List of HTTP status codes

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 (RFC 9110).

The Internet Assigned Numbers Authority (IANA) maintains the official registry of HTTP status codes.[1]

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).[2]
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.[1]

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.[7] [8]
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.[9]
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.[10]

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.[1]

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.[11]

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 (RFC 1945) required the client to perform a temporary redirect with the same method (the original describing phrase was "Moved Temporarily"),[12] 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.[11]
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.[13]
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.

4xx client errors

404 error on Wikimedia

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 (e.g., malformed request syntax, size too large, invalid request message framing, or deceptive request routing).
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. See Basic access authentication and Digest access authentication. 401 semantically means "unauthorised", the user does not have valid authentication credentials for the target resource.
Some sites incorrectly issue HTTP 401 when an IP address is banned from the website (usually the website domain) and that specific address is refused permission to access a website.
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, as proposed, for example, by GNU Taler,[14] but that has not yet happened, and this code is not widely used. Google Developers API uses this status if a particular developer has exceeded the daily limit on requests.[15] Sipgate uses this code if an account does not have sufficient funds to start a call.[16] Shopify uses this code when the store has not paid their fees and is temporarily disabled.[17] Stripe uses this code for failed payments where parameters were correct, for example blocked fraudulent payments.[18]
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 (e.g. creating a duplicate record where only one is allowed). This code is also typically used if the request provided authentication by answering the WWW-Authenticate header field challenge, but the server did not accept that authentication. The request should not be repeated.

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. See Content negotiation.

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 was previously in use but 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" in RFC 2616.[19]

414 URI Too Long

The URI provided was too long for the server to process. Often the result of too much data being encoded as a query-string of a GET request, in which case it should be converted to a POST request. Called "Request-URI Too Long" previously in RFC 2616.[20]

415 Unsupported Media Type

The request entity has a media type which the server or resource does not support. For example, the client uploads an image as image/svg+xml, but the server requires that images use a different format.

416 Range Not Satisfiable

The client has asked for a portion of the file (byte serving), 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 RFC 2616.[21]

417 Expectation Failed

The server cannot meet the requirements of the Expect request-header field.[22]

418 I'm a teapot (RFC 2324, RFC 7168)

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. The RFC specifies this code should be returned by teapots requested to brew coffee.[23] This HTTP status is used as an Easter egg in some websites, such as Google.com's "I'm a teapot" easter egg.[24] [25] [26] Sometimes, this status code is also used as a response to a blocked request, instead of the more appropriate 403 Forbidden.[27] [28]

421 Misdirected Request

The request was directed at a server that is not able to produce a response (for example because of connection reuse).

422 Unprocessable Entity

The request was well-formed but was unable to be followed due to semantic errors.[9]

423 Locked (WebDAV; RFC 4918)

The resource that is being accessed is locked.[9]

424 Failed Dependency (WebDAV; RFC 4918)

The request failed because it depended on another request and that request failed (e.g., a PROPPATCH).[9]

425 Too Early (RFC 8470)

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 (RFC 6585)
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.[29]
429 Too Many Requests (RFC 6585)
The user has sent too many requests in a given amount of time. Intended for use with rate-limiting schemes.[29]
431 Request Header Fields Too Large (RFC 6585)
The server is unwilling to process the request because either an individual header field, or all the header fields collectively, are too large.[29]
451 Unavailable For Legal Reasons (RFC 7725)
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.[30] The code 451 was chosen as a reference to the novel *Fahrenheit 451* (see the Acknowledgements in the RFC).

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 (e.g., a new feature of a web-service API).
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.[31]
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 version used in the request.
506 Variant Also Negotiates (RFC 2295)
Transparent content negotiation for the request results in a circular reference.[32]
507 Insufficient Storage (WebDAV; RFC 4918)
The server is unable to store the representation needed to complete the request.[9]
508 Loop Detected (WebDAV; RFC 5842)

The server detected an infinite loop while processing the request (sent instead of 208 Already Reported).
510 Not Extended (RFC 2774)
Further extensions to the request are required for the server to fulfil it.[33]
511 Network Authentication Required (RFC 6585)
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).[29]

Unofficial codes

The following codes are not specified by any standard.

