

EXHIBIT U

Case No. 1:14-cv-00857-TSC-DAR

Network Working Group
Request for Comments: 2616
Obsoletes: 2068
Category: Standards Track

R. Fielding
UC Irvine
J. Gettys
Compaq/W3C
J. Mogul
Compaq
H. Frystyk
W3C/MIT
L. Masinter
Xerox
P. Leach
Microsoft
T. Berners-Lee
W3C/MIT
June 1999

Hypertext Transfer Protocol -- HTTP/1.1

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers [47]. A feature of HTTP is the typing and negotiation 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 defines the protocol referred to as "HTTP/1.1", and is an update to RFC 2068 [33].

Fielding, et al.

Standards Track

[Page 1]

Table of Contents

1	Introduction	7
1.1	Purpose.....	7
1.2	Requirements	8
1.3	Terminology	8
1.4	Overall Operation	12
2	Notational Conventions and Generic Grammar	14
2.1	Augmented BNF	14
2.2	Basic Rules	15
3	Protocol Parameters	17
3.1	HTTP Version	17
3.2	Uniform Resource Identifiers	18
3.2.1	General Syntax	19
3.2.2	http URL	19
3.2.3	URI Comparison	20
3.3	Date/Time Formats	20
3.3.1	Full Date	20
3.3.2	Delta Seconds	21
3.4	Character Sets	21
3.4.1	Missing Charset	22
3.5	Content Codings	23
3.6	Transfer Codings	24
3.6.1	Chunked Transfer Coding	25
3.7	Media Types	26
3.7.1	Canonicalization and Text Defaults	27
3.7.2	Multipart Types	27
3.8	Product Tokens	28
3.9	Quality Values	29
3.10	Language Tags	29
3.11	Entity Tags	30
3.12	Range Units	30
4	HTTP Message	31
4.1	Message Types	31
4.2	Message Headers	31
4.3	Message Body	32
4.4	Message Length	33
4.5	General Header Fields	34
5	Request	35
5.1	Request-Line	35
5.1.1	Method	36
5.1.2	Request-URI	36
5.2	The Resource Identified by a Request	38
5.3	Request Header Fields	38
6	Response	39
6.1	Status-Line	39
6.1.1	Status Code and Reason Phrase	39
6.2	Response Header Fields	41

Fielding, et al.

Standards Track

[Page 2]

7	Entity	42
7.1	Entity Header Fields	42
7.2	Entity Body	43
7.2.1	Type	43
7.2.2	Entity Length	43
8	Connections	44
8.1	Persistent Connections	44
8.1.1	Purpose	44
8.1.2	Overall Operation	45
8.1.3	Proxy Servers	46
8.1.4	Practical Considerations	46
8.2	Message Transmission Requirements	47
8.2.1	Persistent Connections and Flow Control	47
8.2.2	Monitoring Connections for Error Status Messages	48
8.2.3	Use of the 100 (Continue) Status	48
8.2.4	Client Behavior if Server Prematurely Closes Connection ..	50
9	Method Definitions	51
9.1	Safe and Idempotent Methods	51
9.1.1	Safe Methods	51
9.1.2	Idempotent Methods	51
9.2	OPTIONS	52
9.3	GET	53
9.4	HEAD	54
9.5	POST	54
9.6	PUT	55
9.7	DELETE	56
9.8	TRACE	56
9.9	CONNECT	57
10	Status Code Definitions	57
10.1	Informational 1xx	57
10.1.1	100 Continue	58
10.1.2	101 Switching Protocols	58
10.2	Successful 2xx	58
10.2.1	200 OK	58
10.2.2	201 Created	59
10.2.3	202 Accepted	59
10.2.4	203 Non-Authoritative Information	59
10.2.5	204 No Content	60
10.2.6	205 Reset Content	60
10.2.7	206 Partial Content	60
10.3	Redirection 3xx	61
10.3.1	300 Multiple Choices	61
10.3.2	301 Moved Permanently	62
10.3.3	302 Found	62
10.3.4	303 See Other	63
10.3.5	304 Not Modified	63
10.3.6	305 Use Proxy	64
10.3.7	306 (Unused)	64

Fielding, et al.

