HTTP RESPONSE CODE

HTTP messages are how data is exchanged between a server and **a client**. There are two types of messages: requests sent by the client to trigger an action on the server, and responses, the answer from the server.

HTTP Response is the server's information as a result of the client's request. Additionally, it acts as an acknowledgment that the performance of the requested action is successful. In case there is an error in carrying out the client's request, the server responds with an error message. Moreover, the HTTP responses come as plain text formatted in either JSON or XML format, just like the HTTP requests. In the next section, let us see how an HTTP response looks.

An HTTP response contains:

1. A status line.
2. A series of HTTP headers, or header fields.
3. A message body, which is usually needed.

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* 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. I

101 Switching Protocols

The requester has asked the server to switch protocols and the server has agreed to do so.

102 Processing

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. This prevents the client from timing out and assuming the request was lost.

103 Early Hints

Used to return some response headers before final HTTP message.

2X SUCCESS

This class of status codes indicates the action requested by the client was received, understood, and accepted.

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.

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

The server is a transforming proxy that received a 200 OK from its origin, but is returning a modified version of the origin's response.

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 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

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.

208 Already Reported
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
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.

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 URL redirection.

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.

300 Multiple Choices

Indicates multiple options for the resource from which the client may choose

301 MOVED PERMANENTLY
This and all future requests should be directed to the given URI

302 FOUND

Tells the client to look at (browse to) another URL. 302 has been superseded by 303 and 307. 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"),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, some Web applications and frameworks use the 302 status code as if it were the 303.
303 see other
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 (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.
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 do not obey this status code.

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 (RFC 7538)

The request and all future requests should be repeated using another URI. 307 and 308 parallel the behaviors of 302 and 301, but *do not allow the HTTP method to change*.

4X Client Server

This class of status code is intended for situations in which the error seems to have been caused by the client. 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

400 Bad Request

The server cannot or will not process the request due to an apparent client error

401 Unauthorized

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.

402 Payment Required

Reserved for future use.

403 Forbidden

The request contained valid data and was understood by the server, but the server is refusing action. This may be due to the user not having the necessary permissions for a resource or needing an account of some sort, or attempting a prohibited action.

404 Not Found

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

405 Method Not Allowed

A request method is not supported for the requested resource; for example, a GET request on a form that 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.

407 Proxy Authentication Required

The client must first authenticate itself with the proxy

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."

409 Conflict

Indicates that the request could not be processed because of conflict in the current state of the resource, 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. 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.

412 Precondition Failed

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

413 Payload Too Large

The request is larger than the server is willing or able to process. Previously called "Request Entity Too Large".

416 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. Called "Requested Range Not Satisfiable" previously.

417 Expectation Failed

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

421 Misdirected Request

The request was directed at a server that is not able to produce a response

422 Unprocessable Entity

The request was well-formed but was unable to be followed due to semantic errors.

423 Locked

The resource that is being accessed is locked.

424 Failed Dependency

The request failed because it depended on another request and that request failed (e.g., a PROPPATCH).

425 Too Early

Indicates that the server is unwilling to risk processing a request that might be replayed.

426 Upgrade Required

The client should switch to a different protocol such as TLS/1.3, given in the upgrade header field.

428 Precondition Required

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.

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.

451 Unavailable For a legal reason

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.

5xx Server Errors

The server failed to fulfil a request.

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 an unexpected condition was encountered and no more specific message is suitable.

501 Not Implemented

The server either does not recognize the request method, or it lacks the ability to fulfil the request. Usually this implies future availability.

502 Bad Gateway

The server was acting as a Gateway or proxy and received an invalid response from the upstream server.

503 Service Unavailable

The server cannot handle the request (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

Transparent content negation for the request results in a circular reference.

507 Insufficient Storage

The server is unable to store the representation needed to complete the request.

508 Loop Detected

The server detected an infinite loop while processing the request

510 Not Extended

Further extensions to the request are required for the server to fulfil it.