List of HTTP status codes

From Wikipedia, the free encyclopedia

This is a list of **Hypertext Transfer Protocol (HTTP) response status codes**. It includes codes from IETF internet standards, other IETF RFCs, other specifications, and some additional commonly used codes. The first digit of the status code specifies one of five classes of response; an HTTP client must recognise these five classes at a minimum. The phrases used are the standard wordings, but any human-readable alternative can be provided. Unless otherwise stated, the status code is part of the HTTP/1.1 standard (RFC 7231).[1]

The Internet Assigned Numbers Authority (IANA) maintains the official registry of HTTP status codes.[2]

Microsoft IIS sometimes uses additional decimal sub-codes to provide more specific information,[3] but not all of those are here.

Contents

1xx Informational

Request received, continuing process.

This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line. Since HTTP/1.0 did not define any 1xx status codes, servers *must not*[note 1] send a 1xx response to an HTTP/1.0 client except under experimental conditions.[4]

100 Continue
: The server has received the request headers and the client should proceed to send the request body (in the case of a request for which a body needs to be sent; for example, a POST request). Sending a large request body to a server after a request has been rejected for inappropriate headers would be inefficient. To have a server check the request's headers, a client must send `Expect: 100-continue` as a header in its initial request and receive a `100 Continue` status code in response before sending the body. The response `417 Expectation Failed` indicates the request should not be continued.[5]

101 Switching Protocols
: The requester has asked the server to switch protocols and the server has agreed to do so.[6]

102 Processing (WebDAV; RFC 2518)
: A WebDAV request may contain many sub-requests involving file operations, requiring a long time to

complete the request. This code indicates that the server has received and is processing the request, but no response is available yet.[7] This prevents the client from timing out and assuming the request was lost.

2xx Success

This class of status codes indicates the action requested by the client was received, understood, accepted, and processed successfully.[8]

200 OK
Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request, the response will contain an entity describing or containing the result of the action.[9]

201 Created
The request has been fulfilled, resulting in the creation of a new resource.[10]

202 Accepted
The request has been accepted for processing, but the processing has not been completed. The request might or might not be eventually acted upon, and may be disallowed when processing occurs.[11]

203 Non-Authoritative Information (since HTTP/1.1)
The server is a transforming proxy (e.g. a Web accelerator) that received a 200 OK from its origin, but is returning a modified version of the origin's response.[12][13]

204 No Content
The server successfully processed the request and is not returning any content.[14]

205 Reset Content
The server successfully processed the request, but is not returning any content. Unlike a 204 response, this response requires that the requester reset the document view.[15]

206 Partial Content (RFC 7233)
The server is delivering only part of the resource (byte serving) due to a range header sent by the client. The range header is used by HTTP clients to enable resuming of interrupted downloads, or split a download into multiple simultaneous streams.[16]

207 Multi-Status (WebDAV; RFC 4918)
The message body that follows is an XML message and can contain a number of separate response codes, depending on how many sub-requests were made.[17]

208 Already Reported (WebDAV; RFC 5842)
The members of a DAV binding have already been enumerated in a previous reply to this request, and are not being included again.[18]

226 IM Used (RFC 3229)
The server has fulfilled a request for the resource, and the response is a representation of the result of one or more instance-manipulations applied to the current instance.[19]

3xx Redirection

This class of status code indicates the client must take additional action to complete the request. Many of these status codes are used in URL redirection.[20]

A user agent may carry out the additional action with no user interaction only if the method used in the second request is GET or HEAD. A user agent may automatically redirect a request. A user agent should detect and intervene to prevent cyclical redirects.[21]

300 Multiple Choices
Indicates multiple options for the resource from which the client may choose. For example, this code

could be used to present multiple video format options, to list files with different extensions, or to suggest word sense disambiguation.[22]

301 Moved Permanently

This and all future requests should be directed to the given URI.[23]

