

**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.

**CURLcode**

Almost all "easy" interface functions return a CURLcode error code. No matter what, using *CURLOPT\_ERRORBUFFER* is a good idea as it will give you a human readable error string that may offer more details about the error cause than just the error code does.

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.

**CURLE\_URL\_MALFORMAT (3)**

The URL was not properly formatted.

**CURLE\_URL\_MALFORMAT\_USER (4)**

URL user malformed. The user-part of the URL syntax was not correct.

**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\_FTP\_ACCESS\_DENIED (9)**

We were denied access when trying to login to an FTP server or when trying to change working directory to the one given in the URL.

**CURLE\_FTP\_USER\_PASSWORD\_INCORRECT (10)**

The username and/or the password were incorrect when trying to login to an FTP server.

**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\_WEIRD\_USER\_REPLY (12)**

After having sent user name to the FTP server, libcurl expects a proper reply. This error code indicates that an unexpected code was returned.

**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\_FTP\_CANT\_RECONNECT (16)**

A bad return code on either PASV or EPSV was sent by the FTP server, preventing libcurl from being able to continue.

**CURLE\_FTP\_COULDNT\_SET\_BINARY (17)**

Received an error when trying to set the transfer mode to binary.

**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\_FTP\_WRITE\_ERROR (20)**

After a completed file transfer, the FTP server did not respond a proper

**CURLE\_FTP\_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.

**CURLE\_HTTP\_RETURNED\_ERROR (22)**

This is returned if `CURLOPT_FAILONERROR` is set `TRUE` and the HTTP server returns an error code that is  $\geq 400$ .

**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\_MALFORMAT\_USER (24)**

Malformat user. User name badly specified. \*Not currently used\*

**CURLE\_FTP\_COULDNT\_STOR\_FILE (25)**

FTP couldn't STOR file. The server denied the STOR operation. The error buffer usually contains the server's explanation to 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)**

Out of memory. A memory allocation request failed. This is serious badness and things are severely screwed up if this ever occur.

**CURLE\_OPERATION\_TIMEOUTED (28)**

Operation timeout. The specified time-out period was reached according to the conditions.

**CURLE\_FTP\_COULDNT\_SET\_ASCII (29)**

libcurl failed to set ASCII transfer type (TYPE A).

**CURLE\_FTP\_PORT\_FAILED (30)**

The FTP PORT command returned error. This mostly happen when you haven't specified a good enough address for libcurl to use. See `CURLOPT_FTPPORT`.

**CURLE\_FTP\_COULDNT\_USE\_REST (31)**

The FTP REST command returned error. This should never happen if the server is sane.

**CURLE\_FTP\_COULDNT\_GET\_SIZE (32)**

The FTP SIZE command returned error. SIZE is not a kosher FTP command, it is an extension and not all servers support it. This is not a surprising error.

**CURLE\_HTTP\_RANGE\_ERROR (33)**

The HTTP 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\_FTP\_BAD\_DOWNLOAD\_RESUME (36)**

Attempting FTP resume beyond file size.

**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\_LIBRARY\_NOT\_FOUND (40)**

Library not found. The LDAP library was not found.

**CURLE\_FUNCTION\_NOT\_FOUND (41)**

Function not found. A required LDAP 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\_BAD\_CALLING\_ORDER (44)**

Internal error. A function was called in a bad order.

**CURLE\_HTTP\_PORT\_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.

**CURLE\_BAD\_PASSWORD\_ENTERED (46)**

Bad password entered. An error was signaled when the password was entered. This can also be the result of a "bad password" returned from a specified password callback.

**CURLE\_TOO\_MANY\_REDIRECTS (47)**

Too many redirects. When following redirects, libcurl hit the maximum amount. Set your limit with CURLOPT\_MAXREDIRS.

**CURLE\_UNKNOWN\_TELNET\_OPTION (48)**

An option set with CURLOPT\_TELNETOPTIONS was not recognized/known. Refer to the appropriate documentation.

**CURLE\_TELNET\_OPTION\_SYNTAX (49)**

A telnet option string was illegally formatted.

**CURLE\_OBSOLETE (50)**

This is not an error. This used to be another error code in an old libcurl version and is currently unused.

**CURLE\_SSL\_PEER\_CERTIFICATE (51)**

The remote server's SSL certificate 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\_RECV\_ERROR (57)**

Failure with receiving network data.

**CURLE\_SSL\_CERTPROBLEM (58)**

problem with the local certificate

**CURLE\_SSL\_CIPHER (59)**

couldn't use specified cipher

**CURLE\_SSL\_CACERT (60)**

problem with the CA cert (path? access rights?)

**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\_FTP\_SSL\_FAILED (64)**

Requested FTP SSL level failed

**CURLMcode**

This is the generic return code used by functions in the libcurl multi interface.

This is left to be documented.

**CURLSHcode**

The "share" interface will return a CURLSHcode to indicate when an error has occurred.

CURLSHcode is one of the following:

**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.