218 This is fine (Apache HTTP Server)
Used by Apache servers. A catch-all error condition allowing the passage of message bodies through the server when the `ProxyErrorOverride` setting is enabled. It is displayed in this situation instead of a 4xx or 5xx error message.[34]
419 Page Expired (Laravel Framework)
Used by the Laravel Framework when a CSRF Token is missing or expired.
420 Method Failure (Spring Framework)
A deprecated response used by the Spring Framework when a method has failed.[35]
420 Enhance Your Calm (Twitter)
Returned by version 1 of the Twitter Search and Trends API when the client is being rate limited; versions 1.1 and later use the 429 Too Many Requests response code instead.[36]

The phrase "Enhance your calm" comes from the 1993 movie *Demolition Man*, and its association with this number is likely a reference to cannabis.

430 Request Header Fields Too Large (Shopify)

Used by Shopify, instead of the 429 Too Many Requests response code, when too many URLs are requested within a certain time frame.[37]

450 Blocked by Windows Parental Controls (Microsoft)

The Microsoft extension code indicated when Windows Parental Controls are turned on and are blocking access to the requested webpage.[38]

498 Invalid Token (Esri)

Returned by ArcGIS for Server. Code 498 indicates an expired or otherwise invalid token.[39]

499 Token Required (Esri)

Returned by ArcGIS for Server. Code 499 indicates that a token is required but was not submitted.[39]

509 Bandwidth Limit Exceeded (Apache Web Server/cPanel)

The server has exceeded the bandwidth specified by the server administrator; this is often used by shared hosting providers to limit the bandwidth of customers.[40]

529 Site is overloaded

Used by Qualys in the SSLLabs server testing API to signal that the site can't process the request.[41]

530 Site is frozen

Used by the Pantheon Systems web platform to indicate a site that has been frozen due to inactivity.[42]

598 (Informal convention) Network read timeout error

Used by some HTTP proxies to signal a network read timeout behind the proxy to a client in front of the proxy.[43]
599 Network Connect Timeout Error
An error used by some HTTP proxies to signal a network connect timeout behind the proxy to a client in front of the proxy.

Internet Information Services

Microsoft's Internet Information Services (IIS) web server expands the 4xx error space to signal errors with the client's request.

440 Login Time-out
The client's session has expired and must log in again.[44]
449 Retry With
The server cannot honour the request because the user has not provided the required information.[45]
451 Redirect
Used in Exchange ActiveSync when either a more efficient server is available or the server cannot access the users' mailbox.[46] The client is expected to re-run the HTTP AutoDiscover operation to find a more appropriate server.[47]

IIS sometimes uses additional decimal sub-codes for more specific information,[48] however these sub-codes only appear in the response payload and in documentation, not in the place of an actual HTTP status code.

nginx

The nginx web server software expands the 4xx error space to signal issues with the client's request.[49] [50]

444 No Response
Used internally[51] to instruct the server to return no information to the client and close the connection immediately.
494 Request header too large
Client sent too large request or too long header line.
495 SSL Certificate Error
An expansion of the 400 Bad Request response code, used when the client has provided an invalid client certificate.
496 SSL Certificate Required
An expansion of the 400 Bad Request response code, used when a client certificate is required but not provided.
497 HTTP Request Sent to HTTPS Port
An expansion of the 400 Bad Request response code, used when the client has made a HTTP request to a port listening for HTTPS requests.
499 Client Closed Request
Used when the client has closed the request before the server could send a response.

Cloudflare

Cloudflare's reverse proxy service expands the 5xx series of errors space to signal issues with the origin server.[52]

520 Web Server Returned an Unknown Error
The origin server returned an empty, unknown, or unexpected response to Cloudflare.[53]
521 Web Server Is Down
The origin server refused connections from Cloudflare. Security solutions at the origin may be blocking legitimate connections from certain Cloudflare IP addresses.
522 Connection Timed Out
Cloudflare timed out contacting the origin server.
523 Origin Is Unreachable
Cloudflare could not reach the origin server; for example, if the DNS records for the origin server are incorrect or missing.
524 A Timeout Occurred
Cloudflare was able to complete a TCP connection to the origin server, but did not receive a timely HTTP response.
525 SSL Handshake Failed
Cloudflare could not negotiate a SSL/TLS handshake with the origin server.
526 Invalid SSL Certificate
Cloudflare could not validate the SSL certificate on the origin web server. Also used by Cloud Foundry's gorouter.
527 Railgun Error
Error 527 indicates an interrupted connection between Cloudflare and the origin server's Railgun server.[54]
530
Error 530 is returned along with a 1xxx error.[55]

