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].

IPR2025-01572
Apple EX1046 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

IPR2025-01572
Apple EX1046 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

IPR2025-01572
Apple EX1046 Page 3

10.3.8 307 Temporary Redirect 65
10.4 Client Error 4xx 65
10.4.1 400 Bad Request 65
10.4.2 401 Unauthorized 66
10.4.3 402 Payment Required 66
10.4.4 403 Forbidden 66
10.4.5 404 Not Found 66
10.4.6 405 Method Not Allowed 66
10.4.7 406 Not Acceptable 67
10.4.8 407 Proxy Authentication Required 67
10.4.9 408 Request Timeout 67
10.4.10 409 Conflict 67
10.4.11 410 Gone 68
10.4.12 411 Length Required 68
10.4.13 412 Precondition Failed 68
10.4.14 413 Request Entity Too Large 69
10.4.15 414 Request-URI Too Long 69
10.4.16 415 Unsupported Media Type 69
10.4.17 416 Requested Range Not Satisfiable 69
10.4.18 417 Expectation Failed 70
10.5 Server Error 5xx 70
10.5.1 500 Internal Server Error 70
10.5.2 501 Not Implemented 70
10.5.3 502 Bad Gateway 70
10.5.4 503 Service Unavailable 70
10.5.5 504 Gateway Timeout 71
10.5.6 505 HTTP Version Not Supported 71
11 Access Authentication 71
12 Content Negotiation 71
12.1 Server-driven Negotiation 72
12.2 Agent-driven Negotiation 73
12.3 Transparent Negotiation 74
13 Caching in HTTP 74
13.1.1 Cache Correctness 75
13.1.2 Warnings 76
13.1.3 Cache-control Mechanisms 77
13.1.4 Explicit User Agent Warnings 78
13.1.5 Exceptions to the Rules and Warnings 78
13.1.6 Client-controlled Behavior 79
13.2 Expiration Model 79
13.2.1 Server-Specified Expiration 79
13.2.2 Heuristic Expiration 80
13.2.3 Age Calculations 80
13.2.4 Expiration Calculations 83
13.2.5 Disambiguating Expiration Values 84
13.2.6 Disambiguating Multiple Responses 84
13.3 Validation Model 85
13.3.1 Last-Modified Dates 86

IPR2025-01572
Apple EX1046 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

IPR2025-01572
Apple EX1046 Page 5

14.23 Host 128
14.24 If-Match 129
14.25 If-Modified-Since 130
14.26 If-None-Match 132
14.27 If-Range 133
14.28 If-Unmodified-Since 134
14.29 Last-Modified 134
14.30 Location 135
14.31 Max-Forwards 136
14.32 Pragma 136
14.33 Proxy-Authenticate 137
14.34 Proxy-Authorization 137
14.35 Range 138
14.35.1 Byte Ranges 138
14.35.2 Range Retrieval Requests 139
14.36 Referer 140
14.37 Retry-After 141
14.38 Server 141
14.39 TE 142
14.40 Trailer 143
14.41 Transfer-Encoding 143
14.42 Upgrade 144
14.43 User-Agent 145
14.44 Vary 145
14.45 Via 146
14.46 Warning 148
14.47 WWW-Authenticate 150
15 Security Considerations 150
15.1 Personal Information 151
15.1.1 Abuse of Server Log Information 151
15.1.2 Transfer of Sensitive Information 151
15.1.3 Encoding Sensitive Information in URI's 152
15.1.4 Privacy Issues Connected to Accept Headers 152
15.2 Attacks Based On File and Path Names 153
15.3 DNS Spoofing 154
15.4 Location Headers and Spoofing 154
15.5 Content-Disposition Issues 154
15.6 Authentication Credentials and Idle Clients 155
15.7 Proxies and Caching 155
15.7.1 Denial of Service Attacks on Proxies 156
16 Acknowledgments 156
17 References 158
18 Authors' Addresses 162
19 Appendices 164
19.1 Internet Media Type message/http and application/http 164
19.2 Internet Media Type multipart/byteranges 165
19.3 Tolerant Applications 166
19.4 Differences Between HTTP Entities and RFC 2045 Entities 167

IPR2025-01572
Apple EX1046 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 metainformation 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

IPR2025-01572
Apple EX1046 Page 7

method is to be applied. Messages are passed in a format similar to that used by Internet mail [9] as defined by the Multipurpose Internet Mail Extensions (MIME) [7].

HTTP is also used as a generic protocol for communication between user agents and proxies/gateways to other Internet systems, including those supported by the SMTP [16], NNTP [13], FTP [18], Gopher [2], and WAIS [10] protocols. In this way, HTTP allows basic hypermedia access to resources available from diverse applications.

1.2 Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [34].

An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements for the protocols it implements. An implementation that satisfies all the MUST or REQUIRED level and all the SHOULD level requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST level requirements but not all the SHOULD level requirements for its protocols is said to be "conditionally compliant."

1.3 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 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.

IPR2025-01572
Apple EX1046 Page 8

resource
A network data object or service that can be identified by a URI, as defined in section 3.2. Resources may be available in multiple representations (e.g. multiple languages, data formats, size, and resolutions) or vary in other ways.

entity
The information transferred as the payload of a request or response. An entity consists of metainformation in the form of entity-header fields and content in the form of an entity-body, as described in section 7.

representation
An entity included with a response that is subject to content negotiation, as described in section 12. There may exist multiple representations associated with a particular response status.

content negotiation
The mechanism for selecting the appropriate representation when servicing a request, as described in section 12. The representation of entities in any response can be negotiated (including error responses).

variant
A resource may have one, or more than one, representation(s) associated with it at any given instant. Each of these representations is termed a `varriant'. Use of the term `variant' does not necessarily imply that the resource is subject to content negotiation.

client
A 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. Any given program may be capable of being both a client and a server; our use of these terms refers only to the role being performed by the program for a particular connection, rather than to the program's capabilities in general. Likewise, any server may act as an origin server, proxy, gateway, or tunnel, switching behavior based on the nature of each request.

IPR2025-01572
Apple EX1046 Page 9

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 on, with possible translation, to other servers. A proxy MUST implement both the client and server requirements of this specification. A "transparent proxy" is a proxy that does not modify the request or response beyond what is required for proxy authentication and identification. A "non-transparent proxy" is a proxy that modifies the request or response in order to provide some added service to the user agent, such as group annotation services, media type transformation, protocol reduction, or anonymity filtering. Except where either transparent or non-transparent behavior is explicitly stated, the HTTP proxy requirements apply to both types of proxies.

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.

tunnel
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.

cache
A program's local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cacheable 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 that is acting as a tunnel.

cacheable
A response is cacheable if a cache is allowed to store a copy of the response message for use in answering subsequent requests. The rules for determining the cacheability of HTTP responses are defined in section 13. Even if a resource is cacheable, there may be additional constraints on whether a cache can use the cached copy for a particular request.

IPR2025-01572
Apple EX1046 Page 10

first-hand
A response is first-hand if it comes directly and without unnecessary delay from the origin server, perhaps via one or more proxies. A response is also first-hand if its validity has just been checked directly with the origin server.

explicit expiration time
The time at which the origin server intends that an entity should no longer be returned by a cache without further validation.

heuristic expiration time
An expiration time assigned by a cache when no explicit expiration time is available.

age
The age of a response is the time since it was sent by, or successfully validated with, the origin server.

freshness lifetime
The length of time between the generation of a response and its expiration time.

fresh
A response is fresh if its age has not yet exceeded its freshness lifetime.

stale
A response is stale if its age has passed its freshness lifetime.

semantically transparent
A cache behaves in a "semantically transparent" manner, with respect to a particular response, when its use affects neither the requesting client nor the origin server, except to improve performance. When a cache is semantically transparent, the client receives exactly the same response (except for hop-by-hop headers) that it would have received had its request been handled directly by the origin server.

validator
A protocol element (e.g., an entity tag or a Last-Modified time) that is used to find out whether a cache entry is an equivalent copy of an entity.

upstream/downstream
Upstream and downstream describe the flow of a message: all messages flow from upstream to downstream.

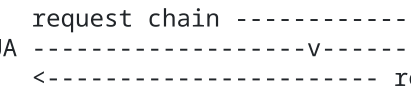
IPR2025-01572
Apple EX1046 Page 11

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"

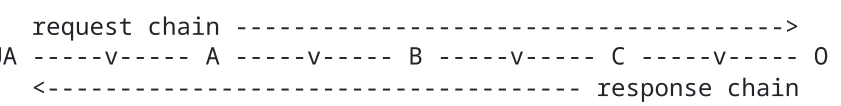
1.4 Overall Operation

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

IPR2025-01572
Apple EX1046 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.

```
          request chain ---------->
       UA -----v----- A -----v----- B - - - - - - C - - - - - - - O
          <--------- response chain
```

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.

