List of HTTP status codes

From Wikipedia, the free encyclopedia

The following is a list of **HyperText Transfer Protocol (HTTP) response status codes**. This includes codes from IETF internet standards as well as unstandardised RFCs, other specifications and some additional commonly used codes. The first digit of the status code specifies one of five classes of response; the bare minimum for an HTTP client is that it recognises these five classes. Microsoft IIS may use additional decimal sub-codes to provide more specific information,[1] but these are not listed here. The phrases used are the standard examples, but any human-readable alternative can be provided. Unless otherwise stated, the status code is part of the HTTP/1.1 standard.

HTTP

Persistence · Compression · HTTP Secure

Headers

ETag · Cookie · Referrer · Location

Status codes

301 Moved permanently

302 Found

303 See Other

403 Forbidden

404 Not Found

Contents

1xx Informational

Request received, continuing process.[2]

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* send a 1xx response to an HTTP/1.0 client except under experimental conditions.

100 Continue
: This means that the server has received the request headers, and that 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). If the request body is large, sending it to a server when a request has already been rejected based upon inappropriate headers is inefficient. To have a server check if the request could be accepted based on the request's headers alone, a client must send `Expect: 100-continue` as a header in its initial request[2] and check if a `100 Continue` status code is received in response before continuing (or receive `417 Expectation Failed` and not continue).[2]

101 Switching Protocols

This means the requester has asked the server to switch protocols and the server is acknowledging that it will do so.[2]

102 Processing (WebDAV) (RFC 2518)
: As a WebDAV request may contain many sub-requests involving file operations, it may take 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.[3] 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.

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.[2]

201 Created
: The request has been fulfilled and resulted in a new resource being created.[2]

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.[2]

203 Non-Authoritative Information (since HTTP/1.1)
: The server successfully processed the request, but is returning information that may be from another source.[2]

204 No Content
: The server successfully processed the request, but is not returning any content.[2]

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.[2]

206 Partial Content
: The server is delivering only part of the resource due to a range header sent by the client. This is used by tools like wget to enable resuming of interrupted downloads, or split a download into multiple simultaneous streams.[2]

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.[4]

3xx Redirection

The client must take additional action to complete the request.[2]

This class of status code indicates that further action needs to be taken by the user agent in order to fulfil 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 user agent *should not* automatically redirect a request more than five times, since such redirections usually indicate an infinite loop.

300 Multiple Choices

Indicates multiple options for the resource that the client may follow. It, for instance, could be used to present different format options for video, list files with different extensions, or word sense disambiguation.[2]

301 Moved Permanently

This and all future requests should be directed to the given URI.[2]

302 Found

This is the most popular redirect code[*citation needed*], but also an example of industrial practice contradicting the standard.[2] HTTP/1.0 specification (RFC 1945) required the client to perform a temporary redirect (the original describing phrase was "Moved Temporarily"),[5] 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. However, the majority of Web applications and frameworks still use the 302 status code as if it were the 303[*citation needed*].

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 PUT, it should be assumed that the server has received the data and the redirect should be issued with a separate GET message.[2]

304 Not Modified

Indicates the resource has not been modified since last requested.[2] Typically, the HTTP client provides a header like the If-Modified-Since header to provide a time against which to compare. Utilizing this saves bandwidth and reprocessing on both the server and client, as only the header data must be sent and received in comparison to the entirety of the page being re-processed by the server, then resent using more bandwidth of the server and client.

305 Use Proxy (since HTTP/1.1)

Many HTTP clients (such as Mozilla[6] and Internet Explorer) do not correctly handle responses with this status code, primarily for security reasons.[2]

306 Switch Proxy

No longer used.[2]

307 Temporary Redirect (since HTTP/1.1)

In this occasion, the request should be repeated with another URI, but future requests can still use the original URI.[2] In contrast to 303, the request method should not be changed when reissuing the original request. For instance, a POST request must be repeated using another POST request.

4xx Client Error

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. These are typically the most common error codes encountered while online.

400 Bad Request

The request contains bad syntax or cannot be fulfilled.[2]

401 Unauthorized

Similar to *403 Forbidden*, but specifically for use when authentication is possible but has failed or not yet been provided.[2] 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.

402 Payment Required

Reserved for future use.[2] 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. As an example of its use, however, Apple's MobileMe service generates a 402 error ("httpStatusCode:402" in the Mac OS X Console log) if the MobileMe account is delinquent.

403 Forbidden

The request was a legal request, but the server is refusing to respond to it.[2] Unlike a *401 Unauthorized* response, authenticating will make no difference.[2]

404 Not Found

The requested resource could not be found but may be available again in the future.[2] Subsequent requests by the client are permissible.

405 Method Not Allowed

A request was made of a resource using a request method not supported by that resource;[2] for example, using GET on a form which requires data to be presented via POST, or using PUT on a read-only resource.

