**HTTP Status Codes:**

|  |  |
| --- | --- |
| **CATEGORY** | **DESCRIPTION** |
| **1xx: Informational** | Communicates transfer protocol-level information. |
| **2xx: Success** | Indicates that the client’s request was accepted successfully. |
| **3xx: Redirection** | Indicates that the client must take some additional action in order to complete their request. |
| **4xx: Client Error** | This category of error status codes points the finger at clients |
| **5xx: Server Error** | The server takes responsibility for these error status codes. |

**1xx: Informational:**

**100 Continue**

The server has received the request headers and the client should proceed to send the request body. 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.

**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.[[7]](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#cite_note-RFC_2518-8) 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.

**2xx: Success:**

**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 (since HTTP/1.1)**

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, 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 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](https://en.wikipedia.org/wiki/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 IM Used**

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:**

[**302 Found**](https://en.wikipedia.org/wiki/HTTP_302)

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**](https://en.wikipedia.org/wiki/HTTP_303)

The response to the request can be found under another [URI](https://en.wikipedia.org/wiki/Uniform_Resource_Identifier) 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](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#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**

The requested resource is available only through a proxy, the address for which is provided in the response. Many HTTP clients do not correctly handle responses with this status code, primarily for security reasons.

**306 Switch Proxy**

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

**307 Temporary Redirect**

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**

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*. So, for example, submitting a form to a permanently redirected resource may continue smoothly.

**4xx: Client Error**

**400 Bad Request**

The server cannot or will not process the request due to an apparent client error (e.g., malformed request syntax, size too large, invalid request message framing, or deceptive request routing).[[33]](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#cite_note-rfc7231-400-34)

**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. The original intention was that this code might be used as part of some form of [digital cash](https://en.wikipedia.org/wiki/Digital_cash) or [micropayment](https://en.wikipedia.org/wiki/Micropayment) scheme, as proposed for example by GNU Taler, but that has not yet happened, and this code is not usually used. [Google Developers](https://en.wikipedia.org/wiki/Google_Developers) API uses this status if a particular developer has exceeded the daily limit on requests.

[**403 Forbidden**](https://en.wikipedia.org/wiki/HTTP_403)

The request was valid, but the server is refusing action. The user might not have the necessary permissions for a resource, or may need an account of some sort.

[**404 Not Found**](https://en.wikipedia.org/wiki/HTTP_404)

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](https://en.wikipedia.org/wiki/POST_(HTTP)), 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 (**[**RFC 7235**](https://tools.ietf.org/html/rfc7235)**)**

The client must first authenticate itself with the [proxy](https://en.wikipedia.org/wiki/Proxy_server).

**5xx: Server Error :**

**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 (e.g., a new feature of a web-service API).[[61]](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#cite_note-62)

**502 Bad Gateway**

The server was acting as a [gateway](https://en.wikipedia.org/wiki/Gateway_(telecommunications)) or proxy and received an invalid response from the upstream server.

**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.

**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.

List of HTTP Status Codes are shown here:

<https://en.wikipedia.org/wiki/List_of_HTTP_status_codes>