IPR2025-01572
Apple EX1046 Page 13

message being transmitted.

```
    general-header = Cache-Control            ; Section 14.9
                   | Connection               ; Section 14.10
                   | Date                     ; Section 14.18
                   | Pragma                   ; Section 14.32
                   | Trailer                  ; Section 14.40
                   | Transfer-Encoding        ; Section 14.41
                   | Upgrade                  ; Section 14.42
                   | Via                      ; Section 14.45
                   | Warning                  ; Section 14.46
```

General-header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields may be given the semantics of general header fields if all parties in the communication recognize them to be general-header fields. Unrecognized header fields are treated as entity-header fields.

5 Request

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

```
     Request       = Request-Line              ; Section 5.1
                     *(( general-header        ; Section 4.5
                      | request-header         ; Section 5.3
                      | entity-header ) CRLF)  ; Section 7.1
                     CRLF
                     [ message-body ]          ; Section 4.3
```

5.1 Request-Line

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

```
     Request-Line   = Method SP Request-URI SP HTTP-Version CRLF
```


IPR2025-01572
Apple EX1046 Page 14

5.1.1 Method

The Method token indicates the method to be performed on the resource identified by the Request-URI. The method is case-sensitive.

```
    Method         = "OPTIONS"                ; Section 9.2
                   | "GET"                    ; Section 9.3
                   | "HEAD"                   ; Section 9.4
                   | "POST"                   ; Section 9.5
                   | "PUT"                    ; Section 9.6
                   | "DELETE"                 ; Section 9.7
                   | "TRACE"                  ; Section 9.8
                   | "CONNECT"                ; Section 9.9
                   | extension-method
    extension-method = token
```

The list of methods allowed by a resource can be specified in an Allow header field (section 14.7). The return code of the response always notifies the client whether a method is currently allowed on a resource, since the set of allowed methods can change dynamically. An origin server SHOULD return the status code 405 (Method Not Allowed) if the method is known by the origin server but not allowed for the requested resource, and 501 (Not Implemented) if the method is unrecognized or not implemented by the origin server. The methods GET and HEAD MUST be supported by all general-purpose servers. All other methods are OPTIONAL; however, if the above methods are implemented, they MUST be implemented with the same semantics as those specified in section 9.

5.1.2 Request-URI

The Request-URI is a Uniform Resource Identifier (section 3.2) and identifies the resource upon which to apply the request.

```
    Request-URI    = "*" | absoluteURI | abs_path | authority
```

The four options for Request-URI are dependent on the nature of the request. The asterisk "*" means that the request does not apply to a particular resource, but to the server itself, and is only allowed when the method used does not necessarily apply to a resource. One example would be

```
    OPTIONS * HTTP/1.1
```

The absoluteURI form is REQUIRED when the request is being made to a proxy. The proxy is requested to forward the request or service it from a valid cache, and return the response. Note that the proxy MAY forward the request on to another proxy or directly to the server

IPR2025-01572
Apple EX1046 Page 15

specified by the absoluteURI. In order to avoid request loops, a proxy MUST be able to recognize all of its server names, including any aliases, local variations, and the numeric IP address. An example Request-Line would be:

```
GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1
```

To allow for transition to absoluteURIs in all requests in future versions of HTTP, all HTTP/1.1 servers MUST accept the absoluteURI form in requests, even though HTTP/1.1 clients will only generate them in requests to proxies.

The authority form is only used by the CONNECT method (section 9.9).

The most common form of Request-URI is that used to identify a resource on an origin server or gateway. In this case the absolute path of the URI MUST be transmitted (see section 3.2.1, abs_path) as the Request-URI, and the network location of the URI (authority) MUST be transmitted in a Host header field. For example, a client wishing to retrieve the resource above directly from the origin server would create a TCP connection to port 80 of the host "www.w3.org" and send the lines:

```
GET /pub/WWW/TheProject.html HTTP/1.1
Host: www.w3.org
```

followed by the remainder of the Request. Note that the absolute path cannot be empty; if none is present in the original URI, it MUST be given as "/" (the server root).

The Request-URI is transmitted in the format specified in section 3.2.1. If the Request-URI is encoded using the "% HEX HEX" encoding [42], the origin server MUST decode the Request-URI in order to properly interpret the request. Servers SHOULD respond to invalid Request-URIs with an appropriate status code.

A transparent proxy MUST NOT rewrite the "abs_path" part of the received Request-URI when forwarding it to the next inbound server, except as noted above to replace a null abs_path with "/".

> Note: The "no rewrite" rule prevents the proxy from changing the meaning of the request when the origin server is improperly using a non-reserved URI character for a reserved purpose. Implementors should be aware that some pre-HTTP/1.1 proxies have been known to rewrite the Request-URI.

IPR2025-01572
Apple EX1046 Page 16

5.2 The Resource Identified by a Request

The exact resource identified by an Internet request is determined by examining both the Request-URI and the Host header field.

An origin server that does not allow resources to differ by the requested host MAY ignore the Host header field value when determining the resource identified by an HTTP/1.1 request. (But see section 19.6.1.1 for other requirements on Host support in HTTP/1.1.)

An origin server that does differentiate resources based on the host requested (sometimes referred to as virtual hosts or vanity host names) MUST use the following rules for determining the requested resource on an HTTP/1.1 request:

1. If Request-URI is an absoluteURI, the host is part of the Request-URI. Any Host header field value in the request MUST be ignored.

2. If the Request-URI is not an absoluteURI, and the request includes a Host header field, the host is determined by the Host header field value.

3. If the host as determined by rule 1 or 2 is not a valid host on the server, the response MUST be a 400 (Bad Request) error message.

Recipients of an HTTP/1.0 request that lacks a Host header field MAY attempt to use heuristics (e.g., examination of the URI path for something unique to a particular host) in order to determine what exact resource is being requested.

5.3 Request Header Fields

The request-header fields allow the client to pass additional information about the request, and about the client itself, to the server. These fields act as request modifiers, with semantics equivalent to the parameters on a programming language method invocation.

```
    request-header = Accept                   ; Section 14.1
                   | Accept-Charset           ; Section 14.2
                   | Accept-Encoding          ; Section 14.3
                   | Accept-Language          ; Section 14.4
                   | Authorization            ; Section 14.8
                   | Expect                   ; Section 14.20
                   | From                     ; Section 14.22
                   | Host                     ; Section 14.23
                   | If-Match                 ; Section 14.24
```


IPR2025-01572
Apple EX1046 Page 17

```
                    | If-Modified-Since        ; Section 14.25
                    | If-None-Match           ; Section 14.26
                    | If-Range                ; Section 14.27
                    | If-Unmodified-Since     ; Section 14.28
                    | Max-Forwards            ; Section 14.31
                    | Proxy-Authorization     ; Section 14.34
                    | Range                   ; Section 14.35
                    | Referer                 ; Section 14.36
                    | TE                      ; Section 14.39
                    | User-Agent              ; Section 14.43
```

Request-header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields MAY be given the semantics of request-header fields if all parties in the communication recognize them to be request-header fields. Unrecognized header fields are treated as entity-header fields.

6 Response

After receiving and interpreting a request message, a server responds with an HTTP response message.

```
    Response      = Status-Line               ; Section 6.1
                    *(( general-header        ; Section 4.5
                     | response-header        ; Section 6.2
                     | entity-header ) CRLF)  ; Section 7.1
                    CRLF
                    [ message-body ]          ; Section 7.2
```

6.1 Status-Line

The first line of a Response message is the Status-Line, consisting of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

```
    Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
```

6.1.1 Status Code and Reason Phrase

The Status-Code element is a 3-digit integer result code of the attempt to understand and satisfy the request. These codes are fully defined in section 10. The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automata and the Reason-Phrase is intended for the human user. The client is not required to examine or display the Reason-Phrase.

