HTTP Status Codes

1xx - information

➔ -> 100: continue

The client should continue the 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.

➔ 101: Switching protocol

The server understands and is willing to comply with the client's request, via the Upgrade message header field, 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 responses.

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.

➔ 102: Processing(WebDAV)

The 102 (Processing) status code is an interim response used to inform the client that the server has accepted the complete request, but has not yet completed it.This status code SHOULD only be sent when the server has a reasonable expectation that the request will take significant time to complete

2xx: Success

➔ 200: ok

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

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 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

Successful creation occurred (via either POST or PUT). Set the Location header to contain a link to the newly-created resource (on POST). Response body content may or may not be present.

➔ 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.

➔ 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.

➔ 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.

➔ 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.

➔ 206: Partial content

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

➔ 207: (Multi-Status)

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.

➔ 208: (Already Reported)

The members of a DAV binding have already been enumerated in a previous reply to this request, and are not being included again.

➔ 226: IM Used

The server has fulfilled a GET request for the resource, and the response is a representation of the result of one or more instance-manipulations applied to the current instance.

3xx: Redirection

➔ 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.

➔ 301: Moved Permanently

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

➔ 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.

➔ 303: See other

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

➔ 304: Not modified

Indicates the resource has not been modified since last requested. Typically, the HTTP client provides a header like the If-Modified-Since header to provide a time against which to compare. Using 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 sent again using more bandwidth of the server and client.

➔ 305: Use proxy

The requested resource MUST be accessed through the proxy given by the Location field.

➔ 306: Unused

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

➔ 307: Temporary Redirect

The requested resource resides temporarily under a different URI.

➔ 308: Permanent Redirect
➔ The request, and all future requests should be repeated using another URI.

4xx: Client error

➔ 400: Bad request

The request could not be understood by the server due to malformed syntax.

➔ 401: Unauthorized

The request requires user authentication.

➔ 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.

➔ 403: Forbidden

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

➔ 404: Not found

The requested resource could not be found but may be available again in the future.

➔ 405: Method not allowed

A request was made of a resource using a request method not supported by that resource;

➔ 406: Not acceptable

The resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request.

➔ 407:Proxy Authn Required

The client must first authenticate itself with the proxy.

➔ 408: Request Timeout

The client did not produce a request within the time that the server was prepared to wait.

➔ 409: Conflict

The request could not be completed due to a conflict with the current state of the resource. This code is only allowed in situations where it is expected that the user might be able to resolve the conflict and resubmit the request.

➔ 410: Gone

The requested resource is no longer available at the server and no forwarding address is known.

➔ 411: Length Required

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

➔ 412: Precondition failed

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

➔ 413: Request entity too large

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

➔ 414: Request-URI Too Long
The URI provided was too long for the server to process.

➔ 415: Unsupported Media Type

The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.

➔ 416: Requested Range Not Satisfiable

The client has asked for a portion of the file, 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.

➔ 417: Expectation Failed
The server cannot meet the requirements of the Expect request-header field.

➔ 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. However, known implementations do exist. An Nginx HTTP server uses this code to simulate goto-like behaviour in its configuration.

➔ 420: Enhance Your Calm (Twitter)

Returned by the Twitter Search and Trends API when the client is being rate limited. The text is a quote from 'Demolition Man' and the '420' code is likely a reference to this number's association with marijuana.

➔ 422: Unprocessable Entity (WebDAV)
The request was well-formed but was unable to be followed due to semantic errors.

➔ 423: Locked (WebDAV)
The resource that is being accessed is locked.

➔ 424: Failed Dependency (WebDAV)
The request failed due to failure of a previous request

➔ 425: Reserved for WebDAV
Defined in drafts of "WebDAV Advanced Collections Protocol", but not present in "Web Distributed Authoring and Versioning (WebDAV) Ordered Collections Protocol".

➔ 426: Upgrade Required
Reliable, interoperable negotiation of Upgrade features requires an unambiguous failure signal.

➔ 428: Precondition Required
The 428-status code indicates that the origin server requires the request to be conditional.

➔ 429: Too Many Requests
The user has sent too many requests in a given amount of time. Intended for use with rate limiting schemes.

➔ 431: Request Header Fields Too Large
The server is unwilling to process the request because either an individual header field, or all the header fields collectively, are too large.

➔ 444::No Response (Nginx)
An Nginx HTTP server extension. The server returns no information to the client and closes the connection (useful as a deterrent for malware)

➔ 449: Retry With (Microsoft)
A Microsoft extension. The request should be retried after performing the appropriate action.

➔ 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.

➔ 451: Unavailable For Legal Reasons
Intended to be used when resource access is denied for legal reasons, e.g. censorship or government-mandated blocked access.

➔ 499: Client Closed Request (Nginx)
An Nginx HTTP server extension. This code is introduced to log the case when the connection is closed by client while HTTP server is processing its request, making server unable to send the HTTP header back.

5xx: Server Error

➔ 500 Internal Server Error
The server encountered an unexpected condition which prevented it from fulfilling the request.

➔ 501 Not Implemented
The server does not support the functionality required to fulfill the request.

- 502 Bad Gateway
  The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.

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

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

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

- 506 Variant Also Negotiates (Experimental)
  The 506 status code indicates that the server has an internal configuration error:

- 507 Insufficient Storage (WebDAV)
  The server is unable to store the representation needed to complete the request.

- 508 Loop Detected (WebDAV)
  The server detected an infinite loop while processing the request

- 509 Bandwidth Limit Exceeded (Apache)
  This status code, while used by many servers, is not specified in any RFCs

- 510 Not Extended
  The policy for accessing the resource has not been met in the request.

- 511 Network Authentication Required
  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).

- 598 Network read timeout error
  This status code is not specified in any RFCs, but is used by some HTTP proxies to signal a network read timeout behind the proxy to a client in front of the proxy.

- 599 Network connect timeout error
  This status code is not specified in any RFCs, but is used by some HTTP proxies to signal a network connect timeout behind the proxy to a client in front of the proxy.