Response codes of the Hypertext Transfer Protocol This is a list of Hypertext Transfer Protocol (HTTP) response status codes. Status codes are issued by a server in response to a client's request made to the server. It includes codes from IETF Request for Comments (RFCs), other specifications, and some additional codes used in some common applications of the HTTP. The first digit of the status code specifies one of five standard classes of responses. The optional message phrases shown are typical, but any human-readable alternative may be provided, or none at all. Unless otherwise stated, the status code is part of the HTTP standard.[1] The Internet Assigned Numbers Authority (IANA) maintains the official registry of HTTP status codes.[2] All HTTP response status codes are separated into five classes or categories. The first digit of the status code defines the class of response, while the last two digits do not have any classifying or categorization role. There are five classes defined by the standard: 1xx informational response – the request was received, continuing process 2xx successful – the request was successfully received, understood, and accepted 3xx redirection – further action needs to 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 fulfil an apparently valid request 1xx informational response An informational response indicates that the request was received and understood. It is issued on a provisional basis while request processing continues. It alerts the client to wait for a final response. The message consists only of the status line and optional header fields, and is terminated by an empty line. As the HTTP/1.0 standard did not define any 1xx status codes, servers must not[note 1] send a 1xx response to an HTTP/1.0 compliant client except under experimental conditions. 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. If the client receives an error code such as 403 (Forbidden) or 405 (Method Not Allowed) then it should not send the request's body. The response 417 Expectation Failed indicates that the request should be repeated without the Expect header as it indicates that the server does not support expectations (this is the case, for example, of HTTP/1.0 servers).[1]: §10.1.1 101 Switching Protocols The requester has asked the server to switch protocols and the server has agreed to do so. 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.[3] This prevents the client from timing out and assuming the request was lost. The status code is deprecated.[4] 103 Early Hints (RFC 8297) Used to return some response headers before final HTTP message.[5] 2xx success This class of status codes indicates the action requested by the client was received, understood, and accepted.[2] 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. 201 Created The request has been fulfilled, resulting in the creation of a new resource.[6] 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. 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.[1]: §15.3.4 [1]: §7.7 204 No Content The server successfully processed the request, and is not returning any content. 205 Reset Content The server successfully processed the request, asks that the requester reset its document view, and is not returning any content. 206 Partial Content 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. 207 Multi-Status (WebDAV; RFC 4918) The message body that follows is by default an XML message and can contain a number of separate response codes, depending on how many sub-requests were made.[7] 208 Already Reported (WebDAV; RFC 5842) The members of a DAV binding have already been enumerated in a preceding part of the (multistatus) response, and are not being included again. 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.[8] 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.[2] 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.[1]: §15.4 300 Multiple Choices Indicates multiple options for the resource from which the client may choose (via agent-driven content negotiation). For example, this code could be used to present multiple video format options, to list files with different filename extensions, or to suggest word-sense disambiguation. 301 Moved Permanently This and all future

requests should be directed to the given URI. 302 Found (Previously "Moved temporarily") Tells the client to look at (browse to) another URL. The HTTP/1.0 specification required the client to perform a temporary redirect with the same method (the original describing phrase was "Moved Temporarily"),[9] but popular browsers implemented 302 redirects by changing the method to GET. Therefore, HTTP/1.1 added status codes 303 and 307 to distinguish between the two behaviours.[1]: §15.4 303 See Other (since HTTP/1.1) The response to the request can be found under another URI using the 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 new GET request to the given URI. 304 Not Modified 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. 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. For security reasons, many HTTP clients (such as Mozilla Firefox and Internet Explorer) do not obey this status code.[10] 306 Switch Proxy No longer used. Originally meant "Subsequent requests should use the specified proxy." 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. 308 Permanent Redirect This and all future requests should be directed to the given URI. 308 parallel the behaviour of 301, but does not allow the HTTP method to change. So, for example, submitting a form to a permanently redirected resource may continue smoothly