IPR2025-01572
Apple EX1046 Page 18

The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. There are 5 values for the first digit:

- 1xx: Informational - Request received, continuing process
- 2xx: Success - The action was successfully received, understood, and accepted
- 3xx: Redirection - Further action must be taken in order to complete the request
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error - The server failed to fulfill an apparently valid request

The individual values of the numeric status codes defined for HTTP/1.1, and an example set of corresponding Reason-Phrase's, are presented below. The reason phrases listed here are only recommendations -- they MAY be replaced by local equivalents without affecting the protocol.

```
   Status-Code    =
         "100"  ; Section 10.1.1: Continue
       | "101"  ; Section 10.1.2: Switching Protocols
       | "200"  ; Section 10.2.1: OK
       | "201"  ; Section 10.2.2: Created
       | "202"  ; Section 10.2.3: Accepted
       | "203"  ; Section 10.2.4: Non-Authoritative Information
       | "204"  ; Section 10.2.5: No Content
       | "205"  ; Section 10.2.6: Reset Content
       | "206"  ; Section 10.2.7: Partial Content
       | "300"  ; Section 10.3.1: Multiple Choices
       | "301"  ; Section 10.3.2: Moved Permanently
       | "302"  ; Section 10.3.3: Found
       | "303"  ; Section 10.3.4: See Other
       | "304"  ; Section 10.3.5: Not Modified
       | "305"  ; Section 10.3.6: Use Proxy
       | "307"  ; Section 10.3.8: Temporary Redirect
       | "400"  ; Section 10.4.1: Bad Request
       | "401"  ; Section 10.4.2: Unauthorized
       | "402"  ; Section 10.4.3: Payment Required
       | "403"  ; Section 10.4.4: Forbidden
       | "404"  ; Section 10.4.5: Not Found
       | "405"  ; Section 10.4.6: Method Not Allowed
       | "406"  ; Section 10.4.7: Not Acceptable
```


IPR2025-01572
Apple EX1046 Page 19

```
            | "407"  ; Section 10.4.8: Proxy Authentication Required
            | "408"  ; Section 10.4.9: Request Time-out
            | "409"  ; Section 10.4.10: Conflict
            | "410"  ; Section 10.4.11: Gone
            | "411"  ; Section 10.4.12: Length Required
            | "412"  ; Section 10.4.13: Precondition Failed
            | "413"  ; Section 10.4.14: Request Entity Too Large
            | "414"  ; Section 10.4.15: Request-URI Too Large
            | "415"  ; Section 10.4.16: Unsupported Media Type
            | "416"  ; Section 10.4.17: Requested range not satisfiable
            | "417"  ; Section 10.4.18: Expectation Failed
            | "500"  ; Section 10.5.1: Internal Server Error
            | "501"  ; Section 10.5.2: Not Implemented
            | "502"  ; Section 10.5.3: Bad Gateway
            | "503"  ; Section 10.5.4: Service Unavailable
            | "504"  ; Section 10.5.5: Gateway Time-out
            | "505"  ; Section 10.5.6: HTTP Version not supported
            | extension-code

      extension-code = 3DIGIT
      Reason-Phrase  = *<TEXT, excluding CR, LF>
```

HTTP status codes are extensible. HTTP applications are not required to understand the meaning of all registered status codes, though such understanding is obviously desirable. However, applications MUST understand the class of any status code, as indicated by the first digit, and treat any unrecognized response as being equivalent to the x00 status code of that class, with the exception that an unrecognized response MUST NOT be cached. For example, if an unrecognized status code of 431 is received by the client, it can safely assume that there was something wrong with its request and treat the response as if it had received a 400 status code. In such cases, user agents SHOULD present to the user the entity returned with the response, since that entity is likely to include human-readable information which will explain the unusual status.

6.2 Response Header Fields

The response-header fields allow the server to pass additional information about the response which cannot be placed in the Status-Line. These header fields give information about the server and about further access to the resource identified by the Request-URI.

```
    response-header = Accept-Ranges           ; Section 14.5
                    | Age                     ; Section 14.6
                    | ETag                    ; Section 14.19
                    | Location                ; Section 14.30
                    | Proxy-Authenticate      ; Section 14.33
```


IPR2025-01572
Apple EX1046 Page 20

```
                    | Retry-After            ; Section 14.37
                    | Server                 ; Section 14.38
                    | Vary                   ; Section 14.44
                    | WWW-Authenticate       ; Section 14.47
```

Response-header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields MAY be given the semantics of response-header fields if all parties in the communication recognize them to be response-header fields. Unrecognized header fields are treated as entity-header fields.

7 Entity

Request and Response messages MAY transfer an entity if not otherwise restricted by the request method or response status code. An entity consists of entity-header fields and an entity-body, although some responses will only include the entity-headers.

In this section, both sender and recipient refer to either the client or the server, depending on who sends and who receives the entity.

7.1 Entity Header Fields

Entity-header fields define metainformation about the entity-body or, if no body is present, about the resource identified by the request. Some of this metainformation is OPTIONAL; some might be REQUIRED by portions of this specification.

```
    entity-header  = Allow                  ; Section 14.7
                   | Content-Encoding       ; Section 14.11
                   | Content-Language       ; Section 14.12
                   | Content-Length         ; Section 14.13
                   | Content-Location       ; Section 14.14
                   | Content-MD5            ; Section 14.15
                   | Content-Range          ; Section 14.16
                   | Content-Type           ; Section 14.17
                   | Expires                ; Section 14.21
                   | Last-Modified          ; Section 14.29
                   | extension-header

    extension-header = message-header
```

The extension-header mechanism allows additional entity-header fields to be defined without changing the protocol, but these fields cannot be assumed to be recognizable by the recipient. Unrecognized header fields SHOULD be ignored by the recipient and MUST be forwarded by transparent proxies.

IPR2025-01572
Apple EX1046 Page 21

7.2 Entity Body

The entity-body (if any) sent with an HTTP request or response is in a format and encoding defined by the entity-header fields.

```
    entity-body    = *OCTET
```

An entity-body is only present in a message when a message-body is present, as described in section 4.3. The entity-body is obtained from the message-body by decoding any Transfer-Encoding that might have been applied to ensure safe and proper transfer of the message.

7.2.1 Type

When an entity-body is included with a message, the data type of that body is determined via the header fields Content-Type and Content-Encoding. These define a two-layer, ordered encoding model:

```
    entity-body := Content-Encoding( Content-Type( data ) )
```

Content-Type specifies the media type of the underlying data. Content-Encoding may be used to indicate any additional content codings applied to the data, usually for the purpose of data compression, that are a property of the requested resource. There is no default encoding.

Any HTTP/1.1 message containing an entity-body SHOULD include a Content-Type header field defining the media type of that body. If and only if the media type is not given by a Content-Type field, the recipient MAY attempt to guess the media type via inspection of its content and/or the name extension(s) of the URI used to identify the resource. If the media type remains unknown, the recipient SHOULD treat it as type "application/octet-stream".

7.2.2 Entity Length

The entity-length of a message is the length of the message-body before any transfer-codings have been applied. Section 4.4 defines how the transfer-length of a message-body is determined.

IPR2025-01572
Apple EX1046 Page 22

8 Connections

8.1 Persistent Connections

8.1.1 Purpose

Prior to persistent connections, a separate TCP connection was established to fetch each URL, increasing the load on HTTP servers and causing congestion on the Internet. The use of inline images and other associated data often require a client to make multiple requests of the same server in a short amount of time. Analysis of these performance problems and results from a prototype implementation are available [26] [30]. Implementation experience and measurements of actual HTTP/1.1 (RFC 2068) implementations show good results [39]. Alternatives have also been explored, for example, T/TCP [27].

Persistent HTTP connections have a number of advantages:

- By opening and closing fewer TCP connections, CPU time is saved in routers and hosts (clients, servers, proxies, gateways, tunnels, or caches), and memory used for TCP protocol control blocks can be saved in hosts.

- HTTP requests and responses can be pipelined on a connection. Pipelining allows a client to make multiple requests without waiting for each response, allowing a single TCP connection to be used much more efficiently, with much lower elapsed time.

- Network congestion is reduced by reducing the number of packets caused by TCP opens, and by allowing TCP sufficient time to determine the congestion state of the network.

- Latency on subsequent requests is reduced since there is no time spent in TCP's connection opening handshake.

- HTTP can evolve more gracefully, since errors can be reported without the penalty of closing the TCP connection. Clients using future versions of HTTP might optimistically try a new feature, but if communicating with an older server, retry with old semantics after an error is reported.

HTTP implementations SHOULD implement persistent connections.

IPR2025-01572
Apple EX1046 Page 23

proxy adds a "Expect: 100-continue" field when it forwards a request, then it need not forward the corresponding 100 (Continue) response(s).)