Standards Track

[Page 3]

- 10.3.8 307 Temporary Redirect65
- 10.4 Client Error 4xx65
 - 10.4.1 400 Bad Request65
 - 10.4.2 401 Unauthorized66
 - 10.4.3 402 Payment Required66
 - 10.4.4 403 Forbidden66
 - 10.4.5 404 Not Found66
 - 10.4.6 405 Method Not Allowed66
 - 10.4.7 406 Not Acceptable67
 - 10.4.8 407 Proxy Authentication Required67
 - 10.4.9 408 Request Timeout67
 - 10.4.10 409 Conflict67
 - 10.4.11 410 Gone68
 - 10.4.12 411 Length Required68
 - 10.4.13 412 Precondition Failed68
 - 10.4.14 413 Request Entity Too Large69
 - 10.4.15 414 Request-URI Too Long69
 - 10.4.16 415 Unsupported Media Type69
 - 10.4.17 416 Requested Range Not Satisfiable69
 - 10.4.18 417 Expectation Failed70
- 10.5 Server Error 5xx70
 - 10.5.1 500 Internal Server Error70
 - 10.5.2 501 Not Implemented70
 - 10.5.3 502 Bad Gateway70
 - 10.5.4 503 Service Unavailable70
 - 10.5.5 504 Gateway Timeout71
 - 10.5.6 505 HTTP Version Not Supported71
- 11 Access Authentication71
- 12 Content Negotiation71
 - 12.1 Server-driven Negotiation72
 - 12.2 Agent-driven Negotiation73
 - 12.3 Transparent Negotiation74
- 13 Caching in HTTP74
 - 13.1.1 Cache Correctness75
 - 13.1.2 Warnings76
 - 13.1.3 Cache-control Mechanisms77
 - 13.1.4 Explicit User Agent Warnings78
 - 13.1.5 Exceptions to the Rules and Warnings78
 - 13.1.6 Client-controlled Behavior79
 - 13.2 Expiration Model79
 - 13.2.1 Server-Specified Expiration79
 - 13.2.2 Heuristic Expiration80
 - 13.2.3 Age Calculations80
 - 13.2.4 Expiration Calculations83
 - 13.2.5 Disambiguating Expiration Values84
 - 13.2.6 Disambiguating Multiple Responses84
 - 13.3 Validation Model85
 - 13.3.1 Last-Modified Dates86

Fielding, et al.

Standards Track

[Page 4]

13.3.2	Entity Tag Cache Validators	86
13.3.3	Weak and Strong Validators	86
13.3.4	Rules for When to Use Entity Tags and Last-Modified Dates	89
13.3.5	Non-validating Conditionals	90
13.4	Response Cacheability	91
13.5	Constructing Responses From Caches	92
13.5.1	End-to-end and Hop-by-hop Headers	92
13.5.2	Non-modifiable Headers	92
13.5.3	Combining Headers	94
13.5.4	Combining Byte Ranges	95
13.6	Caching Negotiated Responses	95
13.7	Shared and Non-Shared Caches	96
13.8	Errors or Incomplete Response Cache Behavior	97
13.9	Side Effects of GET and HEAD	97
13.10	Invalidation After Updates or Deletions	97
13.11	Write-Through Mandatory	98
13.12	Cache Replacement	99
13.13	History Lists	99
14	Header Field Definitions	100
14.1	Accept	100
14.2	Accept-Charset	102
14.3	Accept-Encoding	102
14.4	Accept-Language	104
14.5	Accept-Ranges	105
14.6	Age	106
14.7	Allow	106
14.8	Authorization	107
14.9	Cache-Control	108
14.9.1	What is Cacheable	109
14.9.2	What May be Stored by Caches	110
14.9.3	Modifications of the Basic Expiration Mechanism	111
14.9.4	Cache Revalidation and Reload Controls	113
14.9.5	No-Transform Directive	115
14.9.6	Cache Control Extensions	116
14.10	Connection	117
14.11	Content-Encoding	118
14.12	Content-Language	118
14.13	Content-Length	119
14.14	Content-Location	120
14.15	Content-MD5	121
14.16	Content-Range	122
14.17	Content-Type	124
14.18	Date	124
14.18.1	Clockless Origin Server Operation	125
14.19	ETag	126
14.20	Expect	126
14.21	Expires	127
14.22	From	128

Fielding, et al.