406 Not Acceptable

The requested resource is only capable of generating content not acceptable according to the Accept headers sent in the request.[2]

407 Proxy Authentication Required[2]

408 Request Timeout

The server timed out waiting for the request.[2] According to W3 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."

409 Conflict

Indicates that the request could not be processed because of conflict in the request, such as an edit conflict.[2]

410 Gone

Indicates that the resource requested is no longer available and will not be available again.[2] This should be used when a resource has been intentionally removed; however, it is not necessary to return this code and a *404 Not Found* can be issued instead. Upon receiving a 410 status code, the client should not request the resource again in the future. Clients such as search engines should remove the resource from their indexes.

411 Length Required

The request did not specify the length of its content, which is required by the requested resource.[2]

412 Precondition Failed

The server does not meet one of the preconditions that the requester put on the request.[2]

413 Request Entity Too Large

The request is larger than the server is willing or able to process.[2]

414 Request-URI Too Long

The URI provided was too long for the server to process.[2]

415 Unsupported Media Type

The request did not specify any media types that the server or resource supports.[2] For example the client specified that an image resource should be served as image/svg+xml, but the server cannot find a matching version of the image.

416 Requested Range Not Satisfiable

The client has asked for a portion of the file, but the server cannot supply that portion.[2] For example, if the client asked for a part of the file that lies beyond the end of the file.

417 Expectation Failed

The server cannot meet the requirements of the Expect request-header field.[2]

418 I'm a teapot
: The HTCPCP server is a teapot.[7] The responding entity MAY be short and stout.[7] This code was defined 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.

422 Unprocessable Entity (WebDAV) (RFC 4918)
: The request was well-formed but was unable to be followed due to semantic errors.[4]

423 Locked (WebDAV) (RFC 4918)
: The resource that is being accessed is locked[4]

424 Failed Dependency (WebDAV) (RFC 4918)
: The request failed due to failure of a previous request (e.g. a PROPPATCH).[4]

425 Unordered Collection (RFC 3648)
: Defined in drafts of "WebDAV Advanced Collections Protocol",[8] but not present in "Web Distributed Authoring and Versioning (WebDAV) Ordered Collections Protocol".[9]

426 Upgrade Required (RFC 2817)
: The client should switch to a different protocol such as TLS/1.0.[10]

449 Retry With
: A Microsoft extension. The request should be retried after doing the appropriate action.[11]

450 Blocked by Windows Parental Controls
: A Microsoft extension. This error is given when Windows Parental Controls are turned on and are blocking access to the given webpage.[12]

5xx Server Error

The server failed to fulfill an apparently valid request.[2]

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.

500 Internal Server Error
: A generic error message, given when no more specific message is suitable.[2]

501 Not Implemented
: The server either does not recognise the request method, or it lacks the ability to fulfill the request.[2]

502 Bad Gateway
: The server was acting as a gateway or proxy and received an invalid response from the upstream server.[2]

503 Service Unavailable
: The server is currently unavailable (because it is overloaded or down for maintenance).[2] Generally, this is a temporary state.

504 Gateway Timeout
: The server was acting as a gateway or proxy and did not receive a timely request from the upstream server.[2]

505 HTTP Version Not Supported
: The server does not support the HTTP protocol version used in the request.[2]

506 Variant Also Negotiates (RFC 2295)

Transparent content negotiation for the request, results in a circular reference.[13]

507 Insufficient Storage (WebDAV) (RFC 4918)[4]

509 Bandwidth Limit Exceeded (Apache bw/limited extension)

This status code, while used by many servers, is not specified in any RFCs.

510 Not Extended (RFC 2774)

Further extensions to the request are required for the server to fulfill it.[14]