302 Found

This is an example of industry practice contradicting the standard. The HTTP/1.0 specification (RFC 1945) required the client to perform a temporary redirect (the original describing phrase was "Moved Temporarily"),[24] but popular browsers implemented 302 with the functionality of a 303 See Other. Therefore, HTTP/1.1 added status codes 303 and 307 to distinguish between the two behaviours.[25] However, some Web applications and frameworks use the 302 status code as if it were the 303.[26]

303 See Other (since HTTP/1.1)

The response to the request can be found under another URI using a GET method. When received in response to a POST (or PUT/DELETE), the client should presume that the server has received the data and should issue a redirect with a separate GET message.[27]

304 Not Modified (RFC 7232)

Indicates that the resource has not been modified since the version specified by the request headers If-Modified-Since or If-None-Match. In such case, there is no need to retransmit the resource since the client still has a previously-downloaded copy.[28]

305 Use Proxy (since HTTP/1.1)

The requested resource is available only through a proxy, the address for which is provided in the response. Many HTTP clients (such as Mozilla[29] and Internet Explorer) do not correctly handle responses with this status code, primarily for security reasons.[30]

306 Switch Proxy

No longer used. Originally meant "Subsequent requests should use the specified proxy."[31]

307 Temporary Redirect (since HTTP/1.1)

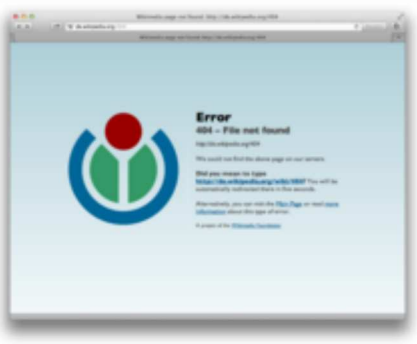
In this case, the request should be repeated with another URI; however, future requests should still use the original URI. In contrast to how 302 was historically implemented, the request method is not allowed to be changed when reissuing the original request. For example, a POST request should be repeated using another POST request.[32]

308 Permanent Redirect (RFC 7538)

The request and all future requests should be repeated using another URI. 307 and 308 parallel the behaviours of 302 and 301, but *do not allow the HTTP method to change*. So, for example, submitting a form to a permanently redirected resource may continue smoothly.[33]

4xx Client Error

The 4xx class of status code is intended for situations 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.[34]



404 error on German Wikipedia

400 Bad Request

The server cannot or will not process the request due to an apparent client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).[35]

401 Unauthorized (RFC 7235)

Similar to *403 Forbidden*, but specifically for use when authentication is required and has failed or has not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. See Basic access authentication and Digest access authentication.[36] 401 semantically means

"unauthenticated",[37] i.e. the user does not have the necessary credentials.
Note: Some sites issue HTTP 401 when an IP address is banned from the website (usually the website domain) and that specific address is refused permission to access a website.

402 Payment Required
Reserved for future use. The original intention was that this code might be used as part of some form of digital cash or micropayment scheme, but that has not happened, and this code is not usually used. Google Developers API uses this status if a particular developer has exceeded the daily limit on requests.[38]

403 Forbidden
The request was a valid request, but the server is refusing to respond to it. 403 error semantically means "unauthorized", i.e. the user does not have the necessary permissions for the resource.

404 Not Found
The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible.[39]

405 Method Not Allowed
A request method is not supported for the requested resource; for example, a GET request on a form which requires data to be presented via POST, or a PUT request on a read-only resource.

406 Not Acceptable
The requested resource is capable of generating only content not acceptable according to the Accept headers sent in the request.[40]

407 Proxy Authentication Required (RFC 7235)
The client must first authenticate itself with the proxy.[41]

408 Request Timeout
The server timed out waiting for the request. According to HTTP specifications: "The client did not produce a request within the time that the server was prepared to wait. The client MAY repeat the request without modifications at any later time."[42]

409 Conflict
Indicates that the request could not be processed because of conflict in the request, such as an edit conflict between multiple simultaneous updates.