Standards Track

[Page 5]

- 14.23 Host128
- 14.24 If-Match129
- 14.25 If-Modified-Since130
- 14.26 If-None-Match132
- 14.27 If-Range133
- 14.28 If-Unmodified-Since134
- 14.29 Last-Modified134
- 14.30 Location135
- 14.31 Max-Forwards136
- 14.32 Pragma136
- 14.33 Proxy-Authenticate137
- 14.34 Proxy-Authorization137
- 14.35 Range138
- 14.35.1 Byte Ranges138
- 14.35.2 Range Retrieval Requests139
- 14.36 Referer140
- 14.37 Retry-After141
- 14.38 Server141
- 14.39 TE142
- 14.40 Trailer143
- 14.41 Transfer-Encoding.....143
- 14.42 Upgrade144
- 14.43 User-Agent145
- 14.44 Vary145
- 14.45 Via146
- 14.46 Warning148
- 14.47 WWW-Authenticate150
- 15 Security Considerations150
- 15.1 Personal Information.....151
- 15.1.1 Abuse of Server Log Information151
- 15.1.2 Transfer of Sensitive Information151
- 15.1.3 Encoding Sensitive Information in URI's152
- 15.1.4 Privacy Issues Connected to Accept Headers152
- 15.2 Attacks Based On File and Path Names153
- 15.3 DNS Spoofing154
- 15.4 Location Headers and Spoofing154
- 15.5 Content-Disposition Issues154
- 15.6 Authentication Credentials and Idle Clients155
- 15.7 Proxies and Caching155
- 15.7.1 Denial of Service Attacks on Proxies.....156
- 16 Acknowledgments156
- 17 References158
- 18 Authors' Addresses162
- 19 Appendices164
- 19.1 Internet Media Type message/http and application/http164
- 19.2 Internet Media Type multipart/byteranges165
- 19.3 Tolerant Applications166
- 19.4 Differences Between HTTP Entities and RFC 2045 Entities167

Fielding, et al.

Standards Track

[Page 6]

19.4.1	MIME-Version	167
19.4.2	Conversion to Canonical Form	167
19.4.3	Conversion of Date Formats	168
19.4.4	Introduction of Content-Encoding	168
19.4.5	No Content-Transfer-Encoding	168
19.4.6	Introduction of Transfer-Encoding	169
19.4.7	MHTML and Line Length Limitations	169
19.5	Additional Features	169
19.5.1	Content-Disposition	170
19.6	Compatibility with Previous Versions	170
19.6.1	Changes from HTTP/1.0	171
19.6.2	Compatibility with HTTP/1.0 Persistent Connections	172
19.6.3	Changes from RFC 2068	172
20	Index	175
21	Full Copyright Statement	176

1 Introduction

1.1 Purpose

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990. The first version of HTTP, referred to as HTTP/0.9, was a simple protocol for raw data transfer across the Internet. HTTP/1.0, as defined by RFC 1945 [6], improved the protocol by allowing messages to be in the format of MIME-like messages, containing meta-information about the data transferred and modifiers on the request/response semantics. However, HTTP/1.0 does not sufficiently take into consideration the effects of hierarchical proxies, caching, the need for persistent connections, or virtual hosts. In addition, the proliferation of incompletely-implemented applications calling themselves "HTTP/1.0" has necessitated a protocol version change in order for two communicating applications to determine each other's true capabilities.

This specification defines the protocol referred to as "HTTP/1.1". This protocol includes more stringent requirements than HTTP/1.0 in order to ensure reliable implementation of its features.