AWS Elastic Load Balancing

Amazon Web Services' Elastic Load Balancing adds a few custom return codes to signal issues either with the client request or with the origin server.[56]

460
Client closed the connection with the load balancer before the idle timeout period elapsed. Typically when client timeout is sooner than the Elastic Load Balancer's timeout.[56]
463
The load balancer received an X-Forwarded-For request header with more than 30 IP addresses.[56]
464
Incompatible protocol versions between Client and Origin server.[56]
561 Unauthorized
An error around authentication returned by a server registered with a load balancer. You configured a listener rule to authenticate users, but the identity provider (IdP) returned an error code when authenticating the user.[56]

Caching warning codes (obsoleted)

The following caching related warning codes were specified under RFC 7234. Unlike the other status codes above, these were not sent as the response status in the HTTP protocol, but as part of the "Warning" HTTP header.[57] [58]

Since this "Warning" header is often neither sent by servers nor acknowledged by clients, this header and its codes were obsoleted by the HTTP Working Group in 2022 with RFC 9111.[59]

110 Response is Stale
The response provided by a cache is stale (the content's age exceeds a maximum age set by a Cache-Control header or heuristically chosen lifetime).
111 Revalidation Failed
The cache was unable to validate the response, due to an inability to reach the origin server.
112 Disconnected Operation
The cache is intentionally disconnected from the rest of the network.
113 Heuristic Expiration
The cache heuristically chose a freshness lifetime greater than 24 hours and the response's age is greater than 24 hours.
199 Miscellaneous Warning
Arbitrary, non-specific warning. The warning text may be logged or presented to the user.
214 Transformation Applied
Added by a proxy if it applies any transformation to the representation, such as changing the content encoding, media type or the like.
299 Miscellaneous Persistent Warning
Same as 199, but indicating a persistent warning.

See also

- Custom error pages
- List of FTP server return codes
- List of HTTP header fields
- List of SMTP server return codes
- Common Log Format

Notes

1. ^ Emphasised words and phrases such as *must* and *should* represent interpretation guidelines as given by RFC 2119

References

1. ^ Jump up to: **a b c** *"Hypertext Transfer Protocol (HTTP) Status Code Registry"*. *Iana.org*. *Archived from the original on December 11, 2011. Retrieved January 8, 2015.*
2. ^ *Fielding, Roy T. "RFC 9110: HTTP Semantics and Content, Section 10.1.1 "Expect""*.
3. ^ *Goland, Yaronn; Whitehead, Jim; Faizi, Asad; Carter, Steve R.; Jensen, Del (February 1999). HTTP Extensions for Distributed Authoring – WEBDAV. IETF. doi:10.17487/RFC2518. RFC 2518. Retrieved October 24, 2009.*
4. ^ *"102 Processing - HTTP MDN"*. 102 status code is deprecated
5. ^ *Oku, Kazuho (December 2017). An HTTP Status Code for Indicating Hints. IETF. doi:10.17487/RFC8297. RFC 8297. Retrieved December 20, 2017.*