10.1.1 100 Continue

The client SHOULD continue with its request. This interim response is used to inform the client that the initial part of the request has been received and has not yet been rejected by the server. The client SHOULD continue by sending the remainder of the request or, if the request has already been completed, ignore this response. The server MUST send a final response after the request has been completed. See section 8.2.3 for detailed discussion of the use and handling of this status code.

10.1.2 101 Switching Protocols

The server understands and is willing to comply with the client's request, via the Upgrade message header field (section 14.42), for a change in the application protocol being used on this connection. The server will switch protocols to those defined by the response's Upgrade header field immediately after the empty line which terminates the 101 response.

The protocol SHOULD be switched only when it is advantageous to do so. For example, switching to a newer version of HTTP is advantageous over older versions, and switching to a real-time, synchronous protocol might be advantageous when delivering resources that use such features.

10.2 Successful 2xx

This class of status code indicates that the client's request was successfully received, understood, and accepted.

10.2.1 200 OK

The request has succeeded. The information returned with the response is dependent on the method used in the request, for example:

GET an entity corresponding to the requested resource is sent in the response;

HEAD the entity-header fields corresponding to the requested resource are sent in the response without any message-body;

POST an entity describing or containing the result of the action;

IPR2025-01572
Apple EX1046 Page 24

TRACE an entity containing the request message as received by the end server.

10.2.2 201 Created

The request has been fulfilled and resulted in a new resource being created. The newly created resource can be referenced by the URI(s) returned in the entity of the response, with the most specific URI for the resource given by a Location header field. The response SHOULD include an entity containing a list of resource characteristics and location(s) from which the user or user agent can choose the one most appropriate. The entity format is specified by the media type given in the Content-Type header field. The origin server MUST create the resource before returning the 201 status code. If the action cannot be carried out immediately, the server SHOULD respond with 202 (Accepted) response instead.

A 201 response MAY contain an ETag response header field indicating the current value of the entity tag for the requested variant just created, see section 14.19.

10.2.3 202 Accepted

The request has been accepted for processing, but the processing has not been completed. The request might or might not eventually be acted upon, as it might be disallowed when processing actually takes place. There is no facility for re-sending a status code from an asynchronous operation such as this.

The 202 response is intentionally non-committal. Its purpose is to allow a server to accept a request for some other process (perhaps a batch-oriented process that is only run once per day) without requiring that the user agent's connection to the server persist until the process is completed. The entity returned with this response SHOULD include an indication of the request's current status and either a pointer to a status monitor or some estimate of when the user can expect the request to be fulfilled.

10.2.4 203 Non-Authoritative Information

The returned metainformation in the entity-header is not the definitive set as available from the origin server, but is gathered from a local or a third-party copy. The set presented MAY be a subset or superset of the original version. For example, including local annotation information about the resource might result in a superset of the metainformation known by the origin server. Use of this response code is not required and is only appropriate when the response would otherwise be 200 (OK).

IPR2025-01572
Apple EX1046 Page 25

10.2.5 204 No Content

The server has fulfilled the request but does not need to return an entity-body, and might want to return updated metainformation. The response MAY include new or updated metainformation in the form of entity-headers, which if present SHOULD be associated with the requested variant.

If the client is a user agent, it SHOULD NOT change its document view from that which caused the request to be sent. This response is primarily intended to allow input for actions to take place without causing a change to the user agent's active document view, although any new or updated metainformation SHOULD be applied to the document currently in the user agent's active view.

The 204 response MUST NOT include a message-body, and thus is always terminated by the first empty line after the header fields.

10.2.6 205 Reset Content

The server has fulfilled the request and the user agent SHOULD reset the document view which caused the request to be sent. This response is primarily intended to allow input for actions to take place via user input, followed by a clearing of the form in which the input is given so that the user can easily initiate another input action. The response MUST NOT include an entity.

10.2.7 206 Partial Content

The server has fulfilled the partial GET request for the resource. The request MUST have included a Range header field (section 14.35) indicating the desired range, and MAY have included an If-Range header field (section 14.27) to make the request conditional.

The response MUST include the following header fields:

- Either a Content-Range header field (section 14.16) indicating the range included with this response, or a multipart/byteranges Content-Type including Content-Range fields for each part. If a Content-Length header field is present in the response, its value MUST match the actual number of OCTETs transmitted in the message-body.

- Date

- ETag and/or Content-Location, if the header would have been sent in a 200 response to the same request

IPR2025-01572
Apple EX1046 Page 26

- Expires, Cache-Control, and/or Vary, if the field-value might differ from that sent in any previous response for the same variant

If the 206 response is the result of an If-Range request that used a strong cache validator (see section 13.3.3), the response SHOULD NOT include other entity-headers. If the response is the result of an If-Range request that used a weak validator, the response MUST NOT include other entity-headers; this prevents inconsistencies between cached entity-bodies and updated headers. Otherwise, the response MUST include all of the entity-headers that would have been returned with a 200 (OK) response to the same request.

A cache MUST NOT combine a 206 response with other previously cached content if the ETag or Last-Modified headers do not match exactly, see 13.5.4.

A cache that does not support the Range and Content-Range headers MUST NOT cache 206 (Partial) responses.

10.3 Redirection 3xx

This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request. The action required MAY be carried out by the user agent without interaction with the user if and only if the method used in the second request is GET or HEAD. A client SHOULD detect infinite redirection loops, since such loops generate network traffic for each redirection.

Note: previous versions of this specification recommended a maximum of five redirections. Content developers should be aware that there might be clients that implement such a fixed limitation.

10.3.1 300 Multiple Choices

The requested resource corresponds to any one of a set of representations, each with its own specific location, and agent-driven negotiation information (section 12) is being provided so that the user (or user agent) can select a preferred representation and redirect its request to that location.

Unless it was a HEAD request, the response SHOULD include an entity containing a list of resource characteristics and location(s) from which the user or user agent can choose the one most appropriate. The entity format is specified by the media type given in the Content-Type header field. Depending upon the format and the capabilities of

IPR2025-01572
Apple EX1046 Page 27

the user agent, selection of the most appropriate choice MAY be performed automatically. However, this specification does not define any standard for such automatic selection.

If the server has a preferred choice of representation, it SHOULD include the specific URI for that representation in the Location field; user agents MAY use the Location field value for automatic redirection. This response is cacheable unless indicated otherwise.

10.3.2 301 Moved Permanently

The requested resource has been assigned a new permanent URI and any future references to this resource SHOULD use one of the returned URIs. Clients with link editing capabilities ought to automatically re-link references to the Request-URI to one or more of the new references returned by the server, where possible. This response is cacheable unless indicated otherwise.

The new permanent URI SHOULD be given by the Location field in the response. Unless the request method was HEAD, the entity of the response SHOULD contain a short hypertext note with a hyperlink to the new URI(s).

If the 301 status code is received in response to a request other than GET or HEAD, the user agent MUST NOT automatically redirect the request unless it can be confirmed by the user, since this might change the conditions under which the request was issued.

> Note: When automatically redirecting a POST request after receiving a 301 status code, some existing HTTP/1.0 user agents will erroneously change it into a GET request.

10.3.3 302 Found

The requested resource resides temporarily under a different URI. Since the redirection might be altered on occasion, the client SHOULD continue to use the Request-URI for future requests. This response is only cacheable if indicated by a Cache-Control or Expires header field.

The temporary URI SHOULD be given by the Location field in the response. Unless the request method was HEAD, the entity of the response SHOULD contain a short hypertext note with a hyperlink to the new URI(s).

IPR2025-01572
Apple EX1046 Page 28

If the 302 status code is received in response to a request other than GET or HEAD, the user agent MUST NOT automatically redirect the request unless it can be confirmed by the user, since this might change the conditions under which the request was issued.

Note: RFC 1945 and RFC 2068 specify that the client is not allowed to change the method on the redirected request. However, most existing user agent implementations treat 302 as if it were a 303 response, performing a GET on the Location field-value regardless of the original request method. The status codes 303 and 307 have been added for servers that wish to make unambiguously clear which kind of reaction is expected of the client.

10.3.4 303 See Other

The response to the request can be found under a different URI and SHOULD be retrieved using a GET method on that resource. This method exists primarily to allow the output of a POST-activated script to redirect the user agent to a selected resource. The new URI is not a substitute reference for the originally requested resource. The 303 response MUST NOT be cached, but the response to the second (redirected) request might be cacheable.

The different URI SHOULD be given by the Location field in the response. Unless the request method was HEAD, the entity of the response SHOULD contain a short hypertext note with a hyperlink to the new URI(s).