410 Gone
Indicates that the resource requested is no longer available and will not be available again. This should be used when a resource has been intentionally removed and the resource should be purged. Upon receiving a 410 status code, the client should not request the resource in the future. Clients such as search engines should remove the resource from their indices.[43] Most use cases do not require clients and search engines to purge the resource, and a "404 Not Found" may be used instead.

411 Length Required
The request did not specify the length of its content, which is required by the requested resource.[44]

412 Precondition Failed (RFC 7232)
The server does not meet one of the preconditions that the requester put on the request.[45]

413 Payload Too Large (RFC 7231)
The request is larger than the server is willing or able to process. Previously called "Request Entity Too Large".[46]

414 URI Too Long (RFC 7231)
The URI provided was too long for the server to process. Often the result of too much data being encoded as a query-string of a GET request, in which case it should be converted to a POST request.[47] Called "Request-URI Too Long" previously.[48]

415 Unsupported Media Type
The request entity has a media type which the server or resource does not support. For example, the client uploads an image as image/svg+xml, but the server requires that images use a different format.

416 Range Not Satisfiable (RFC 7233)
The client has asked for a portion of the file (byte serving), but the server cannot supply that portion. For example, if the client asked for a part of the file that lies beyond the end of the file.[49] Called "Requested Range Not Satisfiable" previously.[50]

417 Expectation Failed

The server cannot meet the requirements of the Expect request-header field.[51]

418 I'm a teapot (RFC 2324)
This code was defined in 1998 as one of the traditional IETF April Fools' jokes, in RFC 2324, *Hyper Text Coffee Pot Control Protocol*, and is not expected to be implemented by actual HTTP servers. The RFC specifies this code should be returned by tea pots requested to brew coffee.[52] This HTTP status is used as an easter egg in some websites, including Google.com.[53]

421 Misdirected Request (RFC 7540)
The request was directed at a server that is not able to produce a response (for example because a connection reuse).[54]

422 Unprocessable Entity (WebDAV; RFC 4918)
The request was well-formed but was unable to be followed due to semantic errors.[17]

423 Locked (WebDAV; RFC 4918)
The resource that is being accessed is locked.[17]

424 Failed Dependency (WebDAV; RFC 4918)
The request failed due to failure of a previous request (e.g., a PROPPATCH).[17]

426 Upgrade Required
The client should switch to a different protocol such as TLS/1.0, given in the Upgrade header field.[55]

428 Precondition Required (RFC 6585)
The origin server requires the request to be conditional. Intended to prevent "the 'lost update' problem, where a client GETs a resource's state, modifies it, and PUTs it back to the server, when meanwhile a third party has modified the state on the server, leading to a conflict."[56]

429 Too Many Requests (RFC 6585)
The user has sent too many requests in a given amount of time. Intended for use with rate limiting schemes.[56]

431 Request Header Fields Too Large (RFC 6585)
The server is unwilling to process the request because either an individual header field, or all the header fields collectively, are too large.[56]

451 Unavailable For Legal Reasons
A server operator has received a legal demand to deny access to a resource or to a set of resources that includes the requested resource.[57] The code 451 was chosen as a reference to the novel Fahrenheit 451.

5xx Server Error

The server failed to fulfill an apparently valid request.[58]

Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has encountered an error or is otherwise incapable of performing the request. Except when responding to a HEAD request, the server *should* include an entity containing an explanation of the error situation, and indicate whether it is a temporary or permanent condition. Likewise, user agents *should* display any included entity to the user. These response codes are applicable to any request method.[59]

500 Internal Server Error
A generic error message, given when an unexpected condition was encountered and no more specific message is suitable.[60]

501 Not Implemented
The server either does not recognize the request method, or it lacks the ability to fulfill the request. Usually this implies future availability (e.g., a new feature of a web-service API).[61]

