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 will give 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 didn't use, it can be a misspelled protocol string or just a protocol libcurl has no code for.

CURLE_FAILED_INIT (2)

Very early initialization code failed. This is likely to be an internal error or problem, or a resource problem where something fundamental couldn't 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)

Couldn't resolve proxy. The given proxy host could not be resolved.

CURLE_COULDNT_RESOLVE_HOST (6)

Couldn't resolve host. The given remote host was not resolved.

CURLE_COULDNT_CONNECT (7)

Failed to connect() to host or proxy.

CURLE_FTP_WEIRD_SERVER_REPLY (8)

After connecting to a FTP server, libcurl expects to get a certain reply back. This error code implies that it got a strange or bad reply. The given remote server is probably not an OK FTP server.

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_ACCEPT-*
>   *TIMEOUT_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 doesn't 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.

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.

CURLE_UPLOAD_FAILED (25)
>   Failed starting the upload. For FTP, the server typically denied the STOR command. The error buf-
>   fer 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.

CURLE_FTP_PORT_FAILED (30)
>   The FTP PORT command returned error. This mostly happens when you haven't 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.

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:// couldn't be opened. Most likely because the file path doesn't 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.

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)
> Internal error. A function was called with a bad parameter.

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)*.

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_TELNET_OPTION_SYNTAX (49)
> A telnet option string was Illegally formatted.

CURLE_PEER_FAILED_VERIFICATION (51)
> The remote server's SSL certificate or SSH md5 fingerprint was deemed not OK.

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 wasn't 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.

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

CURLE_SSL_CIPHER (59)
>   Couldn't use specified cipher.

CURLE_SSL_CACERT (60)
>   Peer certificate cannot be authenticated with known CA certificates.

CURLE_BAD_CONTENT_ENCODING (61)
>   Unrecognized transfer encoding.

CURLE_LDAP_INVALID_URL (62)
>   Invalid LDAP URL.

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 will not be overwritten.

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

CURLE_CONV_FAILED (75)
>   Character conversion failed.

CURLE_CONV_REQD (76)
>   Caller must register conversion callbacks.

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 till it's 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 will be sent with PRET CMD 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, will never be returned by libcurl) No connection available, the session will be 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_OBSOLETE*

These error codes will never be returned. They were used in an old libcurl version and are currently unused.

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 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 isn't an easy handle at all, or possibly that the handle already is in used 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)

CURLSHcode

The "share" interface will return 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 don't have that particular feature enabled. (Added in 7.23.0)

SEE ALSO

curl_easy_strerror(3), **curl_multi_strerror**(3), **curl_share_strerror**(3), **CURLOPT_ERROR-BUFFER**(3), **CURLOPT_VERBOSE**(3), **CURLOPT_DEBUGFUNCTION**(3)