和"$share/consumer1/sport/tennis/+"是不同的共享订阅，尽管它们有相同的共享名。它们可能在某种程度上是相关的，但拥有相同的共享名并不意味着它们之间有某种关系。

通过 SUBSCRIBE 请求中的共享订阅主题过滤器创建共享订阅。只有一个会话订阅了某个共享订阅时，共享订阅行为如同非共享订阅，除了：

- 匹配发布消息时，不考虑"$share"和{共享名}部分。
- 第一次订阅时，保留消息不发送给此会话。其他匹配的发布消息将发送给此会话。

一旦某个共享订阅存在，其他会话就有可能订阅了相同的共享订阅主题过滤器。新的会话作为额外的订阅者关联到此共享订阅。保留消息不发送给此新的订阅者。后续每条与此共享订阅相匹配的应用消息被发送到该共享订阅关联的其中一个会话。

会话可以通过发送包含某共享订阅主题过滤器的 UNSUBSCRIBE 报文来显式的将其从共享订阅中分离。会话终止时，也将从共享订阅中分离。

共享订阅持续到至少有一个与其相关的会话（即，会话已经对此共享订阅主题过滤器发布了成功的 SUBSCRIBE 请求，且尚未完成相应的 UNSUBSCRIBE）。当初始创建此共享订阅的会话取消订阅时，除非没有其他的相关会话，否则共享订阅仍然存在。共享订阅在没有被任何会话订阅时结束，且任何相关的未分发的消息都被删除。

共享订阅注释

- 如果有不止一个会话订阅了某个共享订阅，服务端在消息的基础上自由的选择使用哪个会话，以及使用什么标准来进行该选择。

- 允许不同的订阅客户端在其 SUBSCRIBE 报文中请求不同的 QoS 等级。服务端决定授予每个客户端的最大 QoS 等级，并且允许向不同的订阅者授予不同的最大 QoS 等级。向客户端发送应用消息时，服务端**必须**考虑授予客户端的 QoS 等级 [MQTT-4.8.2-3]，与向订阅者发送消息相同。

- 如果服务端正在向其选中的订阅客户端发送 QoS 等级 2 的消息，并且在分发完成之前网络中断，服务端**必须**在客户端重新连接时完成向该客户端的消息分发 [MQTT-4.8.2-4]，如 4.3.3 节 所述。如果客户端的会话在客户端重连之前终止，服务端**不能**把此消息发送给其他订阅的客户端 [MQTT-4.8.2-5]。

- 如果服务端正在向其选中的订阅客户端发送 QoS 等级 1 的消息，并且服务端在收到此客户端的确认报文之前网络中断，服务端**可以**等客户端重新连接之后将消息重传给客户端。如果客户端的会话在客户端重连之前终止，服务端**应该**把此应用消息发送给与此共享订阅相关的另一个客户端。服务端**可以**在第一个客户端断开连接时就尝试将消息发送给另一个客户端。

- 如果客户端对来自服务端的 PUBLISH 报文使用包含原因码大于等于 0x80 的 PUBACK 或 PUBREC 报文进行响应，服务端**必须**丢弃应用消息而不尝试将其发送给任何其他订阅者 [MQTT-4.8.2-6]。

- 允许客户端向已订阅的共享订阅第二次发送 SUBSCRIBE 请求。比如，它可以通过这样改变其订阅请求的 QoS 等级，或者因为它不确定以前的连接关闭之前订阅是否已完成。这不会增加共享订阅关联的会话个数，因此会话将在其第一次发送 UNSUBSCRIBE 之后脱离此共享订阅。

- 每个共享订阅都是独立于其他共享订阅的。有可能两个共享订阅包含了重叠的过滤器。在这种情况下，与两个共享订阅都相匹配的消息都将被它们单独处理。如果某个客户端既有共享订阅也有非共享订阅，且某个消息与它们都相匹配，客户端将由于存在非共享订阅而接收此消息的副本，此消息的第二个副本将分发给此共享订阅的某个订阅者，因此可能导致两份副本都被发送给此客户端。

4.9 流控

客户端和服务端使用接收最大值来控制接收未被确认的 PUBLISH 报文数量，如 3.1.2.11.4 节 和 3.2.2.3.2 节 所述。接收最大值创建了一个发送配额，用于限制可以在没收到 PUBACK（QoS 等级 1）或 PUBCOMP（QoS 等级 2）的情况下发送的 QoS 等级大于 0 的 PUBLISH 报文数量。PUBACK 和 PUBCOMP 按照下述方式补充配额。

客户端或服务端**必须**将其初始发送配额设置为不超过接收最大值的非 0 值 [MQTT-4.9.0-1]。

每当客户端或服务端发送了一个 QoS 等级大于 0 的 PUBLISH 报文，它就会减少发送配额。如果发送配额减为 0，客户端或服务端**不能**再发送任何 QoS 等级大于 0 的 PUBLISH 报文 [MQTT-4.9.0-2]。它**可以**继续发送 QoS 为 0 的 PUBLISH 报文，也**可以**选择暂停发送这些报文。即使配额为 0，客户端和服务端也**必须**继续处理和响应其他 MQTT 控制报文 [MQTT-4.9.0-3]。

发送配额增加 1：

- 每当收到一个 PUBACK 报文或 PUBCOMP 报文，不管 PUBACK 或 PUBCOMP 报文是否包含错误码。
- 每次收到一个包含返回码大于等于 0x80 的 PUBREC 报文。

如果发送配额已到达初始发送配额，则不继续增加。在初始发送配额之上尝试增加配额可能是由建立新的网络连接后重新发送 PUBREL 数据包引起的。

关于客户端和服务端在超出最大接收值的允许的情况下发送 PUBLISH 报文的描述，参考 3.3.4 节。

发送配额和接收最大值的保留不跨越网络连接，每次建立新的网络连接时按照上面的描述进行初始化。它们不是会话状态的一部分。

4.10 请求/响应

有些应用程序或标准可能希望通过 MQTT 协议运行请求/响应交互。此版本 MQTT 协议包含三个可用于此目的的属性：

- 响应主题，在 3.3.2.3.5 节 中描述
- 对比数据，在 3.3.2.3.6 节 中描述
- 请求响应信息，在 3.1.2.11.7 节 中描述
- 响应信息，在 3.2.2.3.14 节 中描述

以下非规范部分描述了如何使用这些属性。

客户端通过发布一个包含响应主题的应用消息来发送请求消息，如 3.3.2.3.5 节 所述。请求消息可以包含对比数据属性，如 3.3.2.3.6 节 所述。

4.10.1 基本请求响应（非规范）

请求/响应交互过程如下：

1. MQTT 客户端（请求方）向主题发布请求消息。请求消息是具有响应主题的应用消息。
2. 另一个 MQTT 客户端（响应方）订阅了与请求消息发布时使用的主题名相匹配的主题过滤器。结果，它收到请求消息。可能有多个响应方订阅了此主题名，也可能没有响应方。
3. 响应方根据请求消息采取适当的操作，然后往请求消息中携带的响应主题属性中的主题名发布响应消息。
4. 典型用法，请求放订阅了响应主题，从而接收到响应信息。但是，其他某些客户端可能会订阅响应主题，因此它们也将接收和处理响应消息。与请求消息一样，可能有多个客户端订阅了响应消息的发送主题，也可能没有。

如果请求消息包含对比数据属性，则响应方将此属性拷贝到响应消息中，由响应消息的接收端用来将响应消息与原始请求相关联。响应消息不包含响应主题属性。

MQTT 服务端转发请求消息中的响应主题和对比数据属性，和响应消息中的对比数据属性。服务端像处理其他应用程序消息一样处理请求消息和响应消息。

请求放通常在发布请求消息之前订阅响应主题。如果响应消息发送时没有任何订阅者订阅了响应主题，则响应消息将不会传递给任何客户端。

请求消息和响应消息可以具有任何 QoS 等级，并且响应方可以使用具有非 0 会话过期间隔的会话。通常使用 QoS 等级 0 发送请求消息，并且只有在应答者正连接时才发送请求消息。但这不是必须的。

响应者可以使用共享订阅来允许响应客户端池。注意，使用共享订阅时，不保证消息在客户端之间的分发顺序。

请求方有责任确保它具有发布消息到请求消息的主题、并订阅响应主题属性中主题名的必要权限。响应方有责任确保它具有订阅请求主题和发布到响应主题的权限。虽然主题授权不属于本规范，但建议服务端实施此类授权。

4.10.2 确定响应主题值（非规范）

请求方可以通过包括本地配置在内的任何方式来确定作为他们的响应主题的主题名。为避免不同请求方之间的冲突，由请求方客户端使用的响应主题最好对于该客户端是唯一的。由于请求方和响应方通常都需要对这些主题进行授权，因此使用随机主题名称将会对授权造成挑战。

为了解决此问题，本规范在 CONNACK 报文中定义了一个名为响应信息的属性。服务端可以使用此属性指导客户端如何选择使用的响应主题。此机制对于服务端和客户端都是可选的。连接时，客户端通过设置 CONNECT 报文中的请求响应信息属性来请求服务端发送响应信息。这会导致服务端在 CONNACK 报文中插入响应信息属性（UTF-8 编码的字符串）。