502 Bad Gateway
The server was acting as a gateway or proxy and received an invalid response from the upstream server.[62]

503 Service Unavailable
The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this

is a temporary state.[63]

504 Gateway Timeout
The server was acting as a gateway or proxy and did not receive a timely response from the upstream server.[64]

505 HTTP Version Not Supported
The server does not support the HTTP protocol version used in the request.[65]

506 Variant Also Negotiates (RFC 2295)
Transparent content negotiation for the request results in a circular reference.[66]

507 Insufficient Storage (WebDAV; RFC 4918)
The server is unable to store the representation needed to complete the request.[17]

508 Loop Detected (WebDAV; RFC 5842)
The server detected an infinite loop while processing the request (sent in lieu of 208 Already Reported).

510 Not Extended (RFC 2774 (https://www.ietf.org/rfc/rfc2774.txt))
Further extensions to the request are required for the server to fulfil it.[67]

511 Network Authentication Required (RFC 6585)
The client needs to authenticate to gain network access. Intended for use by intercepting proxies used to control access to the network (e.g., "captive portals" used to require agreement to Terms of Service before granting full Internet access via a Wi-Fi hotspot).[56]

Unofficial codes

The following codes are not specified by any RFC, but are used by third-party services to provide semantic or RESTful error responses:

103 Checkpoint
Used in the resumable requests proposal to resume aborted PUT or POST requests.

420 Method Failure (Spring Framework)
A deprecated response used by the Spring Framework when a method has failed.[68]

420 Enhance Your Calm (Twitter)
Returned by version 1 of the Twitter Search and Trends API when the client is being rate limited; versions 1.1 and later use the 429 Too Many Requests response code instead.[69]

450 Blocked by Windows Parental Controls (Microsoft)
A Microsoft extension. This error is given when Windows Parental Controls are turned on and are blocking access to the given webpage.[70]

498 Invalid Token (Esri)
Returned by ArcGIS for Server. A code of 498 indicates an expired or otherwise invalid token.[71]

499 Token Required (Esri)
Returned by ArcGIS for Server. A code of 499 indicates that a token is required but was not submitted.[71]

499 Request has been forbidden by antivirus
Produced by some programs such as Wget when a malicious site is intercepted.

509 Bandwidth Limit Exceeded (Apache Web Server/cPanel)
The server has exceeded the bandwidth specified by the server administrator; this is often used by shared hosting providers to limit the bandwidth of customers.[72]

530 Site is frozen
Used by the Pantheon web platform to indicate a site that has been frozen due to inactivity.

Internet Information Services

The Internet Information Services expands the 4xx error space to signal errors with the client's request.

440 Login Timeout

The client's session has expired and must log in again.[73]

449 Retry With
The server cannot honour the request because the user has not provided the required information.[74]

451 Redirect
Used in Exchange ActiveSync when either a more efficient server is available or the server cannot access the users' mailbox.[75] The client is expected to re-run the HTTP AutoDiscover operation to find a more appropriate server.[76]

nginx

The nginx web server software expands the 4xx error space to signal issues with the client's request.[77][78]

444 No Response
Used to indicate that the server has returned no information to the client and closed the connection.

495 SSL Certificate Error
An expansion of the 400 Bad Request response code, used when the client has provided an invalid client certificate.

496 SSL Certificate Required
An expansion of the 400 Bad Request response code, used when a client certificate is required but not provided.

497 HTTP Request Sent to HTTPS Port
An expansion of the 400 Bad Request response code, used when the client has made a HTTP request to a port listening for HTTPS requests.

499 Client Closed Request
Used when the client has closed the request before the server could send a response.

CloudFlare

CloudFlare's reverse proxy service expands the 5xx error space to signal issues with the origin server.[79]

520 Unknown Error
The 520 error is used as a "catch-all response for when the origin server returns something unexpected", listing connection resets, large headers, and empty or invalid responses as common triggers.

