NAME

libcurl-errors - error codes in libcurl

DESCRIPTION

This man page includes most, if not all, available error codes in libcurl. Why they occur and possibly what you can do to fix the problem are also included.

CURLcode

Almost all "easy" interface functions return a CURLcode error code. No matter what, using the *curl_easy_setopt(3)* option *CURLOPT_ERRORBUFFER(3)* is a good idea as it gives you a human readable error string that may offer more details about the cause of the error than just the error code. *curl_easy_strerror(3)* can be called to get an error string from a given CURLcode number.

CURLcode is one of the following:

CURLE_OK (0)

All fine. Proceed as usual.

CURLE_UNSUPPORTED_PROTOCOL (1)

The URL you passed to libcurl used a protocol that this libcurl does not support. The support might be a compile-time option that you did not use, it can be a misspelled protocol string or just a protocol libcurl has no code for.

CURLE_FAILED_INIT (2)

Early initialization code failed. This is likely to be an internal error or problem, or a resource problem where something fundamental could not get done at init time.

CURLE_URL_MALFORMAT (3)

The URL was not properly formatted.

CURLE_NOT_BUILT_IN (4)

A requested feature, protocol or option was not found built-in in this libcurl due to a build-time decision. This means that a feature or option was not enabled or explicitly disabled when libcurl was built and in order to get it to function you have to get a rebuilt libcurl.

CURLE_COULDNT_RESOLVE_PROXY (5)

Could not resolve proxy. The given proxy host could not be resolved.

CURLE_COULDNT_RESOLVE_HOST (6)

Could not resolve host. The given remote host was not resolved.

CURLE_COULDNT_CONNECT (7)
Failed to connect() to host or proxy.

CURLE_WEIRD_SERVER_REPLY (8)
The server sent data libcurl could not parse. This error code was known as *CURLE_FTP_WEIRD_SERVER_REPLY* before 7.51.0.

CURLE_REMOTE_ACCESS_DENIED (9)
We were denied access to the resource given in the URL. For FTP, this occurs while trying to change to the remote directory.

CURLE_FTP_ACCEPT_FAILED (10)
While waiting for the server to connect back when an active FTP session is used, an error code was sent over the control connection or similar.

CURLE_FTP_WEIRD_PASS_REPLY (11)
After having sent the FTP password to the server, libcurl expects a proper reply. This error code indicates that an unexpected code was returned.

CURLE_FTP_ACCEPT_TIMEOUT (12)
During an active FTP session while waiting for the server to connect, the *CURLOPT_ACCEPTTIMEOUT_MS(3)* (or the internal default) timeout expired.

CURLE_FTP_WEIRD_PASV_REPLY (13)
libcurl failed to get a sensible result back from the server as a response to either a PASV or a EPSV command. The server is flawed.

CURLE_FTP_WEIRD_227_FORMAT (14)
FTP servers return a 227-line as a response to a PASV command. If libcurl fails to parse that line, this return code is passed back.

CURLE_FTP_CANT_GET_HOST (15)
An internal failure to lookup the host used for the new connection.

CURLE_HTTP2 (16)
A problem was detected in the HTTP2 framing layer. This is somewhat generic and can be one out of several problems, see the error buffer for details.

CURLE_FTP_COULDNT_SET_TYPE (17)
Received an error when trying to set the transfer mode to binary or ASCII.

CURLE_PARTIAL_FILE (18)
A file transfer was shorter or larger than expected. This happens when the server first reports an expected transfer size, and then delivers data that does not match the previously given size.

CURLE_FTP_COULDNT_RETR_FILE (19)
This was either a weird reply to a ’RETR’ command or a zero byte transfer complete.

Obsolete error (20)
Not used in modern versions.

CURLE_QUOTE_ERROR (21)
When sending custom "QUOTE" commands to the remote server, one of the commands returned an error code that was 400 or higher (for FTP) or otherwise indicated unsuccessful completion of the command.