Note: Many pre-HTTP/1.1 user agents do not understand the 303 status. When interoperability with such clients is a concern, the 302 status code may be used instead, since most user agents react to a 302 response as described here for 303.

10.3.5 304 Not Modified

If the client has performed a conditional GET request and access is allowed, but the document has not been modified, the server SHOULD respond with this status code. The 304 response MUST NOT contain a message-body, and thus is always terminated by the first empty line after the header fields.

The response MUST include the following header fields:

- Date, unless its omission is required by section 14.18.1

IPR2025-01572
Apple EX1046 Page 29

If a clockless origin server obeys these rules, and proxies and clients add their own Date to any response received without one (as already specified by [RFC 2068], section 14.19), caches will operate correctly.

- ETag and/or Content-Location, if the header would have been sent in a 200 response to the same request

- Expires, Cache-Control, and/or Vary, if the field-value might differ from that sent in any previous response for the same variant

If the conditional GET used a strong cache validator (see section 13.3.3), the response SHOULD NOT include other entity-headers. Otherwise (i.e., the conditional GET used a weak validator), the response MUST NOT include other entity-headers; this prevents inconsistencies between cached entity-bodies and updated headers.

If a 304 response indicates an entity not currently cached, then the cache MUST disregard the response and repeat the request without the conditional.

If a cache uses a received 304 response to update a cache entry, the cache MUST update the entry to reflect any new field values given in the response.

10.3.6 305 Use Proxy

The requested resource MUST be accessed through the proxy given by the Location field. The Location field gives the URI of the proxy. The recipient is expected to repeat this single request via the proxy. 305 responses MUST only be generated by origin servers.

Note: RFC 2068 was not clear that 305 was intended to redirect a single request, and to be generated by origin servers only. Not observing these limitations has significant security consequences.

10.3.7 306 (Unused)

The 306 status code was used in a previous version of the specification, is no longer used, and the code is reserved.

IPR2025-01572
Apple EX1046 Page 30

10.3.8 307 Temporary Redirect

The requested resource resides temporarily under a different URI. Since the redirection MAY be altered on occasion, the client SHOULD continue to use the Request-URI for future requests. This response is only cacheable if indicated by a Cache-Control or Expires header field.

The temporary URI SHOULD be given by the Location field in the response. Unless the request method was HEAD, the entity of the response SHOULD contain a short hypertext note with a hyperlink to the new URI(s) , since many pre-HTTP/1.1 user agents do not understand the 307 status. Therefore, the note SHOULD contain the information necessary for a user to repeat the original request on the new URI.

If the 307 status code is received in response to a request other than GET or HEAD, the user agent MUST NOT automatically redirect the request unless it can be confirmed by the user, since this might change the conditions under which the request was issued.

10.4 Client Error 4xx

The 4xx class of status code is intended for cases in which the client seems to have erred. Except when responding to a HEAD request, the server SHOULD include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition. These status codes are applicable to any request method. User agents SHOULD display any included entity to the user.

If the client is sending data, a server implementation using TCP SHOULD be careful to ensure that the client acknowledges receipt of the packet(s) containing the response, before the server closes the input connection. If the client continues sending data to the server after the close, the server's TCP stack will send a reset packet to the client, which may erase the client's unacknowledged input buffers before they can be read and interpreted by the HTTP application.

10.4.1 400 Bad Request

The request could not be understood by the server due to malformed syntax. The client SHOULD NOT repeat the request without modifications.

IPR2025-01572
Apple EX1046 Page 31

10.4.2 401 Unauthorized

The request requires user authentication. The response MUST include a WWW-Authenticate header field (section 14.47) containing a challenge applicable to the requested resource. The client MAY repeat the request with a suitable Authorization header field (section 14.8). If the request already included Authorization credentials, then the 401 response indicates that authorization has been refused for those credentials. If the 401 response contains the same challenge as the prior response, and the user agent has already attempted authentication at least once, then the user SHOULD be presented the entity that was given in the response, since that entity might include relevant diagnostic information. HTTP access authentication is explained in "HTTP Authentication: Basic and Digest Access Authentication" [43].

10.4.3 402 Payment Required

This code is reserved for future use.

10.4.4 403 Forbidden

The server understood the request, but is refusing to fulfill it. Authorization will not help and the request SHOULD NOT be repeated. If the request method was not HEAD and the server wishes to make public why the request has not been fulfilled, it SHOULD describe the reason for the refusal in the entity. If the server does not wish to make this information available to the client, the status code 404 (Not Found) can be used instead.

10.4.5 404 Not Found

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent. The 410 (Gone) status code SHOULD be used if the server knows, through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding address. This status code is commonly used when the server does not wish to reveal exactly why the request has been refused, or when no other response is applicable.

10.4.6 405 Method Not Allowed

The method specified in the Request-Line is not allowed for the resource identified by the Request-URI. The response MUST include an Allow header containing a list of valid methods for the requested resource.

IPR2025-01572
Apple EX1046 Page 32

A proxy MUST NOT modify the Allow header field even if it does not understand all the methods specified, since the user agent might have other means of communicating with the origin server.

14.8 Authorization

A user agent that wishes to authenticate itself with a server--usually, but not necessarily, after receiving a 401 response--does so by including an Authorization request-header field with the request. The Authorization field value consists of credentials containing the authentication information of the user agent for the realm of the resource being requested.

```
    Authorization  = "Authorization" ":" credentials
```

HTTP access authentication is described in "HTTP Authentication: Basic and Digest Access Authentication" [43]. If a request is authenticated and a realm specified, the same credentials SHOULD be valid for all other requests within this realm (assuming that the authentication scheme itself does not require otherwise, such as credentials that vary according to a challenge value or using synchronized clocks).

When a shared cache (see section 13.7) receives a request containing an Authorization field, it MUST NOT return the corresponding response as a reply to any other request, unless one of the following specific exceptions holds:

1. If the response includes the "s-maxage" cache-control directive, the cache MAY use that response in replying to a subsequent request. But (if the specified maximum age has passed) a proxy cache MUST first revalidate it with the origin server, using the request-headers from the new request to allow the origin server to authenticate the new request. (This is the defined behavior for s-maxage.) If the response includes "s-maxage=0", the proxy MUST always revalidate it before re-using it.

2. If the response includes the "must-revalidate" cache-control directive, the cache MAY use that response in replying to a subsequent request. But if the response is stale, all caches MUST first revalidate it with the origin server, using the request-headers from the new request to allow the origin server to authenticate the new request.

3. If the response includes the "public" cache-control directive, it MAY be returned in reply to any subsequent request.

IPR2025-01572
Apple EX1046 Page 33

The Apache Group, Anselm Baird-Smith, author of Jigsaw, and Henrik Frystyk implemented RFC 2068 early, and we wish to thank them for the discovery of many of the problems that this document attempts to rectify.

17 References

[1] Alvestrand, H., "Tags for the Identification of Languages", RFC 1766, March 1995.

[2] Anklesaria, F., McCahill, M., Lindner, P., Johnson, D., Torrey, D. and B. Alberti, "The Internet Gopher Protocol (a distributed document search and retrieval protocol)", RFC 1436, March 1993.

[3] Berners-Lee, T., "Universal Resource Identifiers in WWW", RFC 1630, June 1994.

[4] Berners-Lee, T., Masinter, L. and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[5] Berners-Lee, T. and D. Connolly, "Hypertext Markup Language - 2.0", RFC 1866, November 1995.

[6] Berners-Lee, T., Fielding, R. and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996.

[7] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[8] Braden, R., "Requirements for Internet Hosts -- Communication Layers", STD 3, RFC 1123, October 1989.

[9] Crocker, D., "Standard for The Format of ARPA Internet Text Messages", STD 11, RFC 822, August 1982.

[10] Davis, F., Kahle, B., Morris, H., Salem, J., Shen, T., Wang, R., Sui, J., and M. Grinbaum, "WAIS Interface Protocol Prototype Functional Specification," (v1.5), Thinking Machines Corporation, April 1990.

[11] Fielding, R., "Relative Uniform Resource Locators", RFC 1808, June 1995.

[12] Horton, M. and R. Adams, "Standard for Interchange of USENET Messages", RFC 1036, December 1987.

IPR2025-01572
Apple EX1046 Page 34

[13] Kantor, B. and P. Lapsley, "Network News Transfer Protocol", RFC 977, February 1986.

[14] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

[15] Nebel, E. and L. Masinter, "Form-based File Upload in HTML", RFC 1867, November 1995.

[16] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, August 1982.

[17] Postel, J., "Media Type Registration Procedure", RFC 1590, November 1996.