本规范不定义响应信息的内容，但它可以被用来传递主题树的全局唯一部分，该部分至少在其会话的整个生命周期内保留给该客户端。使用这种机制，可以在服务端而不是每个客户端中完成该属性的配置。

有关响应信息的定义，参考 3.1.2.11.7 节 。

4.11 服务端重定向

服务端可以通过发送包含原因码为 0x9C（（临时）使用其他服务端）或 0x9D（服务端已（永久）移动）的 CONNACK 或 DISCONNECT 报文请求客户端使用另一台服务端，如 4.13 节 所述。服务端发送这些原因码时**可以**包含一个服务端参考属性，用以说明客户端**应该**使用的服务端位置。

原因码 0x9C （（临时）使用其他服务端） 指定客户端**应该**临时切换到另一台服务端。另一台服务端可能是客户端已知的，也可能是由服务端参考所指定的。

原因码 0x9D （服务端已（永久）移动）指定客户端**应该**永久切换到另一台服务端。另一台服务端可能是客户端已知的，也可能是由服务端参考所指定的。

服务端参考是一个 UTF-8 编码字符串，其值是一个由空格分隔开的参考列表。本规范不指定服务端参考的格式。

非规范评注

推荐每个参考包含名称及可选的端口号。如果名称包含冒号，则名称字符串可以由方括号括起来（“[“和“]”）。由方括号括起来的名称不能包含右方括号(“]”)字符，用于表示使用冒号分隔符的 IPv6 地址。这是一个简化版的 URI 授权，如 [RFC3986] 所述。

非规范评注

服务端参考中的名字通常代表主机名、DNS 名 [RFC1035]、SRV 名 [RFC2782] 或 IP 地址。跟随冒号分隔符的通常是十进制端口号。如果端口信息来自于 DNS（比如包含 SRV）或者使用默认端口，则主机名后无需跟随端口号。

非规范评注

如果给出了多个服务端参考，则期望客户端选择其中一个。

非规范评注

服务端参考示例如下：

```
myserver.xyz.org
myserver.xyz.org:8883
10.10.151.22:8883 [fe80::9610:3eff:fe1c]:1883
```

允许服务端不发送服务端参考，允许客户端忽略服务端参考。此特性可用于负载均衡、服务端重定位和服务端预置服务端。

4.12 增强认证

MQTT CONNECT 报文使用用户名和密码字段支持基本的网络连接认证。这些字段虽然称为简单密码认证，但可以被用来承载其他形式的认证，例如把密码作为令牌（Token）传递。

增强认证包含质询/响应风格的认证，从而扩展了基本认证。它可能涉及在 CONNECT 报文之后、CONNACK 报文之前的客户端和服务端之间 AUTH 报文交换。

服务端通过在 CONNECT 报文中添加认证方法字段来启动增强认证。此字段指定使用的认证方法。如果服务端不支持客户端提供的认证方法，它**可以**发送一个包含原因码 0x8C（无效的认证方法）或 0x87（未授权）的 CONNACK 报文，如 4.13 节 所述，并且**必须**关闭网络连接 [MQTT-4.12.0-1]。

认证方法是客户端和服务端关于认证数据中的数据和 CONNECT 报文中其他字段的含义，以及客户端和服务端完成认证需要交换和处理的协议。

非规范评注

认证方法通常为 SASL（Simple Authentication and Security Layer）机制，使用一个注册过的名称便于信息交换。然而，认证方法不限于使用已注册的 SASL 机制。

如果客户端选择的认证方法指定客户端先发送数据，客户端**应该**在 CONNECT 报文中包含认证数据属性。此属性可被用来提供认证方法指定的数据，认证数据的内容由认证方法定义。

如果服务端需要额外的信息来完成认证，它可以向客户端发送 AUTH 报文，此报文**必须**包含原因码 0x18（继续认证） [MQTT-4.12.0-2]。如果认证方法需要服务端向客户端发送认证相关的数据，这些数据在认证数据（Authentication Data）中发送。

客户端通过发送另一个 AUTH 报文响应来自服务端的 AUTH 报文，此报文**必须**包含原因码 0x18（继续认证） [MQTT-4.12.0-3]。如果认证方法要求客户端向服务端发送认证相关的数据，这些数据在认证数据（Authentication Data）中发送。

客户端和服务端按需交换 AUTH 报文，直到服务端通过发送包含原因码为 0 的 CONNACK 报文接受认证为止。如果接受认证需要向客户端发送数据，这些数据在认证数据中发送。

客户端可以在处理过程中随时关闭连接。它**可以**在关闭之前发送 DISCONNECT 报文。服务端可以在处理过程中随时拒绝认证。它**可以**发送包含原因码大于等于 0x80 的 CONNACK 报文 ，如 4.13 节 所述，并且**必须**关闭网络连接 [MQTT-4.12.0-4]。

如果初始 CONNECT 报文包含认证方法属性，则所有的 AUTH 报文和成功的 CONNACK 报文**必须**包含与 CONNECT 报文中相同的认证方法属性。 [MQTT-4.12.0-5]。

增强认证的实现对于客户端和服务端来说都是可选的。如果客户端在 CONNECT 报文中没有包含认证方法，则服务端不能发送 AUTH 报文，且**不能**在 CONNACK 报文中发送认证方法 [MQTT-4.12.0-6]。如果客户端在 CONNECT 报文中没有包含认证方法，则客户端**不能**向服务端发送 AUTH 报文 [MQTT-4.12.0-7]。

如果客户端在 CONNECT 报文中没有包含认证方法，服务端**应该**使用 CONNECT 报文中的信息、TLS 会话和网络连接进行认证。

SCRAM 认证非规范示例

- 客户端到服务端：CONNECT 认证方法="SCRAM-SHA-1"，认证数据=client-first-data
- 服务端到客户端：AUTH 原因码=0x18，认证方法="SCRAM-SHA-1"，认证数据=server-first-data
- 客户端到服务端：AUTH 原因码=0x18，认证方法="SCRAM-SHA-1"，认证数据=client-final-data
- 服务端到客户端：CONNACK 原因码=0，认证方法="SCRAM-SHA-1"，认证数据=server-final-data

Kerberos 认证非规范示例

- 客户端到服务端：CONNECT 认证方法="GS2-KRB5"
- 服务端到客户端：AUTH 原因码=0x18，认证方法="GS2-KRB5"
- 客户端到服务端：AUTH 原因码=0x18，认证方法="GS2-KRB5"，认证数据=initial context token
- 服务端到客户端：AUTH 原因码=0x18，认证方法="GS2-KRB5"，认证数据=reply context token
- 客户端到服务端：AUTH 原因码=0x18，认证方法="GS2-KRB5"
- 服务端到客户端：CONNACK 原因码=0，认证方法="GS2-KRB5"，认证数据=outcome of authentication

4.12.1 重新认证

如果客户端在 CONNECT 报文中提供了认证方法，它可以在收到 CONNACK 报文之后的任何时间通过发送包含原因码 0x19（重新认证）的 AUTH 报文发起重新认证。客户端**必须**将认证方法设置为与最初验证网

络连接时的认证方法一致 [MQTT-4.12.1-1]。如果认证方法需要客户端先发送数据，则此 AUTH 报文包含第一片认证数据。

服务端通过向客户端发送 AUTH 报文来响应此重新认证请求，包含原因码为 0x00（成功）的 AUTH 报文指示重新认证完成，包含原因码为 0x18（继续认证）的 AUTH 报文指示需要更多的认证数据。客户端可以通过发送包含原因码 0x18（继续认证）的 AUTH 报文来响应附加的认证数据。此流程与原始身份验证一样，直到重新认证完成或重新认证失败。

如果重新认证失败，客户端或服务端**应该**发送包含适当原因码的 DISCONNECT 报文，如 4.13 节 所述。并且**必须**关闭网络连接 [MQTT-4.12.1-2]。

在重新认证的过程中，客户端和服务端的其他报文流可以继续使用之前的认证。

非规范评注

服务端可以通过拒绝重新认证来限制客户端在重新认证中尝试的更改范围。例如，如果服务端不允许更改用户名，它可以使任何尝试更改用户名的重新认证都失败。

4.13 错误处理

4.13.1 无效报文和协议错误

无效报文（Malformed Packet）和协议错误（Protocol Error）的定义见 1.2 节 术语。这些错误案例的部分术语贯穿本规范。客户端或服务端对其收到的 MQTT 控制报文的检查严格程度依赖：

- 客户端或服务端实现的大小。
- 实现支持的性能。
- 接收端对发送端发送的 MQTT 控制报文的信任程度。
- 接收端对用于分发 MQTT 控制报文的网络的信任程度。
- 继续处理错误报文的的后果。

如果发送端遵守此规范，它将不会发送无效报文或导致协议错误。然而，如果客户端在收到 CONNACK 报文之前发送 MQTT 控制报文，它可能会因为错误的估计了服务端的性能而导致协议错误。参考 3.1.4 节 CONNECT 行为。