CURLE_HTTP_RETURNED_ERROR (22)
This is returned if *CURLOPT_FAILONERROR(3)* is set TRUE and the HTTP server returns an error code that is >= 400.

CURLE_WRITE_ERROR (23)
An error occurred when writing received data to a local file, or an error was returned to libcurl from a write callback.

Obsolete error (24)
Not used in modern versions.

CURLE_UPLOAD_FAILED (25)
Failed starting the upload. For FTP, the server typically denied the STOR command. The error buffer usually contains the server’s explanation for this.

CURLE_READ_ERROR (26)
There was a problem reading a local file or an error returned by the read callback.

CURLE_OUT_OF_MEMORY (27)
A memory allocation request failed. This is serious badness and things are severely screwed up if this ever occurs.

CURLE_OPERATION_TIMEDOUT (28)
Operation timeout. The specified time-out period was reached according to the conditions.

Obsolete error (29)
: Not used in modern versions.

CURLE_FTP_PORT_FAILED (30)
: The FTP PORT command returned error. This mostly happens when you have not specified a good enough address for libcurl to use. See *CURLOPT_FTPPORT(3)*.

CURLE_FTP_COULDNT_USE_REST (31)
: The FTP REST command returned error. This should never happen if the server is sane.

Obsolete error (32)
: Not used in modern versions.

CURLE_RANGE_ERROR (33)
: The server does not support or accept range requests.

CURLE_HTTP_POST_ERROR (34)
: This is an odd error that mainly occurs due to internal confusion.

CURLE_SSL_CONNECT_ERROR (35)
: A problem occurred somewhere in the SSL/TLS handshake. You really want the error buffer and read the message there as it pinpoints the problem slightly more. Could be certificates (file formats, paths, permissions), passwords, and others.

CURLE_BAD_DOWNLOAD_RESUME (36)
: The download could not be resumed because the specified offset was out of the file boundary.

CURLE_FILE_COULDNT_READ_FILE (37)
: A file given with FILE:// could not be opened. Most likely because the file path does not identify an existing file. Did you check file permissions?

CURLE_LDAP_CANNOT_BIND (38)
: LDAP cannot bind. LDAP bind operation failed.

CURLE_LDAP_SEARCH_FAILED (39)
: LDAP search failed.

Obsolete error (40)
: Not used in modern versions.

CURLE_FUNCTION_NOT_FOUND (41)
: Function not found. A required zlib function was not found.

CURLE_ABORTED_BY_CALLBACK (42)
: Aborted by callback. A callback returned "abort" to libcurl.

CURLE_BAD_FUNCTION_ARGUMENT (43)
: A function was called with a bad parameter.

Obsolete error (44)
: Not used in modern versions.

CURLE_INTERFACE_FAILED (45)
: Interface error. A specified outgoing interface could not be used. Set which interface to use for outgoing connections' source IP address with *CURLOPT_INTERFACE(3)*.

Obsolete error (46)
: Not used in modern versions.

CURLE_TOO_MANY_REDIRECTS (47)
: Too many redirects. When following redirects, libcurl hit the maximum amount. Set your limit with *CURLOPT_MAXREDIRS(3)*.

CURLE_UNKNOWN_OPTION (48)
: An option passed to libcurl is not recognized/known. Refer to the appropriate documentation. This is most likely a problem in the program that uses libcurl. The error buffer might contain more specific information about which exact option it concerns.

CURLE_SETOPT_OPTION_SYNTAX (49)
: An option passed in to a setopt was wrongly formatted. See error message for details about what option.

Obsolete errors (50-51)
: Not used in modern versions.

CURLE_GOT_NOTHING (52)
: Nothing was returned from the server, and under the circumstances, getting nothing is considered an error.

CURLE_SSL_ENGINE_NOTFOUND (53)

The specified crypto engine was not found.

