NAME

CURLOPT_FOLLOWLOCATION − follow HTTP 3xx redirects

SYNOPSIS

#include <curl/curl.h>

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_FOLLOWLOCATION, long mode);

DESCRIPTION

This option tells the library to follow *Location:* header redirects that an HTTP server sends in a 30x response. The *Location:* header can specify a relative or an absolute URL to follow. The long parameter *mode* instructs how libcurl should act on subsequent requests.

mode only had a single value (1L) for a long time that enables redirect following. Since 8.13.0, two additional modes are also supported. See below.

When following redirects, libcurl issues another request for the new URL and follows subsequent new *Location:* redirects all the way until no more such headers are returned or the maximum limit is reached. *CURLOPT_MAXREDIRS(3)* is used to limit the number of redirects libcurl follows.

libcurl restricts what protocols it automatically follow redirects to. The accepted target protocols are set with *CURLOPT_REDIR_PROTOCOLS_STR(3)*. By default libcurl allows HTTP, HTTPS, FTP and FTPS on redirects.

When following a redirect, the specific 30x response code also dictates which request method libcurl uses in the subsequent request: For 301, 302 and 303 responses libcurl switches method from POST to GET unless *CURLOPT_POSTREDIR(3)* instructs libcurl otherwise. All other redirect response codes make libcurl use the same method again.

When libcurl switches method to GET, it then uses that method without sending any request body. If it does not change the method, it sends the subsequent request the same way as the previous one; including the request body if one was provided.

For users who think the existing location following is too naive, too simple or lacking features, it is easy to instead implement your own redirect follow logic with the use of *curl_easy_getinfo(3)*'s *CURLINFO_REDIRECT_URL(3)* option instead of using *CURLOPT_FOLLOWLOCATION(3)*.

By default, libcurl only sends *Authorization:* or explicitly set *Cookie:* headers to the initial host given in the original URL, to avoid leaking username + password to other sites. *CURLOPT_UNRESTRICTED_AUTH(3)* is provided to change that behavior.

Due to the way HTTP works, almost any header can be made to contain data a client may not want to pass on to other servers than the initially intended host and for all other headers than the two mentioned above, there is no protection from this happening when libcurl is told to follow redirects.

Pick one of the following modes:

CURLFOLLOW_ALL (1)

Before 8.13.0 this bit had no name and 1L was the value to enable this option. This makes a set custom method be used in all HTTP requests, even after redirects.

CURLFOLLOW_OBEYCODE (2)

When there is a custom request method set with *CURLOPT_CUSTOMREQUEST(3)*, that set method replaces what libcurl would otherwise use. If a 301/302/303 response code is returned to signal a redirect, the method is changed from POST to *GET*. For 307/308, the custom method remains set and used.

Note that only POST (or a custom post) is changed to GET on 301/302, its not change PUT etc − and therefore also not when libcurl issues a custom PUT. A 303 response makes it switch to GET independently of the original method (except for HEAD).

To control for which of the 301/302/303 status codes libcurl should *not* switch back to GET for when doing a custom POST (a POST transfer using a modified method), and instead keep the custom method, use *CURLOPT_POSTREDIR(3)*.

If you prefer a custom POST method to be reset to exactly the method *POST*, use CURLFOLLOW_FIRSTONLY instead.

CURLFOLLOW_FIRSTONLY (3)

When there is a custom request method set with *CURLOPT_CUSTOMREQUEST(3)*, that set method replaces what libcurl would otherwise use in the first outgoing request only. The second request is then done according to the redirect response code.

If you prefer your custom method to remain in use after a 307/308 redirect, use CURLFOLLOW_OBEYCODE instead.

NOTE

Since libcurl changes method or not based on the specific HTTP response code, setting *CURLOPT_CUSTOMREQUEST(3)* while following redirects may change what libcurl would otherwise do and if not that carefully may even make it misbehave since *CURLOPT_CUSTOMREQUEST(3)* overrides the method libcurl would otherwise select internally.

Setting the CURLFOLLOW_OBEYCODE bit makes libcurl *not* use the custom set method after redirects for 301, 302 and 303 responses. Unless the *CURLOPT_POSTREDIR(3)* bits are set for those status codes.

DEFAULT

0, disabled

PROTOCOLS

This functionality affects http only

EXAMPLE

```
int main(void)
{
 CURL *curl = curl_easy_init();
 if(curl) {
  CURLcode result;
  curl_easy_setopt(curl, CURLOPT_URL, "https://example.com");

  /* example.com is redirected, so we tell libcurl to follow redirection */
  curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);

  result = curl_easy_perform(curl);
  curl_easy_cleanup(curl);
 }
}
```

AVAILABILITY

Added in curl 7.1

RETURN VALUE

curl_easy_setopt(3) returns a CURLcode indicating success or error.

CURLE_OK (0) means everything was OK, non−zero means an error occurred, see *libcurl−errors(3)*.

SEE ALSO

CURLINFO_REDIRECT_COUNT(3), **CURLINFO_REDIRECT_URL**(3), **CURLOPT_POSTREDIR**(3), **CURLOPT_PROTOCOLS_STR**(3), **CURLOPT_REDIR_PROTOCOLS_STR**(3), **CURLOPT_UNRESTRICTED_AUTH**(3)