6. ^ Stewart, Mark; djna. "Create request with POST, which response codes 200 or 201 and content". Stack Overflow. Archived from the original on October 11, 2016. Retrieved October 16, 2015.
7. ^ "RFC 9110: HTTP Semantics and Content, Section 15.3.4".
8. ^ "RFC 9110: HTTP Semantics and Content, Section 7.7".
9. ^ Jump up to: a b c d e Dusseault, Lisa, ed. (June 2007). HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV). IETF. doi:10.17487/RFC4918. RFC 4918. Retrieved October 24, 2009.
10. ^ Delta encoding in HTTP. IETF. January 2002. doi:10.17487/RFC3229. RFC 3229. Retrieved February 25, 2011.
11. ^ Jump up to: a b "RFC 9110: HTTP Semantics and Content, Section 15.4 "Redirection 3xx"".
12. ^ Berners-Lee, Tim; Fielding, Roy T.; Nielsen, Henrik Frystyk (May 1996). Hypertext Transfer Protocol – HTTP/1.0. IETF. doi:10.17487/RFC1945. RFC 1945. Retrieved October 24, 2009.
13. ^ "Mozilla Bugzilla Bug 187996: Strange behavior on 305 redirect, comment 13". March 3, 2003. Archived from the original on April 21, 2014. Retrieved May 21, 2009.
14. ^ "The GNU Taler tutorial for PHP Web shop developers 0.4.0". docs.taler.net. Archived from the original on November 8, 2017. Retrieved October 29, 2017.
15. ^ "Google API Standard Error Responses". 2016. Archived from the original on May 25, 2017. Retrieved June 21, 2017.
16. ^ "Sipgate API Documentation". Archived from the original on July 10, 2018. Retrieved July 10, 2018.
17. ^ "Shopify Documentation". Archived from the original on July 25, 2018. Retrieved July 25, 2018.
18. ^ "Stripe API Reference – Errors". stripe.com. Retrieved October 28, 2019.

19. ^ *"RFC2616 on status 413"*. Tools.ietf.org. *Archived from the original on March 7, 2011. Retrieved November 11, 2015.*
20. ^ *"RFC2616 on status 414"*. Tools.ietf.org. *Archived from the original on March 7, 2011. Retrieved November 11, 2015.*
21. ^ *"RFC2616 on status 416"*. Tools.ietf.org. *Archived from the original on March 7, 2011. Retrieved November 11, 2015.*
22. ^ *TheDeadLike. "HTTP/1.1 Status Codes 400 and 417, cannot choose which". serverFault. Archived from the original on October 10, 2015. Retrieved October 16, 2015.*
23. ^ *Larry Masinter (April 1, 1998). Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0). doi:10.17487/RFC2324. RFC 2324. "Any attempt to brew coffee with a teapot should result in the error code "418 I'm a teapot". The resulting entity body MAY be short and stout."*
24. ^ I'm a teapot
25. ^ *Barry Schwartz (August 26, 2014). "New Google Easter Egg For SEO Geeks: Server Status 418, I'm A Teapot". Search Engine Land. Archived from the original on November 15, 2015. Retrieved November 4, 2015.*
26. ^ *"Google's Teapot". Retrieved October 23, 2017.*
27. ^ *"Enable extra web security on a website". DreamHost. Retrieved December 18, 2022.*
28. ^ *"I Went to a Russian Website and All I Got Was This Lousy Teapot". PCMag. Retrieved December 18, 2022.*
29. ^ Jump up to: ***a b c d*** *Nottingham, M.; Fielding, R. (April 2012). "RFC 6585 – Additional HTTP Status Codes". Request for Comments. Internet Engineering Task Force. Archived from the original on May 4, 2012. Retrieved May 1, 2012.*
30. ^ *Bray, T. (February 2016). "An HTTP Status Code to Report Legal Obstacles". ietf.org. Archived from the original on March 4, 2016. Retrieved March 7, 2015.*