CURLE_SSL_ENGINE_SETFAILED (54)
: Failed setting the selected SSL crypto engine as default.

CURLE_SEND_ERROR (55)
: Failed sending network data.

CURLE_RECV_ERROR (56)
: Failure with receiving network data.

Obsolete error (57)
: Not used in modern versions.

CURLE_SSL_CERTPROBLEM (58)
: problem with the local client certificate.

CURLE_SSL_CIPHER (59)
: Could not use specified cipher.

CURLE_PEER_FAILED_VERIFICATION (60)
: The remote server's SSL certificate or SSH fingerprint was deemed not OK. This error code has been unified with CURLE_SSL_CACERT since 7.62.0. Its previous value was 51.

CURLE_BAD_CONTENT_ENCODING (61)
: Unrecognized transfer encoding.

Obsolete error (62)
: Not used in modern versions.

CURLE_FILESIZE_EXCEEDED (63)
: Maximum file size exceeded.

CURLE_USE_SSL_FAILED (64)
: Requested FTP SSL level failed.

CURLE_SEND_FAIL_REWIND (65)
: When doing a send operation curl had to rewind the data to retransmit, but the rewinding operation failed.

CURLE_SSL_ENGINE_INITFAILED (66)
Initiating the SSL Engine failed.

CURLE_LOGIN_DENIED (67)
The remote server denied curl to login (Added in 7.13.1)

CURLE_TFTP_NOTFOUND (68)
File not found on TFTP server.

CURLE_TFTP_PERM (69)
Permission problem on TFTP server.

CURLE_REMOTE_DISK_FULL (70)
Out of disk space on the server.

CURLE_TFTP_ILLEGAL (71)
Illegal TFTP operation.

CURLE_TFTP_UNKNOWNID (72)
Unknown TFTP transfer ID.

CURLE_REMOTE_FILE_EXISTS (73)
File already exists and is not overwritten.

CURLE_TFTP_NOSUCHUSER (74)
This error should never be returned by a properly functioning TFTP server.

Obsolete error (75-76)
Not used in modern versions.

CURLE_SSL_CACERT_BADFILE (77)
Problem with reading the SSL CA cert (path? access rights?)

CURLE_REMOTE_FILE_NOT_FOUND (78)
The resource referenced in the URL does not exist.

CURLE_SSH (79)
An unspecified error occurred during the SSH session.

CURLE_SSL_SHUTDOWN_FAILED (80)

Failed to shut down the SSL connection.

CURLE_AGAIN (81)
Socket is not ready for send/recv. Wait until it is ready and try again. This return code is only returned from *curl_easy_recv(3)* and *curl_easy_send(3)* (Added in 7.18.2)

CURLE_SSL_CRL_BADFILE (82)
Failed to load CRL file (Added in 7.19.0)

CURLE_SSL_ISSUER_ERROR (83)
Issuer check failed (Added in 7.19.0)

CURLE_FTP_PRET_FAILED (84)
The FTP server does not understand the PRET command at all or does not support the given argument. Be careful when using *CURLOPT_CUSTOMREQUEST(3)*, a custom LIST command is sent with the PRET command before PASV as well. (Added in 7.20.0)

CURLE_RTSP_CSEQ_ERROR (85)
Mismatch of RTSP CSeq numbers.

CURLE_RTSP_SESSION_ERROR (86)
Mismatch of RTSP Session Identifiers.

CURLE_FTP_BAD_FILE_LIST (87)
Unable to parse FTP file list (during FTP wildcard downloading).

CURLE_CHUNK_FAILED (88)
Chunk callback reported error.

CURLE_NO_CONNECTION_AVAILABLE (89)
(For internal use only, is never returned by libcurl) No connection available, the session is queued. (added in 7.30.0)

CURLE_SSL_PINNEDPUBKEYNOTMATCH (90)
Failed to match the pinned key specified with *CURLOPT_PINNEDPUBLICKEY(3)*.