无效报文和协议错误使用的原因码包括：

- 0x81 无效报文
- 0x82 协议错误
- 0x93 超过接收最大值
- 0x95 报文过大
- 0x9A 不支持保留
- 0x9B 不支持的 QoS 等级
- 0x9E 不支持共享订阅
- 0xA1 不支持订阅标识符
- 0xA2 不支持通配符订阅

当客户端检测到无效报文或协议错误，并且本规范中给出了相应的原因码时，它**应该**关闭网络连接。在 AUTH 报文出错的情况下它**可以**在关闭网络连接之前发送包含原因码的 DISCONNECT 报文。在其他报文出错的情况下它**应该**在关闭网络连接之前发送包含原因码的 DISCONNECT 报文。使用原因码 0x81（错误报文）或 0x82（协议错误），除非包含 3.14.2.1 断开原因码 中定义的更具体的原因码。

当服务端检测到无效报文或协议错误，并且本规范中给出了相应的原因码时，它**必须**关闭网络连接 [MQTT-4.13.1-1]。在 CONNECT 报文出错的情况下它**可以**在关闭网络连接之前发送包含原因码的 CONNACK 报文。在其他报文出错的情况下它**应该**在关闭网络连接之前发送包含原因码的 DISCONNECT 报文。使用原因码 0x81（无效报文）或 0x82（协议错误），除非包含 3.2.2.2 节 - 连接原因码 或 3.14.2.1 节 – 断开原因码 中定义的更具体的原因码。对其他会话没有影响。

如果服务端或客户端省略了检查 MQTT 控制报文的某些特性，它可能无法检测到某个错误，因此可能会导致数据被损坏。

4.13.2 其他错误

发送端无法预料到无效报文和协议错误以外的错误，因为它可能有某些没有告知发送端的约束。客户端或服务端可能在接收时遇到短暂的错误，比如内存不足，导致无法成功的处理某个 MQTT 控制报文。

包含原因码大于等于 0x80 的确认报文 PUBACK，PUBREC，PUBREL，PUBCOMP，SUBACK，UNSUBACK 表明收到了某个报文标识符的报文出错。这不会影响其他会话或此会话上的其他报文。

CONNACK 报文和 DISCONNECT 报文允许使用大于等于 0x80 的原因码以指示网络连接将被关闭。如果某个大于等于 0x80 的原因码被指定，无论是否发送 CONNACK 报文或 DISCONNECT 报文，**必须**关闭网络连接 [MQTT-4.13.2-1]。发送这些原因码不会影响任何其他会话。

如果控制报文包含多个错误，接收端可以按照任意顺序对报文进行验证，并对发现的任何错误采取适当的行为。

5 安全（非规范）

5.1 概述

强烈建议提供 TLS [RFC5246] 的服务端实现使用 TCP 端口 8883（IANA 服务名：secure-mqtt）。

安全是一个快速变化的领域，所以在设计安全解决方案时总是使用最新的建议。

解决方案需要考虑的风险包括：

- 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（DoS）攻击
- 通信可能会被拦截、修改、重定向或泄露
- 虚假 MQTT 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，还有地区因素（例如美国欧盟隐私盾框架 [USEUPRIVSH]），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（例如萨斯班-奥克斯利法案[SARBANES]）。

5.2 MQTT 解决方案：安全和认证

协议实现可能需要提供符合特定行业安全标准，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 [MQTTNIST]）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

广泛采用的加密算法是高级加密标准 [AES] 。对 AES 提供了硬件支持的处理器有很多，但通常不包含嵌入式处理器。加密算法 ChaCha20 [CHACHA20] 软件加解密速度快很多，但不像 AES 那样广泛可用。

推荐使用为资源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

实现或使用 MQTT 时需要考虑许多安全问题。以下章节不应被视为*核对清单* 。

协议实现时可以实现下面的一部分或全部：

5.4.1 客户端身份认证

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 Auth [RFC6749] ，还可以利用操作系统的认证机制。

MQTT v5.0 提供了一种增强认证机制，如 4.12 节 所述。使用此机制需要客户端和服务端双方的支持。

实现可以明文传递认证数据，混淆数据元素，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 TLS 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用于身份验证的凭证。

5.4.2 客户端授权

如果客户端已经成功通过身份认证，服务端实现需要在接受连接之前执行授权检查。

授权可以基于客户端提供的信息如用户名，客户端主机名/IP 地址，或认证机制的结果。

具体来说，实现应该检查客户端是否被授权使用此客户标识符，因为客户标识符提供了对 MQTT 会话状态的访问（如 4.1 节 所述）。此授权检查是为了防止某个客户端偶然或恶意的使用了已被其他客户端所使用的客户标识符。

实现应该提供发生在 CONNECT 之后的访问控制以限制客户端发布消息到特定主体或使用特定主体过滤器进行订阅的能力。实现需要考虑对具有广泛作用域的主题过滤器的访问限制，如"#"主题过滤器。

5.4.3 服务端身份认证

MQTT 协议不是双向信任的。基本认证没有提供客户端验证服务端身份的机制。某些形式的扩展认证允许双向认证。

但是使用 TLS [RFC5246] 时，客户端可以使用服务端发送的 TLS 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] 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。MQTT v5.0 提供了一种增强的认证机制，如 4.12 节 所述，它可以被客户端用于验证服务端。使用此机制需要客户端和服务端双方的支持。

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

5.4.4 应用消息和 MQTT 控制报文的完整性

应用可以在应用消息中单独包含哈希值。这样做可以为 PUBLISH 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 应用消息和 MQTT 控制报文的保密性

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 客户端和服务端盗用检测

使用 TLS [RFC5246] 的客户端和服务端实现应该能够确保，初始化 TLS 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]）和在线整数状态协议（OSCP） [RFC6960] 的功能，拒绝使用被吊销的整数。

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 [IEEE8021AR] 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 异常行为检测

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，服务端实现可以关闭客户端的网络连接。

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 其它安全注意事项

如果客户端或服务端的 TLS 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）。
- 服务端可以关闭客户端的网络连接，并要求他们使用新的凭证重新验证身份。
- 服务端可以要求客户端使用 4.12.1 节 中描述的机制周期性的进行重新认证。

资源受限设备或使用受限网络的客户端可以使用 TLS [RFC5246] 会话恢复，以降低 TLS [RFC5246] 会话重连的成本。

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 SOCK 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。

使用内置的用户名称和密码字段，TLS [RFC5246] 客户端身份认证可被用于（或者替代）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 协议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，要满足下面的条件：

- 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 修改“WebSocket 子协议名”条目下 MQTT 子协议注册信息为下列数据：

图 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/v5.0/os/mqtt-v5.0-os.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 实现可以同时作为 MQTT 客户端和 MQTT 服务端。

7.1 一致性条款

7.1.1 MQTT 服务端一致性条款

服务端的定义，参考术语章节的服务端（Server）部分。

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 MQTT 控制报文的格式符合第二章和第三章描述的格式。
2. 遵守 4.7 节 描述的主题匹配规则和 4.8 节 匹配的订阅规则。
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 - 使用 WebSocket 作为网络层
4. 为了能够与任何其他一致的（MQTT）实现进行互操作，无需使用在规范之外定义的任何扩展。

7.1.2 MQTT 客户端一致性条款

客户端的定义，参考术语章节的客户端（Client） 部分。

MQTT 客户端只有满足下面所有的要求才算是符合本规范：

1. 客户端发送端所有 MQTT 控制报文的格式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 - 使用 WebSocket 作为网络层
3. 为了能够与任何其他一致的（MQTT）实现进行互操作，无需使用在规范之外定义的任何扩展。

Appendix A. 致谢

技术委员会特别感谢 Andy Stanford-Clark 博士和 Arlen Nipper 博士作为 MQTT 协议的原始发明者以及他们对标准化过程的持续支持。

以下成员在本规范制定期间为 OASIS 技术委员会成员，他们的贡献值得感谢：

参与者：

- Senthil Nathan Balasubramaniam (Infiswift)
- Dr. Andrew Banks, editor (IBM)
- Ken Borgendale, editor (IBM)
- Ed Briggs, editor (Microsoft)
- Raphael Cohn (Individual)
- Richard Coppen, chairman (IBM)
- William Cox (Individual)
- Ian Craggs , secretary (IBM)
- Konstantin Dotchkoff (Microsoft)
- Derek Fu (IBM)
- Rahul Gupta, editor (IBM)
- Stefan Hagen (Individual)
- David Horton (Solace Systems)
- Alex Kritikos (Software AG, Inc.)
- Jonathan Levell (IBM)
- Shawn McAllister (Solace Systems)
- William McLane (TIBCO Software Inc.)
- Peter Niblett (IBM)
- Dominik Obermaier (dc-square GmbH)
- Nicholas O'Leary (IBM)
- Brian Raymor, chairman (Microsoft)
- Andrew Schofield (IBM)
- Tobias Sommer (Cumulocity)
- Joe Speed (IBM)
- Dr Andy Stanford-Clark (IBM)
- Allan Stockdill-Mander (IBM)
- Stehan Vaillant (Cumulocity)