521 Web Server Is Down
The origin server has refused the connection from CloudFlare.

522 Connection Timed Out
CloudFlare could not negotiate a TCP handshake with the origin server.

523 Origin Is Unreachable
CloudFlare could not reach the origin server; for example, if the DNS records for the origin server are incorrect.

524 A Timeout Occurred
CloudFlare was able to complete a TCP connection to the origin server, but did not receive a timely HTTP response.

525 SSL Handshake Failed
CloudFlare could not negotiate a SSL/TLS handshake with the origin server.

526 Invalid SSL Certificate
CloudFlare could not validate the SSL/TLS certificate that the origin server presented.

See also

- Custom error pages
- List of FTP server return codes
- List of HTTP header fields

Notes

1. Italicised words and phrases such as *must* and *should* represent interpretation guidelines as given by RFC 2119

References

1. "Hypertext Transfer Protocol -- HTTP/1.1". *IETF*. Retrieved 16 October 2015.
2. "Hypertext Transfer Protocol (HTTP) Status Code Registry". Iana.org. Retrieved January 8, 2015.
3. "The HTTP status codes in IIS 7.0". Microsoft. July 14, 2009. Retrieved April 1, 2009.
4. "10 Status Code Definitions". *W3*. Retrieved 16 October 2015.
5. "HTTP Status Codes". *REST API Tutorial*. Retrieved 16 October 2015.
6. "101". *httpstatus*. Retrieved 16 October 2015.
7. Goland, Yaronn; Whitehead, Jim; Faizi, Asad; Carter, Steve R.; Jensen, Del (February 1999). *HTTP Extensions for Distributed Authoring – WEBDAV* (https://tools.ietf.org/html/rfc2518). IETF. RFC 2518. https://tools.ietf.org/html/rfc2518. Retrieved October 24, 2009.
8. "httpStatusCodes.js". *GitHub Gist*. marlun78. Retrieved 16 October 2015.
9. "qw3rtman \ hsc". *GitHub*. qw3rtman. Retrieved 16 October 2015.
10. Stewart, Mark; djna. "Create request with POST, which response codes 200 or 201 and content". *Stack Overflow*. Retrieved 16 October 2015.
11. "202". *httpstatus*. Retrieved 16 October 2015.
12. "RFC 7231, Section 6.3.4.".
13. "RFC 7230, Section 5.7.2.".
14. Simmance, Chris. "Server Response Codes And What They Mean". *koozai*. Retrieved 16 October 2015.
15. ikitommi; Deraen. "metosin/ring-http-response". *GitHub*. Retrieved 16 October 2015.
16. "diff --git a/linkchecker.module b/linkchecker.module". *Drupal*. Retrieved 16 October 2015.
17. Dusseault, Lisa, ed. (June 2007). *HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)* (https://tools.ietf.org/html/rfc4918). IETF. RFC 4918. https://tools.ietf.org/html/rfc4918. Retrieved October 24, 2009.
18. ikitommi; Daraen. "metosin/ring-http-response". *GitHub*. Retrieved 16 October 2015.
19. *Delta encoding in HTTP* (https://tools.ietf.org/html/rfc3229). IETF. January 2002. RFC 3229. https://tools.ietf.org/html/rfc3229. Retrieved February 25, 2011.
20. Sowa, Karen. "HTTP Diagram and Status Codes". *loggly*. Retrieved 16 October 2015.
21. "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content". *IETF*. Retrieved 13 February 2016.
22. "300". *httpstatus*. Retrieved 16 October 2015.
23. "301". *httpstatus*. Retrieved 16 October 2015.
24. Berners-Lee, Tim; Fielding, Roy T.; Nielsen, Henrik Frystyk (May 1996). *Hypertext Transfer Protocol – HTTP/1.0* (https://tools.ietf.org/html/rfc1945). IETF. RFC 1945. https://tools.ietf.org/html/rfc1945. Retrieved October 24, 2009.
25. "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Section 6.4". IETF. Retrieved June 12, 2014.
26. "Reference of method redirect_to in Ruby Web Framework "Ruby on Rails". It states: The redirection happens as a "302 Moved" header unless otherwise specified.". Retrieved June 30, 2012.
27. "303". *httpstatus*. Retrieved 16 October 2015.
28. Brown, Kevin; CRS... "getting 304 response even with django-cors-headers". *Stack Overflow*. Retrieved 16 October 2015.
29. "Mozilla Bugzilla Bug 187996: Strange behavior on 305 redirect". March 3, 2003. Retrieved May 21, 2009.
30. "Mozilla Bugzilla Bug 187996: Strange behavior on 305 redirect, comment 13". March 3, 2003. Retrieved May 21, 2009.
31. Cohen, Josh. "HTTP/1.1 305 and 306 Response Codes". HTTP Working Group.
32. "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Section 6.4.7 307 Temporary Redirect". IETF. 2014. Retrieved September 20, 2014.
33. "The Hypertext Transfer Protocol Status Code 308 (Permanent Redirect)". Internet Engineering Task Force. April 2015. Retrieved 2015-04-06.
34. "E Explanation of Failure Codes". *Oracle*. Retrieved 16 October 2015.
35. "RFC7231 on code 400". Tools.ietf.org. Retrieved January 8, 2015.
36. "401". *httpstatus*. Retrieved 16 October 2015.
37. "RFC7235 on code 401". Tools.ietf.org. Retrieved February 8, 2015.
38. "Google API Standard Error Responses". Google. 2015. Retrieved September 24, 2015.