31. ^ alex. "What is the correct HTTP status code to send when a site is down for maintenance?". Stack Overflow. Archived from the original on October 11, 2016. Retrieved October 16, 2015.
32. ^ Holtman, Koen; Mutz, Andrew H. (March 1998). Transparent Content Negotiation in HTTP. IETF. doi:10.17487/RFC2295. RFC 2295. Retrieved October 24, 2009.
33. ^ Nielsen, Henrik Frystyk; Leach, Paul; Lawrence, Scott (February 2000). An HTTP Extension Framework. IETF. doi:10.17487/RFC2774. RFC 2774. Retrieved October 24, 2009.
34. ^ "218 This is fine - HTTP status code explained". HTTP.dev. Retrieved July 25, 2023.
35. ^ "Enum HttpStatus". Spring Framework. org.springframework.http. Archived from the original on October 25, 2015. Retrieved October 16, 2015.
36. ^ "Twitter Error Codes & Responses". Twitter. 2014. Archived from the original on September 27, 2017. Retrieved January 20, 2014.
37. ^ "HTTP Status Codes and SEO: what you need to know". ContentKing. Retrieved August 9, 2019.
38. ^ "Screenshot of error page". Archived from the original (bmp) on May 11, 2013. Retrieved October 11, 2009.
39. ^ Jump up to: **a** **b** "Using token-based authentication". ArcGIS Server SOAP SDK. Archived from the original on September 26, 2014. Retrieved September 8, 2014.
40. ^ "HTTP Error Codes and Quick Fixes". Docs.cpanel.net. Archived from the original on November 23, 2015. Retrieved October 15, 2015.
41. ^ "SSL Labs API v3 Documentation". github.com.
42. ^ "Platform Considerations | Pantheon Docs". pantheon.io. Archived from the original on January 6, 2017. Retrieved January 5, 2017.

43. ^ *"HTTP status codes - ascii-code.com"*. *www.ascii-code.com*. *Archived from the original on January 7, 2017. Retrieved December 23, 2016.*
44. ^ *"Error message when you try to log on to Exchange 2007 by using Outlook Web Access: "440 Login Time-out""*. *Microsoft*. *2010. Retrieved November 13, 2013.*
45. ^ *"2.2.6 449 Retry With Status Code"*. *Microsoft*. *2009. Archived from the original on October 5, 2009. Retrieved October 26, 2009.*
46. ^ *"MS-ASCMD, Section 3.1.5.2.2"*. *Msdn.microsoft.com. Archived from the original on March 26, 2015. Retrieved January 8, 2015.*
47. ^ *"Ms-oxdisco"*. *Msdn.microsoft.com. Archived from the original on July 31, 2014. Retrieved January 8, 2015.*
48. ^ *"The HTTP status codes in IIS 7.0"*. *Microsoft*. *July 14, 2009. Archived from the original on April 9, 2009. Retrieved April 1, 2009.*
49. ^ *"ngx_http_request.h"*. *nginx 1.9.5 source code. nginx inc. Archived from the original on September 19, 2017. Retrieved January 9, 2016.*
50. ^ *"ngx_http_special_response.c"*. *nginx 1.9.5 source code. nginx inc. Archived from the original on May 8, 2018. Retrieved January 9, 2016.*
51. ^ "return" directive Archived March 1, 2018, at the Wayback Machine (http_rewrite module) documentation.
52. ^ *"Troubleshooting: Error Pages"*. *Cloudflare*. *Archived from the original on March 4, 2016. Retrieved January 9, 2016.*
53. ^ *"Error 520: web server returns an unknown error"*. *Cloudflare*.
54. ^ *"527 Error: Railgun Listener to origin error"*. *Cloudflare*. *Archived from the original on October 13, 2016. Retrieved October 12, 2016.*
55. ^ *"Error 530"*. *Cloudflare*. *Retrieved November 1, 2019.*
56. ^ Jump up to: ***a b c d e*** *"Troubleshoot Your Application Load Balancers – Elastic Load Balancing"*. *docs.aws.amazon.com. Retrieved May 17, 2023.*

57. ^ *"Hypertext Transfer Protocol (HTTP/1.1): Caching"*. *datatracker.ietf.org. Retrieved September 25, 2021.*
58. ^ *"Warning - HTTP | MDN"*. *developer.mozilla.org. Retrieved August 15, 2021.*
 Some text was copied from this source, which is available under a Creative Commons Attribution-ShareAlike 2.5 Generic (CC BY-SA 2.5) license.
59. ^ *"RFC 9111: HTTP Caching, Section 5.5 "Warning""*. *June 2022.*

External links


Wikimedia Commons has media related to HTTP status codes.

- *"RFC 9110: HTTP Semantics and Content, Section 15 "Status Codes""*.
- Hypertext Transfer Protocol (HTTP) Status Code Registry at the *Internet Assigned Numbers Authority*
- HTTP status codes at *http-statuscode.com*
- MDN status code reference at *mozilla.org*

-