[18] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, October 1985.

[19] Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994.

[20] Sollins, K. and L. Masinter, "Functional Requirements for Uniform Resource Names", RFC 1737, December 1994.

[21] US-ASCII. Coded Character Set - 7-Bit American Standard Code for Information Interchange. Standard ANSI X3.4-1986, ANSI, 1986.

[22] ISO-8859. International Standard -- Information Processing -- 8-bit Single-Byte Coded Graphic Character Sets --
Part 1: Latin alphabet No. 1, ISO-8859-1:1987.
Part 2: Latin alphabet No. 2, ISO-8859-2, 1987.
Part 3: Latin alphabet No. 3, ISO-8859-3, 1988.
Part 4: Latin alphabet No. 4, ISO-8859-4, 1988.
Part 5: Latin/Cyrillic alphabet, ISO-8859-5, 1988.
Part 6: Latin/Arabic alphabet, ISO-8859-6, 1987.
Part 7: Latin/Greek alphabet, ISO-8859-7, 1987.
Part 8: Latin/Hebrew alphabet, ISO-8859-8, 1988.
Part 9: Latin alphabet No. 5, ISO-8859-9, 1990.

[23] Meyers, J. and M. Rose, "The Content-MD5 Header Field", RFC 1864, October 1995.

[24] Carpenter, B. and Y. Rekhter, "Renumbering Needs Work", RFC 1900, February 1996.

[25] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, May 1996.

IPR2025-01572
Apple EX1046 Page 35

[26] Venkata N. Padmanabhan, and Jeffrey C. Mogul. "Improving HTTP Latency", Computer Networks and ISDN Systems, v. 28, pp. 25-35, Dec. 1995. Slightly revised version of paper in Proc. 2nd International WWW Conference '94: Mosaic and the Web, Oct. 1994, which is available at http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/mogul/HTTPLatency.html.

[27] Joe Touch, John Heidemann, and Katia Obraczka. "Analysis of HTTP Performance", <URL: http://www.isi.edu/touch/pubs/http-perf96/>, ISI Research Report ISI/RR-98-463, (original report dated Aug. 1996), USC/Information Sciences Institute, August 1998.

[28] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, March 1992.

[29] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.

[30] S. Spero, "Analysis of HTTP Performance Problems," http://sunsite.unc.edu/mdma-release/http-prob.html.

[31] Deutsch, P. and J. Gailly, "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996.

[32] Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E. and L. Stewart, "An Extension to HTTP: Digest Access Authentication", RFC 2069, January 1997.

[33] Fielding, R., Gettys, J., Mogul, J., Frystyk, H. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

[34] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[35] Troost, R. and Dorner, S., "Communicating Presentation Information in Internet Messages: The Content-Disposition Header", RFC 1806, June 1995.

[36] Mogul, J., Fielding, R., Gettys, J. and H. Frystyk, "Use and Interpretation of HTTP Version Numbers", RFC 2145, May 1997. [jg639]

[37] Palme, J., "Common Internet Message Headers", RFC 2076, February 1997. [jg640]

IPR2025-01572
Apple EX1046 Page 36

[38] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO-10646", RFC 2279, January 1998. [jg641]

[39] Nielsen, H.F., Gettys, J., Baird-Smith, A., Prud'hommeaux, E., Lie, H., and C. Lilley. "Network Performance Effects of HTTP/1.1, CSS1, and PNG," Proceedings of ACM SIGCOMM '97, Cannes France, September 1997.[jg642]

[40] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996. [jg643]

[41] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998. [jg644]

[42] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax and Semantics", RFC 2396, August 1998. [jg645]

[43] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., Sink, E. and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999. [jg646]

[44] Luotonen, A., "Tunneling TCP based protocols through Web proxy servers," Work in Progress. [jg647]

[45] Palme, J. and A. Hopmann, "MIME E-mail Encapsulation of Aggregate Documents, such as HTML (MHTML)", RFC 2110, March 1997.

[46] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[47] Masinter, L., "Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)", RFC 2324, 1 April 1998.

[48] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, November 1996.

[49] Troost, R., Dorner, S. and K. Moore, "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, August 1997.

IPR2025-01572
Apple EX1046 Page 37

18 Authors' Addresses

Roy T. Fielding
Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425, USA

Fax: +1 (949) 824-1715
EMail: fielding@ics.uci.edu

James Gettys
World Wide Web Consortium
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, USA

Fax: +1 (617) 258 8682
EMail: jg@w3.org

Jeffrey C. Mogul
Western Research Laboratory
Compaq Computer Corporation
250 University Avenue
Palo Alto, California, 94305, USA

EMail: mogul@wrl.dec.com

Henrik Frystyk Nielsen
World Wide Web Consortium
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, USA

Fax: +1 (617) 258 8682
EMail: frystyk@w3.org

Larry Masinter
Xerox Corporation
3333 Coyote Hill Road
Palo Alto, CA 94034, USA

EMail: masinter@parc.xerox.com

IPR2025-01572
Apple EX1046 Page 38

Paul J. Leach
Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052, USA

EMail: paulle@microsoft.com

Tim Berners-Lee
Director, World Wide Web Consortium
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, USA

Fax: +1 (617) 258 8682
EMail: timbl@w3.org

IPR2025-01572
Apple EX1046 Page 39

19 Appendices

19.1 Internet Media Type message/http and application/http

In addition to defining the HTTP/1.1 protocol, this document serves as the specification for the Internet media type "message/http" and "application/http". The message/http type can be used to enclose a single HTTP request or response message, provided that it obeys the MIME restrictions for all "message" types regarding line length and encodings. The application/http type can be used to enclose a pipeline of one or more HTTP request or response messages (not intermixed). The following is to be registered with IANA [17].

```
    Media Type name:         message
    Media subtype name:      http
    Required parameters:     none
    Optional parameters:     version, msgtype
     version: The HTTP-Version number of the enclosed message
              (e.g., "1.1"). If not present, the version can be
              determined from the first line of the body.
     msgtype: The message type -- "request" or "response". If not
              present, the type can be determined from the first
              line of the body.
    Encoding considerations: only "7bit", "8bit", or "binary" are
                             permitted
    Security considerations: none

    Media Type name:         application
    Media subtype name:      http
    Required parameters:     none
    Optional parameters:     version, msgtype
     version: The HTTP-Version number of the enclosed messages
              (e.g., "1.1"). If not present, the version can be
              determined from the first line of the body.
     msgtype: The message type -- "request" or "response". If not
              present, the type can be determined from the first
              line of the body.
    Encoding considerations: HTTP messages enclosed by this type
              are in "binary" format; use of an appropriate
              Content-Transfer-Encoding is required when
              transmitted via E-mail.
    Security considerations: none
```


IPR2025-01572
Apple EX1046 Page 40

19.2 Internet Media Type multipart/byteranges

When an HTTP 206 (Partial Content) response message includes the content of multiple ranges (a response to a request for multiple non-overlapping ranges), these are transmitted as a multipart message-body. The media type for this purpose is called "multipart/byteranges".

The multipart/byteranges media type includes two or more parts, each with its own Content-Type and Content-Range fields. The required boundary parameter specifies the boundary string used to separate each body-part.

```
    Media Type name:         multipart
    Media subtype name:      byteranges
    Required parameters:     boundary
    Optional parameters:     none
    Encoding considerations: only "7bit", "8bit", or "binary" are
                             permitted
    Security considerations: none
```

For example:

```
HTTP/1.1 206 Partial Content
Date: Wed, 15 Nov 1995 06:25:24 GMT
Last-Modified: Wed, 15 Nov 1995 04:58:08 GMT
Content-type: multipart/byteranges; boundary=THIS_STRING_SEPARATES

--THIS_STRING_SEPARATES
Content-type: application/pdf
Content-range: bytes 500-999/8000

...the first range...
--THIS_STRING_SEPARATES
Content-type: application/pdf
Content-range: bytes 7000-7999/8000

...the second range
--THIS_STRING_SEPARATES--
```

Notes:

1) Additional CRLFs may precede the first boundary string in the entity.

IPR2025-01572
Apple EX1046 Page 41

2) Although RFC 2046 [40] permits the boundary string to be quoted, some existing implementations handle a quoted boundary string incorrectly.

3) A number of browsers and servers were coded to an early draft of the byteranges specification to use a media type of multipart/x-byteranges, which is almost, but not quite compatible with the version documented in HTTP/1.1.