39. "Introduction". *QAS*. Retrieved 16 October 2015.
40. Singh, Prabhat; user1740567. "Spring 3.x JSON status 406 "characteristics not acceptable according to the request "accept" headers ()" ". *Stack Overflow*. Retrieved 16 October 2015.
41. "407". *httpstatus*. Retrieved 16 October 2015.
42. "408". *httpstatus*. Retrieved 16 October 2015.
43. "Does Google treat 404 and 410 status codes differently? (Youtube)". Google. 2014. Retrieved February 4, 2015.
44. "List of HTTP status codes". *Google Books*. Retrieved 16 October 2015.
45. Kowser; Patel, Amit. "REST response code for invalid data". *Stack Overflow*. Retrieved 16 October 2015.
46. "RFC2616 on status 413". Tools.ietf.org. Retrieved November 11, 2015.
47. user27828. "GET Request - Why is my URI so long?". *Stack Overflow*. Retrieved 16 October 2015.
48. "RFC2616 on status 414". Tools.ietf.org. Retrieved November 11, 2015.
49. Sigler, Chris. "416 Requested Range Not Satisfiable". *GetStatusCode*. Retrieved 16 October 2015.
50. "RFC2616 on status 416". Tools.ietf.org. Retrieved November 11, 2015.
51. TheDeadLike. "HTTP/1.1 Status Codes 400 and 417, cannot choose which". *serverFault*. Retrieved 16 October 2015.
52. Larry Masinter (1 April 1998). *Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)* (https://tools.ietf.org/html/rfc2324). RFC 2324. https://tools.ietf.org/html/rfc2324.
53. Barry Schwartz (26 August 2014). "New Google Easter Egg For SEO Geeks: Server Status 418, I'm A Teapot". *Search Engine Land*.
54. "Hypertext Transfer Protocol version 2". March 2015. Retrieved April 25, 2015.
55. Khare, R; Lawrence, S. "Upgrading to TLS Within HTTP/1.1". *IETF*. Network Working Group. Retrieved 16 October 2015.
56. Nottingham, M.; Fielding, R. (April 2012). "RFC 6585 – Additional HTTP Status Codes". *Request for Comments*. Internet Engineering Task Force. Retrieved May 1, 2012.
57. Bray, T. (February 2016). "An HTTP Status Code to Report Legal Obstacles". *ietf.org*. Retrieved 7 March 2015.
58. "Server Error Codes". *CSGNetwork.com*. Retrieved 16 October 2015.
59. mrGott. "HTTP Status Codes To Handle Errors In Your API". *mrGott*. Retrieved 16 October 2015.
60. Meredith, Kevin. "HTTP Response for Unsuccessful Handling of Request". *Stack Overflow*. Retrieved 16 October 2015.
61. ThatGuyFromFinland. "When Database Table does not exist for a REST service, HTTP error status code should be 501 instead of 500 #107". *GitHub*. Retrieved 16 October 2015.
62. Fisher, Tim. "502 Bad Gateway". *about tech*. Retrieved 16 October 2015.
