超文本传输协议（HTTP/1.1）：消息语法和路由

一、摘要

超文本传输协议（HTTP）是分布式，协作式超文本信息系统的无状态应用程序级别协议。本文档概述了 HTTP 体系结构及其相关术语，定义了“http”和“https”统一资源标识符（URI）方案，定义了 HTTP / 1.1 消息语法和解析要求，并描述了实现的相关安全性问题。

二、介绍

超文本传输协议（HTTP）是一种无状态的应用程序级请求/响应协议，它使用可扩展的语义和自描述消息有效负载来与基于网络的超文本信息系统进行灵活的交互。该文档是共同构成 HTTP / 1.1 规范的一系列文档中的第一篇：

1、消息语法和路由（本片文档）

2、语义和内容（RFC7231）

3、条件请求（RFC7232）

4、范围请求（RFC7233）

5、缓存（RFC7234）

6、认证（RFC7235）

此 HTTP / 1.1 规范淘汰了 RFC 2616 和 RFC 2145（基于 HTTP 版本）。该规范还更新了 CONNECT 的用法，以建立先前在 RFC 2817 中定义的隧道，并定义了在 RFC 2818 中非正式描述的“https”URI 方案。

HTTP 是消息系统的一个通用的接口协议。它旨在通过向客户端呈现与提供的资源类型无关的统一接口来隐藏服务实现方式的详细信息。同样，服务器不需要知道每个客户端的用途：可以将 HTTP 请求视为孤立的请求，而不是与特定类型的客户端或应用程序的预定顺序相关联。结果是可以在许多不同的上下文中有效使用的协议，其实现可以随时间独立发展。

HTTP 还被设计为用作中介协议，用于与非 HTTP 信息系统之间进行通信转换。HTTP 代理和网关可以通过将其各种协议转换为超文本格式来提供对替代信息服务的访问，这些超文本格式可以由客户端以与 HTTP 服务相同的方式进行查看和操作。

这种灵活性的结果是，无法根据接口后面发生的事情来定义协议。相反，我们仅限于定义通信的语法，接收到的通信的意图以及接收者的预期行为。如果孤立地考虑通信，那么成功的动作应该反映在服务器提供的可观察接口的相应更改

中。但是，由于多个客户可能并行运行，甚至有可能互为目的，所以我们不能要求这种变化超出单个响应的范围即可观察到。

本文档描述了 HTTP 中使用或引用的体系结构元素，定义了“http”和“https” URI 方案，描述了整个网络操作和连接管理，并定义了 HTTP 消息成帧和转发要求。我们的目标是定义与消息语义无关的 HTTP 消息处理所需的所有机制，从而定义消息解析器和消息转发中介的完整要求集。

三、客户端与服务器消息

HTTP 是一种无状态请求与响应协议，可以在可靠的传输层或会话层连接之间交换消息进行操作。HTTP 客户端是一种为了发送一个或多个 HTTP 请求而与服务器建立连接的程序。HTTP 服务器是接受连接以便通过发送 HTTP 响应来服务 HTTP 请求的程序。

术语“客户端”和“服务器”仅指这些程序为特定连接执行的角色。同一程序可能在某些连接上充当客户端，而在另一些连接上充当服务器。术语“用户代理”指的是发起请求的各种客户端程序中的任何一个，包括（但不限于）浏览器，Spider（基于 Web 的机器人），命令行工具，自定义应用程序和移动应用程序。术语“源服务器”是指可以针对给定目标资源发起权威响应的程序。术语“发送者”和“收件人”分别是指发送或接收给定消息的任何实现。

HTTP 依赖于统一资源标识符（URI）标准来指示目标资源以及资源之间的关系。消息以类似于 Internet 邮件[RFC5322]和多用途 Internet 邮件扩展（MIME）[RFC2045]使用的格式传递（有关 HTTP 和 MIME 消息之间的区别，请参阅[RFC7231]的附录 A）。

大多数 HTTP 通信都包含一个检索请求（GET），用于表示由 URI 标识的某些资源。在最简单的情况下，这可以通过用户代理（UA）与原始服务器（O）之间的单个双向连接（===）来实现。

```
          请求     >
UA ======================================= O
                                    <     响应
```

客户端以请求消息的形式向服务器发送 HTTP 请求，该请求以包括方法，URI 和协议版本（第 3.1.1 节）的请求行开头，后跟包含请求修饰符，客户端信息的标头字段 ，表示元数据（第 3.2 节），指示标头部分结尾的空行，最后是包含有效内容主体（如果有的话，第 3.3 节）的消息主体。

服务器通过发送一个或多个 HTTP 响应消息来响应客户端的请求，每个 HTTP 响应消息都以一个状态行开头，该状态行包括协议版本，成功或错误代码以及文本原因短语（第 3.1.2 节），可能后跟标头字段 包含服务器信息，资源元数据和表示形式元数据（第 3.2 节），用于指示标头部分结尾的空行，最后是包含有效内容主体（如果有的话，第 3.3 节）的消息主体。