有关对早期版本 MQTT 协议做出贡献的人员列表，参考 MQTT v3.1.1 规范中的附录 A **[MQTTV311]**。

Appendix B. 强制性规范声明（非规范）

此附录是非规范性的，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。参考 第七章 一致性要求限制列表。

规范声明序号	规范声明
[MQTT-1.5.4-1]	UTF-8 编码字符串中的数据**必须**是按照 [Unicode] 规范定义的，在 RFC 3629 [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。
[MQTT-1.5.4-2]	UTF-8 编码的字符串**不能**包含空字符 U+0000。
[MQTT-1.5.4-3]	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-1.5.5-1]	编码值**必须**使用表示该值所需的最少字节数。
[MQTT-1.5.7-1]	所有的字符串都**必须**符合 UTF-8 编码字符串的要求。
[MQTT-2.1.3-1]	如果标记位被标记为“保留”，则保留它以供将来使用，并且**必须**设置为所列出的值。
[MQTT-2.2.1-2]	QoS 等级为 0 的 PUBLISH 报文**不能**包含报文标识符。
[MQTT-2.2.1-3]	客户端每次发送新的 SUBSCRIBE，UNSUBSCRIBE 或 PUBLISH（当 QoS 等级>0）MQTT 控制报文时，它**必须**为其分配一个当前未被使用的非 0 报文标识符。
[MQTT-2.2.1-4]	服务端每次发送新的 PUBLISH（当 QoS 等级>0）MQTT 控制报文时，它**必须**为其分配一个当前未被使用的非 0 报文标识符。
[MQTT-2.2.1-5]	PUBACK，PUBREC，PUBREL 或 PUBCOMP 报文**必须**包含 PUBLISH 报文中发送的原始报文标识符。
[MQTT-2.2.1-6]	SUBACK 和 UNSUBACK 报文**必须**包含相应的 SUBSCRIBE 和 UNSUBSCRIBE 报文中使用的报文标识符。
[MQTT-2.2.2-1]	如果没有属性，属性长度**必须**为 0。
[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]	当协议错误并关闭网络连接时，服务端**必须**处理客户端发送的第二个 CONNECT 报文。
[MQTT-3.1.2-1]	协议名**必须**是 UTF-8 字符串"MQTT"。如果服务端不想接受 CONNECT，并希望透露它是 MQTT 服务端，它**可以**发送一个包含原因码为 0x84（不支持的协议版本）的 CONNACK 报文，然后**必须**关闭网络连接。

[MQTT-3.1.2-2]	如果协议版本不为 5，且服务端不想接受 CONNECT 报文，则服务端**可以**发送一个包含原因码为 0x84（不支持的协议版本）的 CONNACK 报文，然后**必须**关闭网络连接。
[MQTT-3.1.2-3]	服务端**必须**验证 CONNECT 报文的保留标志位（第 0 位）是否为 0。
[MQTT-3.1.2-4]	如果 CONNECT 报文的新开始标志被设置为 1，则客户端和服务端**必须**丢弃任何已存在的会话并开始一个新的会话。
[MQTT-3.1.2-5]	如果 CONNECT 报文的新开始标志被设置为 0，并且存在与该客户标识符相关联的会话，服务端**必须**基于此会话恢复与客户端的通信。
[MQTT-3.1.2-6]	如果 CONNECT 报文的新开始标志被设置为 0，并且不存在与该客户标识符相关联的会话，则服务端**必须**创建一个新的会话。
[MQTT-3.1.2-7]	遗嘱标志被设置为 1，表示遗嘱消息**必须**被存储在服务端并与会话相关联。
[MQTT-3.1.2-8]	在网络连接被关闭且遗嘱延时间隔已过或会话结束时遗嘱消息**必须**被发布，除非遗嘱消息被服务端在收到包含原因码为 0x00（正常关闭）的 DISCONNECT 报文后删除或关于此客户标识符的一个新的网络连接在遗嘱消息间隔过期之前被打开。
[MQTT-3.1.2-9]	如果遗嘱标志被设置为 0，连接标志中的遗嘱 QoS 等级和遗嘱保留字段将会被服务端使用，遗嘱属性、遗嘱主题和遗嘱消息字段**必须**存在于载荷中。
[MQTT-3.1.2-10]	一旦遗嘱消息被发布或者服务端收到包含原因码为 0x00（正常关闭）的 DISCONNECT 报文，遗嘱消息**必须**从服务端的会话中删除。
[MQTT-3.1.2-11]	如果遗嘱标志设置为 0，遗嘱 QoS 等级**必须**也设置为 0 (0x00)。
[MQTT-3.1.2-12]	如果遗嘱标志设置为 1，遗嘱 QoS 等级**可以**被设置为 0（0x00），1（0x01）或 2（0x02）。
[MQTT-3.1.2-13]	如果遗嘱标志被设置为 0，遗嘱保留标志也**必须**设置为 0。
[MQTT-3.1.2-14]	如果遗嘱标志被设置为 1 时，如果遗嘱保留被设置为 0，则服务端**必须**将遗嘱消息当做非保留消息发布。
[MQTT-3.1.2-15]	如果遗嘱保留被设置为 1，则服务端**必须**将遗嘱消息当做保留消息发布。
[MQTT-3.1.2-16]	如果用户名标志被设置为 0，有效载荷中**不能**包含用户名字段。
[MQTT-3.1.2-17]	如果用户名标志被设置为 0，有效载荷中**必须**包含用户名字段。
[MQTT-3.1.2-18]	如果密码标志被设置为 0，有效载荷中**不能**包含密码字段。
[MQTT-3.1.2-19]	如果密码标志被设置为 1，有效载荷中**必须**包含密码字段。
[MQTT-3.1.2-20]	如果保持连接值不为 0，且没有任何其它的 MQTT 控制报文可以发送，客户端**必须**发送一个 PINGREQ 报文。

[MQTT-3.1.2-21]	如果服务端返回的 CONNACK 报文中包含服务端保持连接，客户端**必须**使用此值代替其发送的保持连接。
[MQTT-3.1.2-22]	如果保持连接的值非零，并且服务端在 1.5 倍的保持连接时间内没有收到客户端的 MQTT 控制报文，它**必须**断开客户端的网络连接，并判定网络连接已断开。
[MQTT-3.1.2-23]	如果网络连接关闭时会话过期间隔大于 0，则客户端与服务端**必须**存储会话状态。
[MQTT-3.1.2-24]	服务端**不能**发送超过最大报文长度的报文给客户端。
[MQTT-3.1.2-25]	当报文过大而不能发送时，服务端**必须**丢弃这些报文，然后当做应用消息发送已完成处理。
[MQTT-3.1.2-26]	服务端在一个 PUBLISH 报文中发送的主题别名**不能**超过客户端设置的主题别名最大值。
[MQTT-3.1.2-27]	如果主题别名最大值没有设置，或者设置为零，则服务端**不能**向此客户端发送任何主题别名。
[MQTT-3.1.2-28]	请求响应信息值为 0，表示服务端**不能**返回响应信息。
[MQTT-3.1.2-29]	如果请求问题信息的值为 0，服务端**可以**选择在 CONNACK 或 DISCONNECT 报文中返回原因字符串或用户属性，但**不能**在除 PUBLISH，CONNACK 或 DISCONNECT 之外的报文中发送原因字符串或用户属性。
[MQTT-3.1.2-30]	如果客户端在 CONNECT 报文中设置了认证方法，则客户端在收到 CONNACK 报文之前**不能**发送除 AUTH 或 DISCONNECT 之外的报文。
[MQTT-3.1.3-1]	CONNECT 报文的载荷中包含由可变报头中的标志确定的一个或多个以长度为前缀的字段。这些字段若存在，**必须**按照客户标识符、遗嘱属性、遗嘱主题、遗嘱载荷、用户名、密码的顺序出现。
[MQTT-3.1.3-2]	客户端和服务端都**必须**使用客户标识符识别两者之间的 MQTT 会话相关的状态。
[MQTT-3.1.3-3]	客户标识符**必须**存在，且作为 CONNECT 报文载荷的第一个字段出现。
[MQTT-3.1.3-4]	客户标识符**必须**被编码为 UTF-8 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 编码的客户标识符，客户标识符只能包含这些字符： "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
[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标识符，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标识符给那个客户端。
[MQTT-3.1.3-7]	服务端**必须**假设客户端提供了那个唯一的客户标识符，且**必须**在 CONNACK 报文中返回分配的客户标识符。