19.3 Tolerant Applications

Although this document specifies the requirements for the generation of HTTP/1.1 messages, not all applications will be correct in their implementation. We therefore recommend that operational applications be tolerant of deviations whenever those deviations can be interpreted unambiguously.

Clients SHOULD be tolerant in parsing the Status-Line and servers tolerant when parsing the Request-Line. In particular, they SHOULD accept any amount of SP or HT characters between fields, even though only a single SP is required.

The line terminator for message-header fields is the sequence CRLF. However, we recommend that applications, when parsing such headers, recognize a single LF as a line terminator and ignore the leading CR.

The character set of an entity-body SHOULD be labeled as the lowest common denominator of the character codes used within that body, with the exception that not labeling the entity is preferred over labeling the entity with the labels US-ASCII or ISO-8859-1. See section 3.7.1 and 3.4.1.

Additional rules for requirements on parsing and encoding of dates and other potential problems with date encodings include:

- HTTP/1.1 clients and caches SHOULD assume that an RFC-850 date which appears to be more than 50 years in the future is in fact in the past (this helps solve the "year 2000" problem).

- An HTTP/1.1 implementation MAY internally represent a parsed Expires date as earlier than the proper value, but MUST NOT internally represent a parsed Expires date as later than the proper value.

- All expiration-related calculations MUST be done in GMT. The local time zone MUST NOT influence the calculation or comparison of an age or expiration time.

IPR2025-01572
Apple EX1046 Page 42

- If an HTTP header incorrectly carries a date value with a time zone other than GMT, it MUST be converted into GMT using the most conservative possible conversion.

19.4 Differences Between HTTP Entities and RFC 2045 Entities

HTTP/1.1 uses many of the constructs defined for Internet Mail (RFC 822 [9]) and the Multipurpose Internet Mail Extensions (MIME [7]) to allow entities to be transmitted in an open variety of representations and with extensible mechanisms. However, RFC 2045 discusses mail, and HTTP has a few features that are different from those described in RFC 2045. These differences were carefully chosen to optimize performance over binary connections, to allow greater freedom in the use of new media types, to make date comparisons easier, and to acknowledge the practice of some early HTTP servers and clients.

This appendix describes specific areas where HTTP differs from RFC 2045. Proxies and gateways to strict MIME environments SHOULD be aware of these differences and provide the appropriate conversions where necessary. Proxies and gateways from MIME environments to HTTP also need to be aware of the differences because some conversions might be required.

19.4.1 MIME-Version

HTTP is not a MIME-compliant protocol. However, HTTP/1.1 messages MAY include a single MIME-Version general-header field to indicate what version of the MIME protocol was used to construct the message. Use of the MIME-Version header field indicates that the message is in full compliance with the MIME protocol (as defined in RFC 2045[7]). Proxies/gateways are responsible for ensuring full compliance (where possible) when exporting HTTP messages to strict MIME environments.

```
    MIME-Version   = "MIME-Version" ":" 1*DIGIT "." 1*DIGIT
```

MIME version "1.0" is the default for use in HTTP/1.1. However, HTTP/1.1 message parsing and semantics are defined by this document and not the MIME specification.

19.4.2 Conversion to Canonical Form

RFC 2045 [7] requires that an Internet mail entity be converted to canonical form prior to being transferred, as described in section 4 of RFC 2049 [48]. Section 3.7.1 of this document describes the forms allowed for subtypes of the "text" media type when transmitted over HTTP. RFC 2046 requires that content with a type of "text" represent line breaks as CRLF and forbids the use of CR or LF outside of line

IPR2025-01572
Apple EX1046 Page 43

break sequences. HTTP allows CRLF, bare CR, and bare LF to indicate a line break within text content when a message is transmitted over HTTP.

Where it is possible, a proxy or gateway from HTTP to a strict MIME environment SHOULD translate all line breaks within the text media types described in section 3.7.1 of this document to the RFC 2049 canonical form of CRLF. Note, however, that this might be complicated by the presence of a Content-Encoding and by the fact that HTTP allows the use of some character sets which do not use octets 13 and 10 to represent CR and LF, as is the case for some multi-byte character sets.

Implementors should note that conversion will break any cryptographic checksums applied to the original content unless the original content is already in canonical form. Therefore, the canonical form is recommended for any content that uses such checksums in HTTP.

19.4.3 Conversion of Date Formats

HTTP/1.1 uses a restricted set of date formats (section 3.3.1) to simplify the process of date comparison. Proxies and gateways from other protocols SHOULD ensure that any Date header field present in a message conforms to one of the HTTP/1.1 formats and rewrite the date if necessary.

19.4.4 Introduction of Content-Encoding

RFC 2045 does not include any concept equivalent to HTTP/1.1's Content-Encoding header field. Since this acts as a modifier on the media type, proxies and gateways from HTTP to MIME-compliant protocols MUST either change the value of the Content-Type header field or decode the entity-body before forwarding the message. (Some experimental applications of Content-Type for Internet mail have used a media-type parameter of ";conversions=<content-coding>" to perform a function equivalent to Content-Encoding. However, this parameter is not part of RFC 2045.)

19.4.5 No Content-Transfer-Encoding

HTTP does not use the Content-Transfer-Encoding (CTE) field of RFC 2045. Proxies and gateways from MIME-compliant protocols to HTTP MUST remove any non-identity CTE ("quoted-printable" or "base64") encoding prior to delivering the response message to an HTTP client.

Proxies and gateways from HTTP to MIME-compliant protocols are responsible for ensuring that the message is in the correct format and encoding for safe transport on that protocol, where "safe

IPR2025-01572
Apple EX1046 Page 44

transport" is defined by the limitations of the protocol being used. Such a proxy or gateway SHOULD label the data with an appropriate Content-Transfer-Encoding if doing so will improve the likelihood of safe transport over the destination protocol.

19.4.6 Introduction of Transfer-Encoding

HTTP/1.1 introduces the Transfer-Encoding header field (section 14.41). Proxies/gateways MUST remove any transfer-coding prior to forwarding a message via a MIME-compliant protocol.

A process for decoding the "chunked" transfer-coding (section 3.6) can be represented in pseudo-code as:

```
length := 0
read chunk-size, chunk-extension (if any) and CRLF
while (chunk-size > 0) {
   read chunk-data and CRLF
   append chunk-data to entity-body
   length := length + chunk-size
   read chunk-size and CRLF
}
read entity-header
while (entity-header not empty) {
   append entity-header to existing header fields
   read entity-header
}
Content-Length := length
Remove "chunked" from Transfer-Encoding
```

19.4.7 MHTML and Line Length Limitations

HTTP implementations which share code with MHTML [45] implementations need to be aware of MIME line length limitations. Since HTTP does not have this limitation, HTTP does not fold long lines. MHTML messages being transported by HTTP follow all conventions of MHTML, including line length limitations and folding, canonicalization, etc., since HTTP transports all message-bodies as payload (see section 3.7.2) and does not interpret the content or any MIME header lines that might be contained therein.

19.5 Additional Features

RFC 1945 and RFC 2068 document protocol elements used by some existing HTTP implementations, but not consistently and correctly across most HTTP/1.1 applications. Implementors are advised to be aware of these features, but cannot rely upon their presence in, or interoperability with, other HTTP/1.1 applications. Some of these

IPR2025-01572
Apple EX1046 Page 45

describe proposed experimental features, and some describe features that experimental deployment found lacking that are now addressed in the base HTTP/1.1 specification.

A number of other headers, such as Content-Disposition and Title, from SMTP and MIME are also often implemented (see RFC 2076 [37]).

19.5.1 Content-Disposition

The Content-Disposition response-header field has been proposed as a means for the origin server to suggest a default filename if the user requests that the content is saved to a file. This usage is derived from the definition of Content-Disposition in RFC 1806 [35].

```
content-disposition = "Content-Disposition" ":"
                      disposition-type *( ";" disposition-parm )
disposition-type = "attachment" | disp-extension-token
disposition-parm = filename-parm | disp-extension-parm
filename-parm = "filename" "=" quoted-string
disp-extension-token = token
disp-extension-parm = token "=" ( token | quoted-string )
```

An example is

```
Content-Disposition: attachment; filename="fname.ext"
```

The receiving user agent SHOULD NOT respect any directory path information present in the filename-parm parameter, which is the only parameter believed to apply to HTTP implementations at this time. The filename SHOULD be treated as a terminal component only.

If this header is used in a response with the application/octet-stream content-type, the implied suggestion is that the user agent should not display the response, but directly enter a `save response as...' dialog.

See section 15.5 for Content-Disposition security issues.