See also

- List of HTTP headers
- List of FTP server return codes
- Custom error page

References

1. ^ "The HTTP status codes in IIS 7.0" (http://support.microsoft.com/kb/943891/) . Microsoft. July 14, 2009. http://support.microsoft.com/kb/943891/. Retrieved April 1, 2009.
2. ^ *a b c d e f g h i j k l m n o p q r s t u v w x y z aa ab ac ad ae af ag ah ai aj ak al am an ao ap aq ar as at* Fielding, Roy T.; Gettys, James; Mogul, Jeffrey C.; Nielsen, Henrik Frystyk; Masinter, Larry; Leach, Paul J.; Berners-Lee, Tim (June 1999). *Hypertext Transfer Protocol -- HTTP/1.1* (http://tools.ietf.org/html/rfc2616) . IETF. RFC 2616. http://tools.ietf.org/html/rfc2616. Retrieved October 24, 2009.
3. ^ Goland, Yaron; Whitehead, Jim; Faizi, Asad; Carter, Steve R.; Jensen, Del (February 1999). *HTTP Extensions for Distributed Authoring -- WEBDAV* (http://tools.ietf.org/html/rfc2518) . IETF. RFC 2518. http://tools.ietf.org/html/rfc2518. Retrieved October 24, 2009.
4. ^ *a b c d e* Dusseault, Lisa, ed (June 2007). *HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)* (http://tools.ietf.org/html/rfc4918) . IETF. RFC 4918. http://tools.ietf.org/html/rfc4918. Retrieved October 24, 2009.
5. ^ Berners-Lee, Tim; Fielding, Roy T.; Nielsen, Henrik Frystyk (May 1996). *Hypertext Transfer Protocol -- HTTP/1.0* (http://tools.ietf.org/html/rfc1945) . IETF. RFC 1945. http://tools.ietf.org/html/rfc1945. Retrieved October 24, 2009.
6. ^ "Mozilla Bugzilla Bug 187996: Strange behavior on 305 redirect" (https://bugzilla.mozilla.org/show_bug.cgi?id=187996) . March 3, 2003. https://bugzilla.mozilla.org/show_bug.cgi?id=187996. Retrieved May 21, 2009.
7. ^ *a b* Masinter, Larry (April 1, 1998). *Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)* (http://tools.ietf.org/html/rfc2324) . IETF. RFC 2324. http://tools.ietf.org/html/rfc2324. Retrieved October 24, 2009.
8. ^ Slein, Judy; Whitehead, Jim; Davis, Jim; Clemm, Geoffrey; Fay, Chuck; Crawford, Jason; Chihaya, Tyson (June 18, 1999). *WebDAV Advanced Collections Protocol* (http://tools.ietf.org/html/draft-ietf-webdav-collection-protocol-04) . IETF. I-D draft-ietf-webdav-collection-protocol-04. http://tools.ietf.org/html/draft-ietf-webdav-collection-protocol-04. Retrieved October 24, 2009.
9. ^ Whitehead, Jim (December 2003). Reschke, Julian F.. ed. *Web Distributed Authoring and Versioning (WebDAV) Ordered Collections Protocol* (http://tools.ietf.org/html/rfc3648) . IETF. RFC 3648. http://tools.ietf.org/html/rfc3648. Retrieved October 24, 2009.
10. ^ Khare, Rohit; Lawrence, Scott (May 2000). *Upgrading to TLS Within HTTP/1.1* (http://tools.ietf.org/html/rfc2817) . IETF. RFC 2817. http://tools.ietf.org/html/rfc2817. Retrieved October 24, 2009.
11. ^ "2.2.6 449 Retry With Status Code" (http://msdn.microsoft.com/en-us/library/dd891478(PROT.10).aspx) . Microsoft. 2009. http://msdn.microsoft.com/en-us/library/dd891478(PROT.10).aspx. Retrieved October 26, 2009.
12. ^ "Screenshot of error page" (http://zfhb6a.bay.livefilestore.com/y1pKZJpcqDcSF9uKwaTmx301Ilr7cbJGN94HXCoHvPGwuwAlt5DA4ln0Y-F1WE6ZUC3URdiJdRe4hILTo87jWx2Yg) (bmp). http://zfhb6a.bay.livefilestore.com/y1pKZJpcqDcSF9uKwaTmx301Ilr7cbJGN94HXCoHvPGwuwAlt5DA4ln0Y-F1WE6ZUC3URdiJdRe4hILTo87jWx2Yg. Retrieved

October 11, 2009.
13. ^ Holtman, Koen; Mutz, Andrew H. (March 1998). *Transparent Content Negotiation in HTTP* (http://tools.ietf.org/html/rfc2295) . IETF. RFC 2295. http://tools.ietf.org/html/rfc2295. Retrieved October 24, 2009.
14. ^ Nielsen, Henrik Frystyk; Leach, Paul J.; Lawrence, Scott (February 2000). *An HTTP Extension Framework* (http://tools.ietf.org/html/rfc2774) . IETF. RFC 2774. http://tools.ietf.org/html/rfc2774. Retrieved October 24, 2009.

External Links

- Adobe Flash status code definitions (i.e. 408) (http://livedocs.adobe.com/fms/2/docs/00000338.html)
- Google Help: HTTP status codes (http://www.google.com/support/webmasters/bin/answer.py?hl=en&answer=40132)
- Help for HTTP errors (http://www.getnetgoing.com/HTTP.html)
- IANA registry (http://www.iana.org/assignments/http-status-codes)
- Microsoft Internet Information Services Status Codes and Sub-Codes (http://support.microsoft.com/kb/943891/)

Retrieved from "http://en.wikipedia.org/wiki/List_of_HTTP_status_codes"
Categories: HTTP | HTTP status codes | Internet-related lists