[MQTT-3.1.3-8]	如果服务端拒绝了某个客户标识符，它**可以**发送包含原因码 0x85（客户标识符无效）的 CONNACK 报文作为对客户端的 CONNECT 报文的回应，如 4.13 节所述。之后**必须**关闭网络连接。
[MQTT-3.1.3-9]	如果某个会话在遗嘱延时间隔到期之前创建了新的网络连接，则服务端**不能**发送遗嘱消息。
[MQTT-3.1.3-10]	服务端在发布遗嘱消息时**必须**维护用户属性的顺序。
[MQTT-3.1.3-11]	遗嘱主题**必须**为 UTF-8 编码的字符串。
[MQTT-3.1.3-12]	如果用户名标志被设置为 1，用户名为载荷中下一个字段。用户名**必须**是 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，服务端关闭网络连接。
[MQTT-3.1.4-2]	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，**应该**执行身份验证和授权检查。如果任何一项检查没通过，服务端**必须**关闭网络连接。
[MQTT-3.1.4-3]	如果客户标识符所代表的客户端已经连接到此服务端，那么向原有的客户端发送一个包含原因码为 0x8E（会话被接管）的 DISCONNECT 报文，并且**必须**关闭原有的网络连接。
[MQTT-3.1.4-4]	服务端**必须**对新开始标志进行处理。
[MQTT-3.1.4-5]	服务端**必须**使用包含原因码为 0x00（成功）的 CONNACK 报文对客户端的 CONNECT 报文进行确认。
[MQTT-3.1.4-6]	如果服务端拒绝了 CONNECT 报文，它**不能**处理客户端在 CONNECT 报文之后发送的任何除 AUTH 以外的报文。
[MQTT-3.2.0-1]	服务端在发送任何除 AUTH 以外的报文之前**必须**先发送包含原因码为 0x00（成功）的 CONNACK 报文。
[MQTT-3.2.0-2]	服务端在一次网络连接中**不能**发送多个 CONNACK 报文。
[MQTT-3.2.2-1]	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。
[MQTT-3.2.2-2]	如果服务端接受一个新开始为 1 的连接，服务端在 CONNACK 报文中除了把原因码设置为 0x00（成功）之外，还**必须**把会话存在标志设置为 0。
[MQTT-3.2.2-3]	如果服务端接受一个新开始为 0 的连接，并且服务端已经保存了此客户标识符的会话状态，服务端在 CONNACK 报文中**必须**把会话存在标志设置为 1。否则，服务端**必须**把会话存在标志设置为 0。无论如何，服务端在 CONNACK 报文中**必须**把原因码设置为 0x00（成功）。
[MQTT-3.2.2-4]	如果客户端没有保存的会话状态，但收到会话存在标志为 1，客户端**必须**关闭网络连接。
[MQTT-3.2.2-5]	如果客户端保存了会话状态，但收到的会话存在标志为 0，客户端若要继续此网络连接，它**必须**丢弃其保存的会话状态。

[MQTT-3.2.2-6]	如果服务端发送的 CONNACK 报文中原因码非 0，它**必须**把会话存在标志设置为 0。
[MQTT-3.2.2-7]	如果服务端发送了一个包含原因码大于等于 128 的 CONNACK 报文，它随后**必须**关闭网络连接。
[MQTT-3.2.2-8]	服务端发送的 CONNACK 报文**必须**设置一种原因码。
[MQTT-3.2.2-9]	如果服务端不支持 Qos 为 1 或 2 的 PUBLISH 报文，服务端**必须**在 CONNACK 报文中发送最大服务质量以指定其支持的最大 QoS 值。
[MQTT-3.2.2-10]	即使不支持 QoS 为 1 或 2 的 PUBLISH 报文，服务端也**必须**接受请求 QoS 为 0、1 或 2 的 SUBSCRIBE 报文。
[MQTT-3.2.2-11]	如果从服务端接收到了最大 QoS 等级，则客户端**不能**发送超过最大 QoS 等级所指定的 QoS 等级的 PUBLISH 报文。
[MQTT-3.2.2-12]	如果服务端收到包含遗嘱的 QoS 超过服务端处理能力的 CONNECT 报文，服务端**必须**拒绝此连接。服务端**应该**使用包含原因码为 0x9B（不支持的 QoS 等级）的 CONNACK 报文进行错误处理，随后**必须**关闭网络连接。
[MQTT-3.2.2-13]	如果服务端收到一个包含保留标志位 1 的遗嘱消息的 CONNECT 报文且服务端不支持保留消息，服务端**必须**拒绝此连接请求，且**应该**发送包含原因码为 0x9A（不支持保留）的 CONNACK 报文，随后**必须**关闭网络连接。
[MQTT-3.2.2-14]	从服务端接收到的保留可用标志为 0 时，客户端**不能**发送保留标志设置为 1 的 PUBLISH 报文。
[MQTT-3.2.2-15]	客户端**不应该**发送超过最大报文长度的报文给服务端。
[MQTT-3.2.2-16]	如果客户端使用长度为 0 的客户标识符，服务端**必须**回复包含分配客户标识符的 CONNACK 报文。分配客户标识符**必须**是没有被服务端的其他会话所使用的新客户标识符。
[MQTT-3.2.2-17]	客户端在一个 PUBLISH 报文中发送的主题别名值**不能**超过服务端设置的主题别名最大值。
[MQTT-3.2.2-18]	如果主题别名最大值没有设置，或者设置为 0，则客户端**不能**向此服务端发送任何主题别名。
[MQTT-3.2.2-19]	如果加上原因字符串之后的 CONNACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此原因字符串。
[MQTT-3.2.2-20]	如果加上用户属性之后的 CONNACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此属性。
[MQTT-3.2.2-21]	如果服务端发送了服务端保持连接属性，客户端**必须**使用此值代替其在 CONNECT 报文中发送的保持连接时间值。
[MQTT-3.2.2-22]	如果服务端没有发送服务端保持连接属性，服务端**必须**使用客户端在 CONNECT 报文中设置的保持连接时间值。

[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 为 0 的消息，DUP 标志**必须**设置为 0。
[MQTT-3.3.1-3]	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 报文的 2 个 QoS 比特位**不能**同时设置为 1。
[MQTT-3.3.1-5]	如果客户端发给服务端的 PUBLISH 报文的保留标志被设置为 1，服务端**必须**存储此应用消息，并用其替换此话题下任何已存在的消息。
[MQTT-3.3.1-6]	如果载荷为空，消息可以正常被服务端所处理，但是此话题下的任何保留消息**必须**被丢弃，并且此话题未来的订阅者将不会收到保留消息。
[MQTT-3.3.1-7]	载荷为空的保留消息将**不能**被存储在服务端。
[MQTT-3.3.1-8]	如果客户端发给服务端的 PUBLISH 报文的保留标志位为 0，服务器**不能**把此消息存储为保留消息，也**不能**丢弃或替换任何已存在的保留消息。
[MQTT-3.3.1-9]	如果保留消息处理属性被设置为 0，服务端**必须**发送主题与客户端订阅的主题过滤器相匹配的所有保留消息。
[MQTT-3.3.1-10]	如果保留消息处理属性被设置为 1，如果尚不存在匹配的订阅，服务端**必须**发送主题与客户端订阅的主题过滤器相匹配的所有保留消息。如果已存在相匹配的订阅，服务器**不能**发送这些保留消息。
[MQTT-3.3.1-11]	如果保留消息处理属性被设置为 2，服务器**不能**发送这些保留消息。
[MQTT-3.3.1-12]	如果发布保留订阅选项被设置为 0，服务端在转发应用消息时**必须**将保留标志设置为 0，而不管收到的 PUBLISH 报文中保留标志位如何设置的。
[MQTT-3.3.1-13]	如果发布保留订阅选项被设置为 1，服务端在转发应用消息时**必须**将保留标志设置为与收到的 PUBLISH 消息中的保留标志位相同。
[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 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 报文中的主题名**必须**匹配该订阅的主题过滤器。
[MQTT-3.3.2-4]	服务端**必须**把接收到的应用消息中的载荷格式指示原封不动的发给所有的订阅者。
[MQTT-3.3.2-5]	如果消息过期间隔已过期，服务端还没开始向匹配的订阅者交付该消息，则服务端**必须**删除该订阅者的消息副本。
[MQTT-3.3.2-6]	服务端发送给客户端的 PUBLISH 报文中**必须**包含消息过期间隔，值为接收时间减去消息在服务端的等待时间。