CURLE_SSL_INVALIDCERTSTATUS (91)
Status returned failure when asked with *CURLOPT_SSL_VERIFYSTATUS(3)*.

CURLE_HTTP2_STREAM (92)

Stream error in the HTTP/2 framing layer.

CURLE_RECURSIVE_API_CALL (93)
An API function was called from inside a callback.

CURLE_AUTH_ERROR (94)
An authentication function returned an error.

CURLE_HTTP3 (95)
A problem was detected in the HTTP/3 layer. This is somewhat generic and can be one out of several problems, see the error buffer for details.

CURLE_QUIC_CONNECT_ERROR (96)
QUIC connection error. This error may be caused by an SSL library error. QUIC is the protocol used for HTTP/3 transfers.

CURLE_PROXY (97)
Proxy handshake error. *CURLINFO_PROXY_ERROR(3)* provides extra details on the specific problem.

CURLE_SSL_CLIENTCERT (98)
SSL Client Certificate required.

CURLE_UNRECOVERABLE_POLL (99)
An internal call to poll() or select() returned error that is not recoverable.

CURLE_TOO_LARGE (100)
A value or data field grew larger than allowed.

CURLE_ECH_REQUIRED (101)"
ECH was attempted but failed.

CURLMcode

This is the generic return code used by functions in the libcurl multi interface. Also consider *curl_multi_strerror(3)*.

CURLM_CALL_MULTI_PERFORM (-1)
This is not really an error. It means you should call *curl_multi_perform(3)* again without doing select() or similar in between. Before version 7.20.0 (released on February 9 2010) this could be returned by *curl_multi_perform(3)*, but in later versions this return code is never used.

CURLM_OK (0)
: Things are fine.

CURLM_BAD_HANDLE (1)
: The passed-in handle is not a valid *CURLM* handle.

CURLM_BAD_EASY_HANDLE (2)
: An easy handle was not good/valid. It could mean that it is not an easy handle at all, or possibly that the handle already is in use by this or another multi handle.

CURLM_OUT_OF_MEMORY (3)
: You are doomed.

CURLM_INTERNAL_ERROR (4)
: This can only be returned if libcurl bugs. Please report it to us!

CURLM_BAD_SOCKET (5)
: The passed-in socket is not a valid one that libcurl already knows about. (Added in 7.15.4)

CURLM_UNKNOWN_OPTION (6)
: curl_multi_setopt() with unsupported option (Added in 7.15.4)

CURLM_ADDED_ALREADY (7)
: An easy handle already added to a multi handle was attempted to get added a second time. (Added in 7.32.1)

CURLM_RECURSIVE_API_CALL (8)
: An API function was called from inside a callback.

CURLM_WAKEUP_FAILURE (9)
: Wake up is unavailable or failed.

CURLM_BAD_FUNCTION_ARGUMENT (10)
: A function was called with a bad parameter.

CURLM_ABORTED_BY_CALLBACK (11)
: A multi handle callback returned error.

CURLM_UNRECOVERABLE_POLL (12)
: An internal call to poll() or select() returned error that is not recoverable.

CURLSHcode

The "share" interface returns a **CURLSHcode** to indicate when an error has occurred. Also consider *curl_share_strerror(3)*.

CURLSHE_OK (0)
All fine. Proceed as usual.

CURLSHE_BAD_OPTION (1)
An invalid option was passed to the function.

CURLSHE_IN_USE (2)
The share object is currently in use.

CURLSHE_INVALID (3)
An invalid share object was passed to the function.

CURLSHE_NOMEM (4)
Not enough memory was available. (Added in 7.12.0)

CURLSHE_NOT_BUILT_IN (5)
The requested sharing could not be done because the library you use do not have that particular feature enabled. (Added in 7.23.0)

CURLUcode

The URL interface returns a *CURLUcode* to indicate when an error has occurred. Also consider *curl_url_strerror(3)*.

