HTTP Status Codes Cheatsheet

By Dejan Panovski • Updated on Nov 8, 2026

HTTP status codes from 1xx to 5xx, common error causes, Nginx and Cloudflare codes, and curl and log commands for checking responses.

Use this cheatsheet to identify HTTP response codes, trace common server errors, and check responses with curl or access logs. The Nginx and curl examples were tested with Nginx 1.28 and curl 8.18 on Ubuntu 26.04.

Status Code Classes

Clients must understand the class even when they do not know the exact code.

`1xx Informational`	Interim response; the request is still in progress
`2xx Success`	The server received and accepted the request
`3xx Redirection`	The client must take another step, usually follow `Location`
`4xx Client error`	The server sees a problem with the request; some codes, such as `408` and `429`, allow a later retry
`5xx Server error`	The request looked valid, but the server or an upstream failed
`Unknown code`	Treated as the `x00` code of its class, so `599` acts as `500`
`Reason phrase`	Text after the code; servers word it differently (Nginx sends `503 Service Temporarily Unavailable`), and HTTP/2 omits it

1xx and 2xx Codes

Informational and success responses.

`100 Continue`	Headers accepted; send the body (reply to `Expect: 100-continue`)
`101 Switching Protocols`	The connection switches protocol, for example to WebSocket
`103 Early Hints`	Early `Link` headers so the browser can preload assets
`200 OK`	Request succeeded; the body holds the result
`201 Created`	A new resource was created; `Location` usually points to it
`202 Accepted`	Request queued; the work is not finished yet
`204 No Content`	Success with no body, common for `DELETE` and API updates
`206 Partial Content`	Range response, used by resumed downloads (`curl -C -`, `wget -c`)

3xx Redirection

Redirects usually provide the new address in `Location`; `304` tells the client to reuse its cached copy.

`301 Moved Permanently`	Permanent move; a strong signal for search engines to use the target URL
`302 Found`	Temporary move; does not signal a permanent URL change
`303 See Other`	Fetch the result with `GET`, used after a form `POST`
`304 Not Modified`	The cached copy is still valid (`If-None-Match` or `If-Modified-Since` matched); no body
`307 Temporary Redirect`	Like `302`, but the client must repeat the same method and body
`308 Permanent Redirect`	Like `301`, but the client must repeat the same method and body
`POST to a 301 or 302`	Browsers and `curl -L` switch to `GET` and drop the body; use `307` or `308` for APIs

4xx: Common Client Errors

Check the request first, then the server rules that matched it.

`400 Bad Request`	Malformed request: bad syntax, invalid JSON, or headers larger than the server buffers
`401 Unauthorized`	Missing or wrong credentials; the response includes `WWW-Authenticate`
`403 Forbidden`	Access refused: file permissions, a `deny` rule, or a directory with no index file
`404 Not Found`	Nothing at that path: wrong URL, wrong `root` or `alias`, or a missing rewrite or `try_files` rule
`405 Method Not Allowed`	The URL does not accept that method; Nginx returns it for a `POST` to a static file
`408 Request Timeout`	The client sent the request too slowly; Nginx `client_header_timeout` and `client_body_timeout` (60s default)
`410 Gone`	The resource is no longer available and its removal is likely permanent

4xx: Limits and API Errors

Size limits, rate limits, and request validation.

`409 Conflict`	The request conflicts with the current state, such as an edit to an outdated version
`413 Content Too Large`	Body over the server limit; Nginx calls it `Request Entity Too Large` and sets it with `client_max_body_size` (`1m` default); see the 413 guide
`414 URI Too Long`	Request line over the limit, usually a huge query string; Nginx `large_client_header_buffers`
`415 Unsupported Media Type`	Wrong `Content-Type`, for example JSON sent without `application/json`
`422 Unprocessable Content`	Valid syntax that failed validation; common in REST APIs
`429 Too Many Requests`	Rate limit reached; honor `Retry-After` if present. Nginx `limit_req` sends `503` unless `limit_req_status 429;` is set
`431 Request Header Fields Too Large`	Headers or cookies too large; Nginx answers this case with a `400` instead
`451 Unavailable For Legal Reasons`	Blocked for legal reasons, such as a court order

5xx Server Errors

The server error log usually names the cause.

`500 Internal Server Error`	Unhandled error in the application or config: a PHP fatal error, a bad `.htaccess` rule, wrong file permissions; see the 500 guide
`501 Not Implemented`	The server does not support the request method at all
`502 Bad Gateway`	The proxy got no valid response from the upstream: backend stopped, wrong port or socket, PHP-FPM down. Nginx logs `connect() failed (111: Connection refused)`
`503 Service Unavailable`	Overload, maintenance, rate limiting, or no healthy backend; may send `Retry-After`; see the 503 guide
`504 Gateway Timeout`	The upstream did not answer in time; Nginx `proxy_read_timeout` or `fastcgi_read_timeout` (60s default) and the log line `upstream timed out (110: Connection timed out)`
`505 HTTP Version Not Supported`	The server rejects the HTTP version in the request
`507 Insufficient Storage`	A WebDAV server, such as Nextcloud, is out of space or over quota
`511 Network Authentication Required`	A captive portal (hotel or airport Wi-Fi) wants you to sign in

Nginx-Specific Codes

Nginx uses these codes internally. By default, codes `494` to `497` reach the client and the access log as `400`; an `error_page` rule can change the response. Codes `444` and `499` appear only in the access log.