[MQTT-3.3.2-7]	接收端**不能**将任何主题别名映射从一个网络连接转发到另一个网络连接。
[MQTT-3.3.2-8]	发送端**不能**发送包含主题别名值为 0 的 PUBLISH 报文。
[MQTT-3.3.2-9]	客户端**不能**发送主题别名值大于服务端的 CONNACK 报文中指定的主题别名最大值的 PUBLISH 报文。
[MQTT-3.3.2-10]	客户端**必须**接受所有值大于 0 且小于等于其发送的 CONNECT 报文中的主题别名最大值的主题别名。
[MQTT-3.3.2-11]	服务端**不能**发送包含主题别名值大于客户端在 CONNECT 报文中指定的主题别名最大值的 PUBLISH 报文。
[MQTT-3.3.2-12]	服务端**必须**接受所有值大于 0 且小于等于其发送的 CONNACK 报文中的主题别名最大值的主题别名。
[MQTT-3.3.2-13]	响应主题**必须**是 UTF-8 编码的字符串。
[MQTT-3.3.2-14]	响应主题**不能**包含通配符。
[MQTT-3.3.2-15]	服务端在收到应用消息时**必须**将响应主题原封不动的发送给所有的订阅者。
[MQTT-3.3.2-16]	服务端在收到应用消息时**必须**原封不动的把对比数据发送给所有的订阅者。
[MQTT-3.3.2-17]	服务端在转发应用消息到客户端时**必须**原封不动的把所有的用户属性放在 PUBLISH 报文中。
[MQTT-3.3.2-18]	服务端在转发应用消息时**必须**保持所有用户属性的先后顺序。
[MQTT-3.3.2-19]	内容类型**必须**是 UTF-8 编码的字符串。
[MQTT-3.3.2-20]	服务端**必须**把收到的应用消息中的内容类型原封不动的发送给所有的订阅者。
[MQTT-3.3.4-1]	PUBLISH 报文的接收端**必须**按照 PUBLISH 报文中的 QoS 等级发送响应报文。
[MQTT-3.3.4-2]	这种情况下，服务端**必须**按照所有匹配的订阅中最大的 QoS 等级把消息发送给客户端。
[MQTT-3.3.4-3]	如果客户端在这些重叠的订阅中指定了订阅标识符，服务端在发布这些订阅相匹配的消息时**必须**包含这些订阅标识符。
[MQTT-3.3.4-4]	如果服务端对这些重叠的订阅只发送一条相匹配的消息，服务端**必须**在 PUBLISH 报文中包含所有的相匹配的订阅标识符（如果存在），但没有顺序要求。
[MQTT-3.3.4-5]	如果服务端对这些重叠的订阅**必须**分别发送相匹配的消息，则每个 PUBLISH 报文中包含与订阅相匹配的订阅标识符（如果存在）。
[MQTT-3.3.4-6]	从客户端发送给服务端的 PUBLISH 报文**不能**包含订阅标识符。

[MQTT-3.3.4-7]	客户端在收到服务端的 PUBACK，PUBCOMP 或包含原因码大于等于 128 的 PUBREC 报文之前，**不能**发送数量超过服务端的接收最大值的 QoS 为 1 和 2 的 PUBLISH 报文。
[MQTT-3.3.4-8]	客户端**不能**延迟发送任何报文，除了 PUBLISH 报文--如果已发送且没有收到确认的 PUBLISH 报文数量已达到服务端的接收最大值。
[MQTT-3.3.4-9]	服务端在接收到客户端的 PUBACK，PUBCOMP 或包含原因码大于等于 128 的 PUBREC 报文之前，**不能**发送数量超过客户端的接收最大值的 QoS 为 1 和 2 的 PUBLISH 报文。
[MQTT-3.3.4-10]	服务端**不能**延迟发送任何报文，除了 PUBLISH 报文--如果已发送且没有收到确认的 PUBLISH 报文数量已到达客户端的接收最大值。
[MQTT-3.4.2-1]	服务端或客户端发送 PUBACK 报文时**必须**设置其中一种 PUBACK 原因码。
[MQTT-3.4.2-2]	如果加上原因字符串之后的 PUBACK 报文长度超出了接收端指定的最大报文长度，则发送端**不能**发送此原因字符串。
[MQTT-3.4.2-3]	如果加上用户属性之后的 PUBACK 报文长度超出了接收端指定的最大报文长度，则发送端**不能**发送此属性。
[MQTT-3.5.2-1]	服务端或客户端发送 PUBREC 报文时**必须**设置其中一种原因码。
[MQTT-3.5.2-2]	发送端使用此值向接收端提供附加信息。如果加上原因字符串之后的 PUBREC 报文长度超出了接收端指定的最大报文长度，则发送端**不能**发送此属性。
[MQTT-3.5.2-3]	如果加上用户属性之后的 PUBREC 报文长度超出了接收端指定的最大报文长度，则发送端**不能**发送此属性。
[MQTT-3.6.1-1]	PUBREL 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-3.6.2-1]	客户端或服务端发送 PUBREL 报文时**必须**设置其中一种 PUBREL 原因码。
[MQTT-3.6.2-2]	如果加上原因字符串之后的 PUBREL 报文长度超出了接收端指定的最大报文长度，则发送端**不能**发送此原因字符串。
[MQTT-3.6.2-3]	如果加上用户属性之后的 PUBREL 报文长度超出了接收端指定的最大报文长度，则发送端**不能**发送此属性。
[MQTT-3.7.2-1]	服务端或客户端发送 PUBCOMP 报文时**必须**设置一种 PUBCOMP 原因码。
[MQTT-3.7.2-2]	如果加上原因字符串之后的 PUBCOMP 报文长度超出了接收端指定的最大报文长度，则发送端**不能**发送此原因字符串。
[MQTT-3.7.2-3]	如果加上用户属性之后的 PUBCOMP 报文长度超出了接收端指定的最大报文长度，则发送端**不能**发送此属性。
[MQTT-3.8.1-1]	SUBSCRIBE 报文固定报头第 3，2，1，0 比特位是保留位，**必须**被设置为 0，0，1，0。服务端**必须**将其他的任何值都当做是不合法的并关闭网络连接。
[MQTT-3.8.3-1]	主题过滤器**必须**为 UTF-8 编码的字符串。

[MQTT-3.8.3-2]	载荷**必须**包含至少一个主题过滤器/订阅选项对。
[MQTT-3.8.3-3]	订阅选项的第 2 比特表示非本地选项。值为 1，表示应用消息**不能**被转发给发布此消息的客户标识符。
[MQTT-3.8.3-4]	共享订阅时把非本地选项设为 1 将造成协议错误。
[MQTT-3.8.3-5]	订阅选项的第 6 和 7 比特为将来所保留。服务端**必须**把此保留位非 0 的 SUBSCRIBE 报文当做无效报文。
[MQTT-3.8.4-1]	当服务端收到来自客户端的 SUBSCRIBE 报文时，**必须**使用 SUBACK 报文作为相应。
[MQTT-3.8.4-2]	SUBACK 报文**必须**和待确认的 SUBSCRIBE 报文有相同的报文标识符。
[MQTT-3.8.4-3]	如果服务端收到的 SUBSCRIBE 报文中的一个主题过滤器与当前会话的一个非共享订阅相同，那么**必须**使用新的订阅替换现存的订阅。
[MQTT-3.8.4-4]	如果保留处理选项为 0，任何匹配该主题过滤器的保留消息**必须**被重发，但替换订阅不能造成应用消息的丢失。
[MQTT-3.8.4-5]	如果服务端收到的 SUBSCRIBE 报文包含多个主题过滤器，服务端**必须**当做收到一系列多个 SUBSCRIBE 报文来处理--除了将它们的响应组合为单个 SUBACK 响应。
[MQTT-3.8.4-6]	服务端发送给客户端的 SUBACK 报文**必须**为每一个主题过滤器/订阅选项对包含一个原因码。
[MQTT-3.8.4-7]	此原因码**必须**说明为该订阅授予的最大 QoS 等级，或指示订阅失败。
[MQTT-3.8.4-8]	响应该订阅的应用消息 QoS 等级**必须**为该消息发布时的 QoS 等级和服务端授予的最大 QoS 等级二者最小值。
[MQTT-3.9.2-1]	如果加上原因字符串之后的 SUBACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此原因字符串。
[MQTT-3.9.2-2]	如果加上用户属性之后的 SUBACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此属性。
[MQTT-3.9.3-1]	SUBACK 报文中的原因码顺序**必须**与 SUBSCRIBE 报文中的主题过滤器顺序相匹配。
[MQTT-3.9.3-2]	服务端发送 SUBACK 报文时**必须**对收到的每一个主题过滤器设置一种原因码。
[MQTT-3.10.1-1]	UNSUBSCRIBE 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**为 UTF-8 编码的字符串。
[MQTT-3.10.3-2]	UNSUBSCRIBE 报文有效载荷**必须**包含至少一个主题过滤器。