63. alex. "What is the correct HTTP status code to send when a site is down for maintenance?". *Stack Overflow*. Retrieved 16 October 2015.
64. "HTTP Error 504 Gateway timeout". *Check Up Down*. Retrieved 16 October 2015.
65. "HTTP Error 505 - HTTP version not supported". *Check Up Down*. Retrieved 16 October 2015.
66. Holtman, Koen; Mutz, Andrew H. (March 1998). *Transparent Content Negotiation in HTTP* (https://tools.ietf.org/html/rfc2295). IETF. RFC 2295. https://tools.ietf.org/html/rfc2295. Retrieved October 24, 2009.
67. Nielsen, Henrik Frystyk; Leach, Paul J.; Lawrence, Scott (February 2000). *An HTTP Extension Framework* (https://tools.ietf.org/html/rfc2774). IETF. RFC 2774. https://tools.ietf.org/html/rfc2774. Retrieved October 24, 2009.
68. "Enum HttpStatus". *Spring Framework*. org.springframework.http. Retrieved 16 October 2015.
69. "Twitter Error Codes & Responses". Twitter. 2014. Retrieved January 20, 2014.
70. "Screenshot of error page" (bmp). Retrieved October 11, 2009.
71. "Using token-based authentication". *ArcGIS Server SOAP SDK*.
72. "HTTP Error Codes and Quick Fixes". Docs.cpanel.net. Retrieved October 15, 2015.
73. "Error message when you try to log on to Exchange 2007 by using Outlook Web Access: "440 Login Timeout" ". Microsoft. 2010. Retrieved November 13, 2013.
74. "2.2.6 449 Retry With Status Code". Microsoft. 2009. Retrieved October 26, 2009.
75. "MS-ASCMD, Section 3.1.5.2.2". Msdn.microsoft.com. Retrieved January 8, 2015.
76. "Ms-oxdisco". Msdn.microsoft.com. Retrieved January 8, 2015.
77. "ngx_http_request.h". *nginx 1.9.5 source code*. nginx inc. Retrieved 2016-01-09.
78. "ngx_http_special_response.c". *nginx 1.9.5 source code*. nginx inc. Retrieved 2016-01-09.
79. "Troubleshooting: Error Pages". Cloudflare. Retrieved 2016-01-09.

External links

- Official website (https://www.iana.org/assignments/http-status-codes) Hypertext Transfer Protocol (HTTP) Status Code Registry
- Microsoft Knowledge Base: MSKB943891: The HTTP status codes in IIS 7.0 (http://support.microsoft.com/kb/943891/)
- Help for HTTP errors (http://www.getnetgoing.com/HTTP.html)
- Test any HTTP status code in a web browser (http://savanttools.com/test-http-status-codes)
- Checking tool for URL for status code with description (http://www.checkhttpstatuscodes.com)
- Real time server-side 503 HTTP status code checker tool (http://www.serviceunavailable.info/)
- Choosing an HTTP Status Code — Stop Making It Hard (http://racksburg.com/choosing-an-http-status-code/)

Wikimedia Commons has media related to ***HTTP***.

Categories: Hypertext Transfer Protocol status codes | Internet-related lists

---