四、实现多样性

考虑 HTTP 的设计时，很容易陷入陷阱，认为所有用户代理都是通用浏览器，所有原始服务器都是大型公共网站。实际上不是这种情况。常见的 HTTP 用户代理包括各种形状和尺寸的家用电器，立体声音响，秤，固件更新脚本，命令行程序，移动应用程序和通信设备。同样，常见的 HTTP 原始服务器包括家庭自动化单元，可配置的网络组件，办公机器，自主机器人，新闻源，交通摄像机，广告选择器和视频投放平台。

术语“用户代理”并不意味着在请求时有人类用户直接与软件代理进行交互。在许多情况下，用户代理被安装或配置为在后台运行，并保存其结果以供以后检查（或仅保存可能有趣或错误的结果的子集）。例如，通常会将蜘蛛指定给起始 URI，并将其配置为在将 Web 作为超文本图进行爬网时遵循某些行为。

HTTP 的实现多样性意味着并非所有的用户代理都可以向其用户提出交互式建议或针对安全性或隐私问题提供适当的警告。在此规范要求向用户报告错误的少数情况下，仅在错误控制台或日志文件中才可以观察到这种报告是可以接受的。同样，可以通过预先配置选择，运行时选项或简单避免不安全操作来满足用户在继续操作之前确认自动操作的要求；确认并不意味着任何特定的用户界面或正常处理的中断（如果用户已做出选择）。

五、中介程序

HTTP 允许使用中介程序通过连接链满足请求。HTTP 中介有三种常见的形式：代理、网关和隧道。在某些情况下，单个中介可以充当源服务器，代理，网关或隧道，根据每个请求的性质来切换行为。

```
          >             >             >             >
UA ========== A ========== B ========== C ========== O
          <             <             <             <
```

上图显示了用户代理和原始服务器之间的三个中介（A，B 和 C）。遍及整个链的请求或响应消息将通过四个单独的连接。某些 HTTP 通信选项可能仅适用于

与最近的非隧道邻居的连接，仅适用于链的端点或适用于链中的所有连接。尽管该图是线性的，但每个参与者都可能参与多个同时进行的通信。例如，B 正在处理 A 的同时可能正在接收来自 A 以外的许多客户端的请求，可以将请求转发到 C 以外的服务器。同样的，之后的请求可能基于动态配置的负载均衡，被通过其他路径的连接所发送。

术语“上游”和“下游”用于描述与消息流有关的方向性要求：所有消息从上游流向下游。术语“入站”和“出站”用于描述与请求路由有关的方向性要求：“入站”是指指向原始服务器、“出站”是指指向用户代理。

“代理”是消息转发代理，客户端通常通过本地配置规则选择该代理来接收对某些绝对 URI 类型的请求，并尝试通过 HTTP 接口进行转换来满足这些请求。某些转换是最少的，例如针对“http”URI 的代理请求，而其他请求可能需要与完全不同的应用程序级协议进行转换。出于安全性，注释服务或共享缓存的目的，代理通常用于通过通用中介对组织的 HTTP 请求进行分组。如第 5.7.2 节中所述，某些代理被设计为在转发所选消息或有效负载时将其转换。

“网关”（也称为“反向代理”）是充当出站连接的原始服务器，但转换接收到的请求并将其入站转发到另一个服务器的中介。网关通常用于封装旧的或不受信任的信息服务，以通过“加速器”缓存提高服务器性能，并在多台计算机之间实现 HTTP 服务的分区或负载平衡。

适用于原始服务器的所有 HTTP 要求也适用于网关的出站通信。网关使用其所需的任何协议与入站服务器进行通信，包括本规范范围之外的 HTTP 专用扩展。但是，希望与第三方 HTTP 服务器进行互操作的 HTTP 到 HTTP 网关应该符合用户代理对网关入站连接的要求。

“隧道”充当两个连接之间的盲中继，而不更改消息。一旦激活，隧道就不会被认为是 HTTP 通信的一方，尽管该隧道可能是由 HTTP 请求发起的。当中继连接的两端都关闭时，隧道不再存在。隧道用于通过中介扩展虚拟连接，例如，当传输层安全性（TLS，[RFC5246]）用于通过共享防火墙代理建立机密通信时。

上面的中介类别仅考虑那些充当 HTTP 通信参与者的角色。还有一些中介可以在网络协议栈的较低层起作用，在没有消息发送者的知识或许可的情况下过滤或重定向 HTTP 流量。网络中间人（在协议级别）与中间人攻击是无法区分的，由于错误地违反了 HTTP 语义，经常会引入安全漏洞或互操作性问题。

例如，“拦截代理”[RFC3040]（通常也称为“透明代理”[RFC1919]或“强制门户”）与 HTTP 代理不同，因为客户端未选择它。取而代之的是，拦截代理会过滤或重定向传出的 TCP 端口 80 数据包（有时还会重定向其他常见端口流量）。拦截代理通常在公共网络访问点上找到，作为在允许使用非本地 Internet 服务之前强制执行用户订阅的方法，并且在公司防火墙内可以强制执行网络使用策略。

HTTP 被定义为无状态协议，这意味着可以单独理解每个请求消息。许多实现依赖于 HTTP 的无状态设计，以便在多个服务器之间重用代理连接或动态负载平衡请求。因此，除非连接是安全的并且特定于该代理，否则服务器不得假定同一连接上的两个请求来自同一用户代理。已知某些非标准的 HTTP 扩展名（例如[RFC4559]）违反了此要求，从而导致安全性和互操作性问题。