[MQTT-3.10.4-1]	服务端**必须**对客户端的 UNSUBSCRIBE 报文中提供的主题过滤器（不管是否包含通配符）逐个字符与当前持有的主题过滤器集进行比较。如果任何过滤器完全匹配，则**必须**删除其拥有的订阅。
[MQTT-3.10.4-2]	当服务端收到 UNSUBSCRIBE 报文，它**必须**停止添加为了交付给客户端的与主题过滤器相匹配的任何新消息。
[MQTT-3.10.4-3]	当服务端收到 UNSUBSCRIBE 报文，它**必须**完成任何已经开始发送给客户端的、与主题过滤器相匹配的、QoS 等级为 1 或 2 的消息。
[MQTT-3.10.4-4]	服务端**必须**发送 UNSUBACK 报文以响应客户端的 UNSUBSCRIBE 请求。
[MQTT-3.10.4-5]	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。
[MQTT-3.10.4-6]	如果服务端收到的 UNSUBSCRIBE 报文包含多个主题过滤器，服务端**必须**当做收到一系列多个 UNSUBSCRIBE 报文来处理--除了将它们的响应组合为单个 SUBACK 响应。
[MQTT-3.11.2-1]	如果加上原因字符串之后的 UNSUBACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此原因字符串。
[MQTT-3.11.2-2]	如果加上用户属性之后的 UNSUBACK 报文长度超出了客户端指定的最大报文长度，则服务端**不能**发送此属性。
[MQTT-3.11.3-1]	UNSUBACK 报文中的原因码顺序**必须**与 UNSUBSCRIBE 报文中的主题过滤器顺序相匹配。
[MQTT-3.11.3-2]	服务端发送 UNSUBACK 报文时对于每个收到的主题过滤器，**必须**使用一个取消订阅原因码。
[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-3.14.0-1]	服务端**不能**发送 DISCONNECT 报文，直到它发送了包含原因码小于 0x80 的 CONNACK 报文之后。
[MQTT-3.14.1-1]	服务端或客户端**必须**验证所有的保留位都被设置为 0，如果他们不为 0，发送包含原因码为 0x81（无效报文）的 DISCONNECT 报文。
[MQTT-3.14.2-1]	客户端或服务端发送 DISCONNECT 报文时**必须**使用一种 DISCONNECT 原因码。
[MQTT-3.14.2-2]	会话过期间隔**不能**由服务端的 DISCONNECT 报文发送。
[MQTT-3.14.2-3]	如果此属性使得 DISCONNECT 报文的长度超出了接收端指定的最大报文长度，则发送端**不能**发送此属性。
[MQTT-3.14.2-4]	如果加上用户属性之后的 DISCONNECT 报文长度超出了接收端指定的最大报文长度，则发送端**不能**发送此属性。
[MQTT-3.14.4-1]	发送端发送完 DISCONNECT 报文之后**不能**再在此网络连接上发送任何 MQTT 控制报文。

[MQTT-3.14.4-2]	发送端发送完 DISCONNECT 报文之后**必须**关闭网络连接。
[MQTT-3.14.4-3]	接收到包含原因码为 0x00（成功）的 DISCONNECT 时，服务端**必须**丢弃任何与当前连接相关的遗嘱消息，而不发布它。
[MQTT-3.15.1-1]	AUTH 报文固定报头第 3，2，1，0 位是保留位，**必须**全设置为 0。客户端或服务端**必须**把其他值当做无效值并关闭网络连接。
[MQTT-3.15.2-1]	AUTH 报文的发送端**必须**使用一种认证原因码。
[MQTT-3.15.2-2]	如果加上原因字符串之后的 AUTH 报文长度超出了接收端所指定的最大报文长度，则发送端**不能**发送此属性。
[MQTT-3.15.2-3]	如果加上用户属性之后的 AUTH 报文长度超出了接收端指定的最大报文长度，则服务端**不能**发送此属性。
[MQTT-4.1.0-1]	当网络连接打开时，客户端和服务端**不能**丢弃会话状态。
[MQTT-4.2.0-1]	客户端或服务端**必须**支持使用一个或多个提供有序的、可靠的、双向传输（从客户端到服务端和从服务端到客户端）字节流传输的底层传输协议。
[MQTT-4.1.0-2]	当网络连接被关闭并且会话过期间隔已过时，服务端**必须**丢弃会话状态。
[MQTT-4.3.1-1]	对于 QoS 等级 0 的分发协议，发送端**必须**发送 QoS 等于 0，DUP 等于 0 的 PUBLISH 报文。
[MQTT-4.3.2-1]	对于 QoS 等级 1 的分发协议，发送端每次发送新的应用消息都**必须**分配一个未使用的用户标识符。
[MQTT-4.3.2-2]	对于 QoS 等级 1 的分发协议，发送端发送的 PUBLISH 报文**必须**包含报文标识符且 QoS 等于 1，DUP 等于 0。
[MQTT-4.3.2-3]	对于 QoS 等级 1 的分发协议，发送端**必须**将这个 PUBLISH 报文看作是*未确认的*，直到从接收端那收到对应的 PUBACK 报文。
[MQTT-4.3.2-4]	对于 QoS 等级 1 的分发协议，接收端响应的 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。
[MQTT-4.3.2-5]	对于 QoS 等级 1 的分发协议，接收端发送了 PUBACK 报文之后，接收端**必须**将任何包含相同报文标识符的入站 PUBLISH 报文当做一个新的消息，并忽略它的 DUP 标志的值。
[MQTT-4.3.3-1]	对于 QoS 等级 2 的分发协议，发送端**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。
[MQTT-4.3.3-2]	对于 QoS 等级 2 的分发协议，发送端 PUBLISH 报文**必须**包含报文标识符且报文的 QoS 等于 2，DUP 等于 0。
[MQTT-4.3.3-3]	对于 QoS 等级 2 的分发协议，发送端**必须**将这个 PUBLISH 报文看作是*未确认的*，直到从接收端那收到对应的 PUBREC 报文。

[MQTT-4.3.3-4]	对于 QoS 等级 2 的分发协议，收到发送端发送的包含原因码小于 0x80 的 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。
[MQTT-4.3.3-5]	对于 QoS 等级 2 的分发协议，发送端**必须**将这个 PUBREL 报文看作是*未确认的*，直到从接收端那收到对应的 PUBCOMP 报文。
[MQTT-4.3.3-6]	对于 QoS 等级 2 的分发协议，发送端一旦发送了对应的 PUBREL 报文就**不能**重发这个 PUBLISH 报文。
[MQTT-4.3.3-7]	对于 QoS 等级 2 的分发协议，如果 PUBLISH 报文已发送，**不能**应用消息过期属性。
[MQTT-4.3.3-8]	对于 QoS 等级 2 的分发协议，接收端响应的 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。
[MQTT-4.3.3-9]	对于 QoS 等级 2 的分发协议，如果接收端发送了包含原因码大于等于 0x80 的 PUBREC 报文，它**必须**将后续包含相同报文标识符的 PUBLISH 报文当做是新的应用消息。
[MQTT-4.3.3-10]	对于 QoS 等级 2 的分发协议，接收端在收到对应的 PUBREL 报文之前，接收端**必须**发送 PUBREC 报文确认任何后续的具有相同报文标识符的 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。
[MQTT-4.3.3-11]	对于 QoS 等级 2 的分发协议，接收端**必须**发送包含与 PUBREL 相同报文标识符的 PUBCOMP 报文作为对 PUBREL 报文的响应。
[MQTT-4.3.3-12]	对于 QoS 等级 2 的分发协议，接收端发送 PUBCOMP 报文之后，**必须**将后续包含相同报文标识符的 PUBLISH 报文当做是新的应用消息。
[MQTT-4.3.3-13]	对于 QoS 等级 2 的分发协议，接收端**必须**继续 QoS 等级 2 确认序列，即使它已经应用了消息过期属性。
[MQTT-4.4.0-1]	客户端以新开始标志为 0 且会话存在的情况下重连时，客户端和服务端都**必须**使用原始报文标识符重新发送任何未被确认的 PUBLISH 报文（当 QoS > 0）和 PUBREL 报文。这是唯一**要求**客户端或服务端重发消息的情况。客户端和服务端**不能**在其他任何时间重发消息。
[MQTT-4.4.0-2]	如果收到包含原因码大于等于 0x80 的 PUBACK 或 PUBREC，则对应的 PUBLISH 报文被看作已确认，且**不能**被重传。
[MQTT-4.5.0-1]	当服务端接受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。
[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，不管它是否选择处理其包含的应用消息。
[MQTT-4.6.0-1]	重发任何之前的 PUBLISH 报文时，客户端**必须**按原始 PUBLISH 报文的发送顺序重发（适用于 QoS 等级 1 和 QoS 等级 2 消息）。
[MQTT-4.6.0-2]	客户端**必须**按照对应的 PUBLISH 报文的顺序发送 PUBACK 报文（QoS 等级 1 消息）。
[MQTT-4.6.0-3]	客户端**必须**按照对应的 PUBLISH 报文的顺序发送 PUBREC 报文（QoS 等级 2 消息）。

