

HTTP Working Group
INTERNET-DRAFT
<draft-ietf-http-v10-spec-05.txt>
Expires August 19, 1996

T. Berners-Lee, MIT/LCS
R. Fielding, UC Irvine
H. Frystyk, MIT/LCS
February 19, 1996

Hypertext Transfer Protocol -- HTTP/1.0

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

To learn the current status of any Internet-Draft, please check the "lid-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited. Please send comments to the HTTP working group at <http-wg@cuckoo.hpl.hp.com>. Discussions of the working group are archived at <URL:http://www.ics.uci.edu/pub/ietf/http/>. General discussions about HTTP and the applications which use HTTP should take place on the <www-talk@w3.org> mailing list.

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods (commands). A feature of HTTP is the typing of data representation, allowing systems to be built independently of the data being transferred.

HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification reflects common usage of the protocol referred to as "HTTP/1.0".

Table of Contents

1. Introduction
 - 1.1 Purpose
 - 1.2 Terminology
 - 1.3 Overall Operation
 - 1.4 HTTP and MIME
2. Notational Conventions and Generic Grammar
 - 2.1 Augmented BNF
 - 2.2 Basic Rules

3. Protocol Parameters
 - 3.1 HTTP Version
 - 3.2 Uniform Resource Identifiers
 - 3.2.1 General Syntax
 - 3.2.2 http URL
 - 3.3 Date/Time Formats
 - 3.4 Character Sets
 - 3.5 Content Codings
 - 3.6 Media Types
 - 3.6.1 Canonicalization and Text Defaults
 - 3.6.2 Multipart Types
 - 3.7 Product Tokens
4. HTTP Message
 - 4.1 Message Types
 - 4.2 Message Headers
 - 4.3 General Header Fields
5. Request
 - 5.1 Request-Line
 - 5.1.1 Method
 - 5.1.2 Request-URI
 - 5.2 Request Header Fields
6. Response
 - 6.1 Status-Line
 - 6.1.1 Status Code and Reason Phrase
 - 6.2 Response Header Fields
7. Entity
 - 7.1 Entity Header Fields
 - 7.2 Entity Body
 - 7.2.1 Type
 - 7.2.2 Length
8. Method Definitions
 - 8.1 GET
 - 8.2 HEAD
 - 8.3 POST
9. Status Code Definitions
 - 9.1 Informational 1xx
 - 9.2 Successful 2xx
 - 9.3 Redirection 3xx
 - 9.4 Client Error 4xx
 - 9.5 Server Error 5xx
10. Header Field Definitions
 - 10.1 Allow
 - 10.2 Authorization
 - 10.3 Content-Encoding
 - 10.4 Content-Length
 - 10.5 Content-Type
 - 10.6 Date
 - 10.7 Expires
 - 10.8 From
 - 10.9 If-Modified-Since
 - 10.10 Last-Modified
 - 10.11 Location
 - 10.12 Pragma

- 10.13 Referer
- 10.14 Server
- 10.15 User-Agent
- 10.16 WWW-Authenticate

- 11. Access Authentication
 - 11.1 Basic Authentication Scheme

- 12. Security Considerations
 - 12.1 Authentication of Clients
 - 12.2 Safe Methods
 - 12.3 Abuse of Server Log Information
 - 12.4 Transfer of Sensitive Information
 - 12.5 Attacks Based On File and Path Names

- 13. Acknowledgments

- 14. References

- 15. Authors' Addresses

- Appendix A. Internet Media Type message/http

- Appendix B. Tolerant Applications

- Appendix C. Relationship to MIME
 - C.1 Conversion to Canonical Form
 - C.2 Conversion of Date Formats
 - C.3 Introduction of Content-Encoding
 - C.4 No Content-Transfer-Encoding
 - C.5 HTTP Header Fields in Multipart Body-Parts

- Appendix D. Additional Features
 - D.1 Additional Request Methods
 - D.1.1 PUT
 - D.1.2 DELETE
 - D.1.3 LINK
 - D.1.4 UNLINK
 - D.2 Additional Header Field Definitions
 - D.2.1 Accept
 - D.2.2 Accept-Charset
 - D.2.3 Accept-Encoding
 - D.2.4 Accept-Language
 - D.2.5 Content-Language
 - D.2.6 Link
 - D.2.7 MIME-Version
 - D.2.8 Retry-After
 - D.2.9 Title
 - D.2.10 URI

1. Introduction

1.1 Purpose

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification reflects common usage of the protocol referred

to as "HTTP/1.0". This specification describes the features that seem to be consistently implemented in most HTTP/1.0 clients and servers. The specification is split into two sections. Those features of HTTP for which implementations are usually consistent are described in the main body of this document. Those features which have few or inconsistent implementations are listed in [Appendix D](#).

Practical information systems require more functionality than simple retrieval, including search, front-end update, and annotation. HTTP allows an open-ended set of methods to be used to indicate the purpose of a request. It builds on the discipline of reference provided by the Uniform Resource Identifier (URI) [2], as a location (URL) [4] or name (URN) [16], for indicating the resource on which a method is to be applied. Messages are passed in a format similar to that used by Internet Mail [7] and the Multipurpose Internet Mail Extensions (MIME) [5].

HTTP is also used as a generic protocol for communication between user agents and proxies/gateways to other Internet protocols, such as SMTP [12], NNTP [11], FTP [14], Gopher [1], and WAIS [8], allowing basic hypermedia access to resources available from diverse applications and simplifying the implementation of user agents.

1.2 Terminology

This specification uses a number of terms to refer to the roles played by participants in, and objects of, the HTTP communication.

connection

A transport layer virtual circuit established between two application programs for the purpose of communication.

message

The basic unit of HTTP communication, consisting of a structured sequence of octets matching the syntax defined in [Section 4](#) and transmitted via the connection.

request

An HTTP request message (as defined in [Section 5](#)).

response

An HTTP response message (as defined in [Section 6](#)).

resource

A network data object or service which can be identified by a URI ([Section 3.2](#)).

entity

A particular representation or rendition of a data resource, or reply from a service resource, that may be enclosed within a request or response message. An entity consists of meta-information in the form of entity headers and content in the form of an entity body.

client

An application program that establishes connections for the purpose of sending requests.

user agent

The client which initiates a request. These are often browsers, editors, spiders (web-traversing robots), or other end user tools.

server

An application program that accepts connections in order to service requests by sending back responses.

origin server

The server on which a given resource resides or is to be created.

proxy

An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them, with possible translation, on to other servers. A proxy must interpret and, if necessary, rewrite a request message before forwarding it. Proxies are often used as client-side portals through network firewalls and as helper applications for handling requests via protocols not implemented by the user agent.

gateway

A server which acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the origin server for the requested resource; the requesting client may not be aware that it is communicating with a gateway. Gateways are often used as server-side portals through network firewalls and as protocol translators for access to resources stored on non-HTTP systems.

tunnel

A tunnel is an intermediary program which is acting as a blind relay between two connections. Once active, a tunnel is not considered a party to the HTTP communication, though the tunnel may have been initiated by an HTTP request. The tunnel ceases to exist when both ends of the relayed connections are closed. Tunnels are used when a portal is necessary and the intermediary cannot, or should not, interpret the relayed communication.

cache

A program's local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cachable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, though a cache cannot be used by a server while it is acting as a tunnel.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts



Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research



With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips



Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.