444	`return 444;` closes the connection without a response; curl prints `curl: (52) Empty reply from server`
494	Request header or cookie too large; the page reads `400 Request Header Or Cookie Too Large`; see the fix guide
495	The client certificate failed verification (`ssl_verify_client`)
496	A client certificate is required but was not sent; the page reads `400 No required SSL certificate was sent`
497	Plain HTTP sent to an HTTPS port; the page reads `400 The plain HTTP request was sent to HTTPS port`
499	The client closed the connection before Nginx answered, often a browser or `curl -m` giving up on a slow upstream

Cloudflare Origin Errors

Cloudflare returns these codes when it cannot get a usable response from your origin server. Check origin availability, firewall rules, DNS, and Cloudflare settings.

520	The origin returned an empty, unknown, or unexpected response
521	The origin refused the connection: web server down, or a firewall blocking Cloudflare IP ranges
522	Connection to the origin timed out: no `SYN+ACK` within 19 seconds, or no `ACK` within 90 seconds
523	Origin unreachable: wrong DNS record or a routing problem
524	The origin connected but exceeded the default 125-second read timeout or the 30-second write timeout
525	The SSL handshake between Cloudflare and the origin failed (Full or Full (strict) mode)
526	Cloudflare cannot verify the origin certificate (Full (strict) mode)
530	Cloudflare cannot resolve the origin hostname; the page also shows a `1xxx` error

Check a Status Code with curl

Replace `URL` with the address you are testing. The `-sS` options hide progress but keep errors visible; `000` means no HTTP status was received.

`curl -I URL`	Show the status line and headers (sends `HEAD`)
`curl -sS -o /dev/null -D - URL`	Show headers from a real `GET`, for apps that answer `HEAD` differently
`curl -sS -o /dev/null -w '%{http_code}\n' URL`	Print only the status code
`curl -sS -o /dev/null -w '%{redirect_url}\n' URL`	Print the redirect target without following it
`curl -sSIL URL`	Follow redirects and show the headers of every hop
`curl -sSL -o /dev/null -w '%{http_code} %{num_redirects} %{url_effective}\n' URL`	Final code, redirect count, and final URL
`curl -fsS URL`	Exit with code 22 on a `4xx` or `5xx` and print `curl: (22) The requested URL returned error: 404`
`wget --spider -S URL`	Show response headers without downloading; exits with code 8 on a `4xx` or `5xx`

Find Status Codes in Logs

In the default combined log format, field `$9` is the status code. A malformed request line without the protocol shifts the fields.

`sudo awk '{print $9}' /var/log/nginx/access.log | sort | uniq -c | sort -rn`	Count responses by status code
`sudo awk '$9 ~ /^5/' /var/log/nginx/access.log | tail`	Show the latest `5xx` responses
`sudo awk '$9 == 404 {print $7}' /var/log/nginx/access.log | sort | uniq -c | sort -rn | head`	List the most requested missing URLs
`sudo tail -f /var/log/nginx/error.log`	Watch the error log for the cause behind `5xx` responses
`/var/log/apache2/`	Apache logs on Ubuntu and Debian
`/var/log/httpd/`	Apache logs on RHEL, Rocky, and Fedora

Send a Status Code from the Server

Nginx directives go in a `server` or `location` block; Apache directives go in a virtual host or `.htaccess`.

`return 301 https://$host$request_uri;`	Nginx: redirect to HTTPS permanently
`return 410;`	Nginx: mark a removed URL as gone
`add_header Retry-After 120 always; return 503;`	Nginx: maintenance response with a retry hint
`error_page 404 /404.html;`	Nginx: serve a custom error page
`error_page 497 =301 https://$host:$server_port$request_uri;`	Nginx: redirect plain HTTP sent to an HTTPS port
`limit_req_status 429;`	Nginx: answer rate-limited requests with `429` instead of `503`
`Redirect 301 /old /new`	Apache: permanent redirect
`Redirect gone /old`	Apache: return `410`
`ErrorDocument 404 /404.html`	Apache: serve a custom error page

Related Guides

Use these guides to fix specific errors and configure redirects.

500 Internal Server Error	Causes and fixes for Apache, Nginx, and PHP
503 Service Unavailable	Overload, maintenance, and backend failures
Nginx 413 Request Entity Too Large	Raise the upload size limit
Fix Nginx “Request Header or Cookie Too Large”	Clear cookies or raise the header buffers
Redirect HTTP to HTTPS in Nginx	Permanent redirects with `return 301`
Nginx Error and Access Logs	Log locations, formats, and levels
curl Command in Linux	Requests, headers, and redirects from the terminal
Nginx Cheatsheet	Server blocks, proxying, and logging at a glance
curl Cheatsheet	Common curl options and requests