[MQTT-4.6.0-4]	客户端**必须**按照对应的 PUBREC 报文的顺序发送 PUBREL 报文（QoS 等级 2 消息）。
[MQTT-4.6.0-5]	当服务端处理发布到有序主题的消息时，它**必须**按照消息从任何给定客户端接收的顺序发送 PUBLISH 报文给消费端（对于同一主题和 QoS 等级）。
[MQTT-4.6.0-6]	默认情况下，服务端转发非共享订阅的消息时，**必须**将每个主题都视为有序主题。
[MQTT-4.7.0-1]	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-4.7.1-1]	多层通配符**必须**单独指定，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]	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。在使用它时，它**必须**占据过滤器的整个层级。
[MQTT-4.7.2-1]	服务端**不能**将$字符开头的主题名匹配通配符（#或+）开头的主题过滤器。
[MQTT-4.7.3-1]	所有的主题名和主题过滤器**必须**至少包含一个字符。
[MQTT-4.7.3-2]	主题名和主题过滤器**不能**包含空字符（Unicode U+0000）。
[MQTT-4.7.3-3]	主题名和主题过滤器是 UTF-8 编码字符串，它们**不能**超过 65,535 字节。
[MQTT-4.7.3-4]	匹配订阅时，服务端**不能**对主题名或主题过滤器执行任何规范化处理，**不能**修改或替换任何未识别的字符。
[MQTT-4.8.2-1]	共享订阅主题过滤器**必须**以"$share/"开始，且**必须**包含至少一个字符长度的共享名。
[MQTT-4.8.2-2]	共享名不能包含字符"/"，"+"或"#"，但**必须**跟在"/"字符后面。此"/"字符后面**必须**跟随一个主题过滤器。
[MQTT-4.8.2-3]	向客户端发送应用消息时，服务端**必须**考虑授予客户端的 QoS 等级。
[MQTT-4.8.2-4]	服务端**必须**在客户端重新连接时完成向该客户端的消息分发。
[MQTT-4.8.2-5]	如果客户端的会话在客户端重连之前终止，服务端**不能**把此消息发送给其他订阅的客户端。
[MQTT-4.8.2-6]	如果客户端对来自服务端的 PUBLISH 报文使用包含原因码大于等于 0x80 的 PUBACK 或 PUBREC 报文进行响应，服务端**必须**丢弃应用消息而不尝试将其发送给任何其他订阅者。
[MQTT-4.9.0-1]	客户端或服务端**必须**将其初始发送配额设置为不超过接收最大值的非 0 值。
[MQTT-4.9.0-2]	每当客户端或服务端发送了一个 QoS 等级大于 0 的 PUBLISH 报文，它就会减少发送配额。如果发送配额减为 0，客户端或服务端**不能**再发送任何 QoS 等级大于 0 的 PUBLISH 报文。

[MQTT-4.9.0-3]	它可以继续发送 QoS 为 0 的 PUBLISH 报文，也可以选择暂停发送这些报文。即使配额为 0，客户端和服务端也**必须**继续处理和响应其他 MQTT 控制报文。
[MQTT-4.12.0-1]	如果服务端不支持客户端提供的认证方法，它**可以**发送一个包含原因码 0x8C（无效的认证方法）或 0x87（未授权）的 CONNACK 报文，并且**必须**关闭网络连接。
[MQTT-4.12.0-2]	如果服务端需要额外的信息来完成认证，它可以向客户端发送 AUTH 报文，此报文**必须**包含原因码 0x18（继续认证）。
[MQTT-4.12.0-3]	客户端通过发送另一个 AUTH 报文响应来自服务端的 AUTH 报文，此报文**必须**包含原因码 0x18（继续认证）。
[MQTT-4.12.0-4]	服务端可以在处理过程中随时拒绝认证。它**可以**发送包含原因码大于等于 0x80 的 CONNACK 报文，如 4.13 节所述，并且**必须**关闭网络连接。
[MQTT-4.12.0-5]	如果初始 CONNECT 报文包含认证方法属性，则所有的 AUTH 报文和成功的 CONNACK 报文**必须**包含与 CONNECT 报文中相同的认证方法属性。
[MQTT-4.12.0-6]	如果客户端在 CONNECT 报文中没有包含认证方法，则服务端**不能**发送 AUTH 报文，且不能在 CONNACK 报文中发送认证方法。
[MQTT-4.12.0-7]	如果客户端在 CONNECT 报文中没有包含认证方法，则客户端**不能**向服务端发送 AUTH 报文。
[MQTT-4.12.1-1]	如果客户端在 CONNECT 报文中提供了认证方法，它可以在收到 CONNACK 报文之后的任何时间通过发送包含原因码 0x19（重新认证）的 AUTH 报文发起重新认证。客户端**必须**将认证方法设置为与最初验证网络连接时的认证方法一致。
[MQTT-4.12.1-2]	如果重新认证失败，客户端或服务端**应该**发送包含适当原因码的 DISCONNECT 报文，如 section 4.13 节 所述。并且**必须**关闭网络连接。
[MQTT-4.13.1-1]	当服务端检测到无效报文或协议错误，并且本规范中给出了相应的原因码时，它**必须**关闭网络连接。
[MQTT-4.13.2-1]	CONNACK 报文和 DISCONNECT 报文允许使用大于等于 0x80 的原因码以指示网络连接将被关闭。如果某个大于等于 0x80 的原因码被指定，无论是否发送 CONNACK 报文或 DISCONNECT 报文，**必须**关闭网络连接。
[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"。

Appendix C. MQTT v5.0 新特性总结（非规范）

MQTT v5.0添加了以下特性

- 会话过期
  把清理会话标志拆分成新开始标志（指示会话应该在不使用现有会话的情况下开始）和会话过期间隔标志（指示连接断开之后会话保留的时间）。会话过期间隔时间可以在断开时修改。把新开始标志设置为1且会话过期间隔标志设置为0，等同于在MQTT v3.1.1中把清理会话（CleanSession）设置为1。

- 消息过期
  允许消息在发布时设置一个过期间隔。

- 所有确认报文原因码
  更改所有响应报文以包含原因码，包括CONNACK，PUBACK，PUBREC，PUBREL，PUBCOMP，SUBACK，UNSUBACK，DISCONNECT和AUTH，以使得调用方确定请求的函数是否成功。

- 所有确认报文原因字符串
  更改大部分报文以包含原因码同时也允许一个可选的原因字符串。这是为问题定位而设计的，并且不应由接收端所解析。

- 服务端断开
  允许服务端发送DISCONNECT报文，以指示连接被关闭的原因。

- 载荷格式和内容类型
  允许在消息发布时指定载荷格式（二进制、文本）和MIME样式内容类型。这些信息被转发到消息的接收端。

- 请求/响应
  规定MQTT请求/响应模式，提供响应主题和对比数据属性，以使得响应消息被路由回请求的发布者。此外，为客户端添加从服务端获取获取关于构造响应主题的配置信息的能力。

- 共享订阅
  添加对共享订阅的支持，以允许多个订阅消费者进行负载均衡。

- 订阅标识符
  允许在SUBSCRIBE报文中指定一个数字订阅标识符，并在消息分发时返回此标识符。这使得客户端收到分发的消息时确定此消息是由哪个或哪些订阅导致的。

- 主题别名
  通过将主题名缩写为小整数来减小MQTT报文的开销大小。客户端和服务端分别指定它们允许的主题别名的数量。

- 流量控制
  允许客户端和服务端分别指定未完成的可靠消息（QoS>0）的数量。发送端可以暂停发送此类消息以保持消息数量低于配额。这被用于限制可靠消息的速率和某一时刻的传输中（in-flight）消息数量。

- 用户属性
  为大多数报文添加用户属性。PUBLISH报文的用户属性由客户端应用程序定义。PUBLISH报文和遗嘱报文的用户属性由服务端转发给应用消息的接收端。CONNECT，SUBSCRIBE和UNSUBSCRIBE报文

的用户属性由服务端实现定义。CONNACK，PUBACK，PUBREC，PUBREL，PUBCOMP，SUBACK，UNSUBACK和AUTH报文的用户属性由发送端定义，且对发送端具有唯一性。MQTT规范不定义用户属性的意义。

- 最大报文长度
  允许客户端和服务端各自指定它们支持的最大报文长度。会话参与方发送更大的报文将造成错误。

- 可选的服务端功能可用性
  提供定义一组服务端不允许的功能，并告知客户端的机制。可以使用这种方式指定的功能包括：最大QoS等级，保留可用，通配符订阅可用，订阅标识符可用和共享订阅可用。客户端使用服务端通知了（不可用）的功能将造成错误。

  在早期版本的MQTT协议中，服务端没有实现的功能通过未授权告知客户端。当客户端使用其中一种（不可用的）功能时，此功能允许服务端告知客户端，并添加特定的原因码。

- 增强的认证
  提供一种机制来启用包括互相认证在内的质询/响应风格的认证。这允许在客户端和服务端都支持的情况下使用SASL风格的认证，包括客户端在连接中重新认证的功能。

- 订阅选项
  提供主要用于定义允许消息桥接应用的订阅选项。包括不要把消息发送给消息源客户端（非本地）的选项和订阅时处理保留消息的选项。

- 遗嘱延迟
  提供指定遗嘱消息在连接中断后延时发送的能力。设计此特性是为了在会话的连接重建的情况下不发送遗嘱消息。此特性允许连接短暂中断而不通知其他客户端。

- 服务端保持连接
  允许服务端指定其希望客户端使用的保持连接值。此特性允许服务端设置最大允许的保持连接值并被客户端使用。

- 分配客户标识符
  服务端分配了客户标识符的情况下，向客户端返回此客户标识符。服务端分配客户标识符只能用于新开始标志为1的连接。

- 服务端参考

  允许服务端使用 CONNACK 或 DISCONNECT 报文指定备用服务端。此特性被用于（服务端）重定向或做准备。