CURLUE_OK (0)
All fine. Proceed as usual.

CURLUE_BAD_HANDLE (1)
An invalid URL handle was passed as argument.

CURLUE_BAD_PARTPOINTER (2)
An invalid ’part’ argument was passed as argument.

CURLUE_MALFORMED_INPUT (3)
A malformed input was passed to a URL API function.

CURLUE_BAD_PORT_NUMBER (4)

The port number was not a decimal number between 0 and 65535.

CURLUE_UNSUPPORTED_SCHEME (5)
: This libcurl build does not support the given URL scheme.

CURLUE_URLDECODE (6)
: URL decode error, most likely because of rubbish in the input.

CURLUE_OUT_OF_MEMORY (7)
: A memory function failed.

CURLUE_USER_NOT_ALLOWED (8)
: Credentials was passed in the URL when prohibited.

CURLUE_UNKNOWN_PART (9)
: An unknown part ID was passed to a URL API function.

CURLUE_NO_SCHEME (10)
: There is no scheme part in the URL.

CURLUE_NO_USER (11)
: There is no user part in the URL.

CURLUE_NO_PASSWORD (12)
: There is no password part in the URL.

CURLUE_NO_OPTIONS (13)
: There is no options part in the URL.

CURLUE_NO_HOST (14)
: There is no host part in the URL.

CURLUE_NO_PORT (15)
: There is no port part in the URL.

CURLUE_NO_QUERY (16)
: There is no query part in the URL.

CURLUE_NO_FRAGMENT (17)
: There is no fragment part in the URL.

CURLUE_NO_ZONEID (18)
 There is no zone id set in the URL.

CURLUE_BAD_FILE_URL (19)
 The file:// URL is invalid.

CURLUE_BAD_FRAGMENT (20)
 The fragment part of the URL contained bad or invalid characters.

CURLUE_BAD_HOSTNAME (21)
 The hostname contained bad or invalid characters.

CURLUE_BAD_IPV6 (22)
 The IPv6 address hostname contained bad or invalid characters.

CURLUE_BAD_LOGIN (23)
 The login part of the URL contained bad or invalid characters.

CURLUE_BAD_PASSWORD (24)
 The password part of the URL contained bad or invalid characters.

CURLUE_BAD_PATH (25)
 The path part of the URL contained bad or invalid characters.

CURLUE_BAD_QUERY (26)
 The query part of the URL contained bad or invalid characters.

CURLUE_BAD_SCHEME (27)
 The scheme part of the URL contained bad or invalid characters.

CURLUE_BAD_SLASHES (28)
 The URL contained an invalid number of slashes.

CURLUE_BAD_USER (29)
 The user part of the URL contained bad or invalid characters.

CURLUE_LACKS_IDN (30)
 libcurl lacks IDN support.

CURLUE_TOO_LARGE (31)

A value or data field is larger than allowed.

CURLHcode

The header interface returns a *CURLHcode* to indicate when an error has occurred.

CURLHE_OK (0)
: All fine. Proceed as usual.

CURLHE_BADINDEX (1)
: There is no header with the requested index.

CURLHE_MISSING (2)
: No such header exists.

CURLHE_NOHEADERS (3)
: No headers at all have been recorded.

CURLHE_NOREQUEST (4)
: There was no such request number.

CURLHE_OUT_OF_MEMORY (5)
: Out of resources

CURLHE_BAD_ARGUMENT (6)
: One or more of the given arguments are bad.

CURLHE_NOT_BUILT_IN (7)
: HTTP support or the header API has been disabled in the build.

SEE ALSO

CURLOPT_DEBUGFUNCTION(3), **CURLOPT_ERRORBUFFER**(3), **CURLOPT_VERBOSE**(3), **curl_easy_strerror**(3), **curl_multi_strerror**(3), **curl_share_strerror**(3), **curl_url_strerror**(3)