19.6 Compatibility with Previous Versions

It is beyond the scope of a protocol specification to mandate compliance with previous versions. HTTP/1.1 was deliberately designed, however, to make supporting previous versions easy. It is worth noting that, at the time of composing this specification (1996), we would expect commercial HTTP/1.1 servers to:

- recognize the format of the Request-Line for HTTP/0.9, 1.0, and 1.1 requests;

IPR2025-01572
Apple EX1046 Page 46

- understand any valid request in the format of HTTP/0.9, 1.0, or 1.1;

- respond appropriately with a message in the same major version used by the client.

And we would expect HTTP/1.1 clients to:

- recognize the format of the Status-Line for HTTP/1.0 and 1.1 responses;

- understand any valid response in the format of HTTP/0.9, 1.0, or 1.1.

For most implementations of HTTP/1.0, each connection is established by the client prior to the request and closed by the server after sending the response. Some implementations implement the Keep-Alive version of persistent connections described in section 19.7.1 of RFC 2068 [33].

19.6.1 Changes from HTTP/1.0

This section summarizes major differences between versions HTTP/1.0 and HTTP/1.1.

19.6.1.1 Changes to Simplify Multi-homed Web Servers and Conserve IP Addresses

The requirements that clients and servers support the Host request-header, report an error if the Host request-header (section 14.23) is missing from an HTTP/1.1 request, and accept absolute URIs (section 5.1.2) are among the most important changes defined by this specification.

Older HTTP/1.0 clients assumed a one-to-one relationship of IP addresses and servers; there was no other established mechanism for distinguishing the intended server of a request than the IP address to which that request was directed. The changes outlined above will allow the Internet, once older HTTP clients are no longer common, to support multiple Web sites from a single IP address, greatly simplifying large operational Web servers, where allocation of many IP addresses to a single host has created serious problems. The Internet will also be able to recover the IP addresses that have been allocated for the sole purpose of allowing special-purpose domain names to be used in root-level HTTP URLs. Given the rate of growth of the Web, and the number of servers already deployed, it is extremely

IPR2025-01572
Apple EX1046 Page 47

important that all implementations of HTTP (including updates to existing HTTP/1.0 applications) correctly implement these requirements:

- Both clients and servers MUST support the Host request-header.
- A client that sends an HTTP/1.1 request MUST send a Host header.
- Servers MUST report a 400 (Bad Request) error if an HTTP/1.1 request does not include a Host request-header.
- Servers MUST accept absolute URIs.

19.6.2 Compatibility with HTTP/1.0 Persistent Connections

Some clients and servers might wish to be compatible with some previous implementations of persistent connections in HTTP/1.0 clients and servers. Persistent connections in HTTP/1.0 are explicitly negotiated as they are not the default behavior. HTTP/1.0 experimental implementations of persistent connections are faulty, and the new facilities in HTTP/1.1 are designed to rectify these problems. The problem was that some existing 1.0 clients may be sending Keep-Alive to a proxy server that doesn't understand Connection, which would then erroneously forward it to the next inbound server, which would establish the Keep-Alive connection and result in a hung HTTP/1.0 proxy waiting for the close on the response. The result is that HTTP/1.0 clients must be prevented from using Keep-Alive when talking to proxies.

However, talking to proxies is the most important use of persistent connections, so that prohibition is clearly unacceptable. Therefore, we need some other mechanism for indicating a persistent connection is desired, which is safe to use even when talking to an old proxy that ignores Connection. Persistent connections are the default for HTTP/1.1 messages; we introduce a new keyword (Connection: close) for declaring non-persistence. See section 14.10.

The original HTTP/1.0 form of persistent connections (the Connection: Keep-Alive and Keep-Alive header) is documented in RFC 2068. [33]

19.6.3 Changes from RFC 2068

This specification has been carefully audited to correct and disambiguate key word usage; RFC 2068 had many problems in respect to the conventions laid out in RFC 2119 [34].

Clarified which error code should be used for inbound server failures (e.g. DNS failures). (Section 10.5.5).

IPR2025-01572
Apple EX1046 Page 48

CREATE had a race that required an Etag be sent when a resource is first created. (Section 10.2.2).

Content-Base was deleted from the specification: it was not implemented widely, and there is no simple, safe way to introduce it without a robust extension mechanism. In addition, it is used in a similar, but not identical fashion in MHTML [45].

Transfer-coding and message lengths all interact in ways that required fixing exactly when chunked encoding is used (to allow for transfer encoding that may not be self delimiting); it was important to straighten out exactly how message lengths are computed. (Sections 3.6, 4.4, 7.2.2, 13.5.2, 14.13, 14.16)

A content-coding of "identity" was introduced, to solve problems discovered in caching. (section 3.5)

Quality Values of zero should indicate that "I don't want something" to allow clients to refuse a representation. (Section 3.9)

The use and interpretation of HTTP version numbers has been clarified by RFC 2145. Require proxies to upgrade requests to highest protocol version they support to deal with problems discovered in HTTP/1.0 implementations (Section 3.1)

Charset wildcarding is introduced to avoid explosion of character set names in accept headers. (Section 14.2)

A case was missed in the Cache-Control model of HTTP/1.1; s-maxage was introduced to add this missing case. (Sections 13.4, 14.8, 14.9, 14.9.3)

The Cache-Control: max-age directive was not properly defined for responses. (Section 14.9.3)

There are situations where a server (especially a proxy) does not know the full length of a response but is capable of serving a byterange request. We therefore need a mechanism to allow byteranges with a content-range not indicating the full length of the message. (Section 14.16)

Range request responses would become very verbose if all meta-data were always returned; by allowing the server to only send needed headers in a 206 response, this problem can be avoided. (Section 10.2.7, 13.5.3, and 14.27)

IPR2025-01572
Apple EX1046 Page 49

Fix problem with unsatisfiable range requests; there are two cases: syntactic problems, and range doesn't exist in the document. The 416 status code was needed to resolve this ambiguity needed to indicate an error for a byte range request that falls outside of the actual contents of a document. (Section 10.4.17, 14.16)

Rewrite of message transmission requirements to make it much harder for implementors to get it wrong, as the consequences of errors here can have significant impact on the Internet, and to deal with the following problems:

1. Changing "HTTP/1.1 or later" to "HTTP/1.1", in contexts where this was incorrectly placing a requirement on the behavior of an implementation of a future version of HTTP/1.x

2. Made it clear that user-agents should retry requests, not "clients" in general.

3. Converted requirements for clients to ignore unexpected 100 (Continue) responses, and for proxies to forward 100 responses, into a general requirement for 1xx responses.

4. Modified some TCP-specific language, to make it clearer that non-TCP transports are possible for HTTP.

5. Require that the origin server MUST NOT wait for the request body before it sends a required 100 (Continue) response.

6. Allow, rather than require, a server to omit 100 (Continue) if it has already seen some of the request body.

7. Allow servers to defend against denial-of-service attacks and broken clients.

This change adds the Expect header and 417 status code. The message transmission requirements fixes are in sections 8.2, 10.4.18, 8.1.2.2, 13.11, and 14.20.

Proxies should be able to add Content-Length when appropriate. (Section 13.5.2)

Clean up confusion between 403 and 404 responses. (Section 10.4.4, 10.4.5, and 10.4.11)

Warnings could be cached incorrectly, or not updated appropriately. (Section 13.1.2, 13.2.4, 13.5.2, 13.5.3, 14.9.3, and 14.46) Warning also needed to be a general header, as PUT or other methods may have need for it in requests.

IPR2025-01572
Apple EX1046 Page 50

Transfer-coding had significant problems, particularly with interactions with chunked encoding. The solution is that transfer-codings become as full fledged as content-codings. This involves adding an IANA registry for transfer-codings (separate from content codings), a new header field (TE) and enabling trailer headers in the future. Transfer encoding is a major performance benefit, so it was worth fixing [39]. TE also solves another, obscure, downward interoperability problem that could have occurred due to interactions between authentication trailers, chunked encoding and HTTP/1.0 clients.(Section 3.6, 3.6.1, and 14.39)

The PATCH, LINK, UNLINK methods were defined but not commonly implemented in previous versions of this specification. See RFC 2068 [33].

The Alternates, Content-Version, Derived-From, Link, URI, Public and Content-Base header fields were defined in previous versions of this specification, but not commonly implemented. See RFC 2068 [33].

20 Index

Please see the PostScript version of this RFC for the INDEX.

IPR2025-01572
Apple EX1046 Page 51

21. Full Copyright Statement

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

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

IPR2025-01572
Apple EX1046 Page 52