Practical information systems require more functionality than simple retrieval, including search, front-end update, and annotation. HTTP allows an open-ended set of methods and headers that indicate the purpose of a request [47]. It builds on the discipline of reference provided by the Uniform Resource Identifier (URI) [3], as a location (URL) [4] or name (URN) [20], for indicating the resource to which a

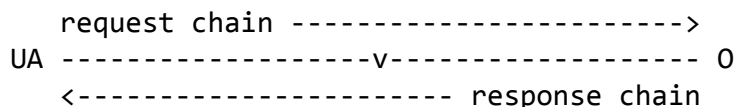
inbound/outbound

Inbound and outbound refer to the request and response paths for messages: "inbound" means "traveling toward the origin server", and "outbound" means "traveling toward the user agent"

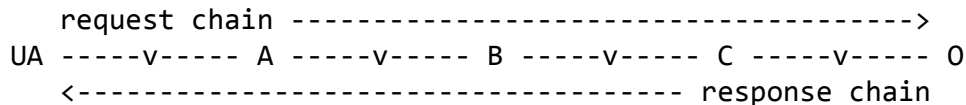
1.4 Overall Operation

The HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a connection with a server. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible entity-body content. The relationship between HTTP and MIME is described in appendix 19.4.

Most HTTP communication is initiated by a user agent and consists of a request to be applied to a resource on some origin server. In the simplest case, this may be accomplished via a single connection (v) between the user agent (UA) and the origin server (O).



A more complicated situation occurs when one or more intermediaries are present in the request/response chain. There are three common forms of intermediary: proxy, gateway, and tunnel. A proxy is a forwarding agent, receiving requests for a URI in its absolute form, rewriting all or part of the message, and forwarding the reformatted request toward the server identified by the URI. A gateway is a receiving agent, acting as a layer above some other server(s) and, if necessary, translating the requests to the underlying server's protocol. A tunnel acts as a relay point between two connections without changing the messages; tunnels are used when the communication needs to pass through an intermediary (such as a firewall) even when the intermediary cannot understand the contents of the messages.



The figure above shows three intermediaries (A, B, and C) between the user agent and origin server. A request or response message that travels the whole chain will pass through four separate connections. This distinction is important because some HTTP communication options

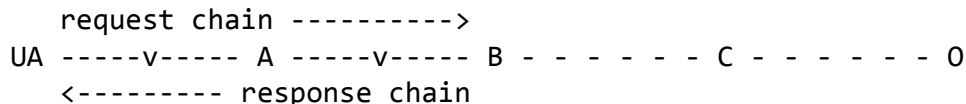
Fielding, et al.

Standards Track

[Page 12]

may apply only to the connection with the nearest, non-tunnel neighbor, only to the end-points of the chain, or to all connections along the chain. Although the diagram is linear, each participant may be engaged in multiple, simultaneous communications. For example, B may be receiving requests from many clients other than A, and/or forwarding requests to servers other than C, at the same time that it is handling A's request.

Any party to the communication which is not acting as a tunnel may employ an internal cache for handling requests. The effect of a cache is that the request/response chain is shortened if one of the participants along the chain has a cached response applicable to that request. The following illustrates the resulting chain if B has a cached copy of an earlier response from O (via C) for a request which has not been cached by UA or A.



Not all responses are usefully cacheable, and some requests may contain modifiers which place special requirements on cache behavior. HTTP requirements for cache behavior and cacheable responses are defined in section 13.

In fact, there are a wide variety of architectures and configurations of caches and proxies currently being experimented with or deployed across the World Wide Web. These systems include national hierarchies of proxy caches to save transoceanic bandwidth, systems that broadcast or multicast cache entries, organizations that distribute subsets of cached data via CD-ROM, and so on. HTTP systems are used in corporate intranets over high-bandwidth links, and for access via PDAs with low-power radio links and intermittent connectivity. The goal of HTTP/1.1 is to support the wide diversity of configurations already deployed while introducing protocol constructs that meet the needs of those who build web applications that require high reliability and, failing that, at least reliable indications of failure.

HTTP communication usually takes place over TCP/IP connections. The default port is TCP 80 [19], but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used; the mapping of the HTTP/1.1 request and response structures onto the transport data units of the protocol in question is outside the scope of this specification.

Fielding, et al.

Standards Track

[Page 13]