NAME

CURLOPT_POST − make an HTTP POST

SYNOPSIS

#include <curl/curl.h>

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_POST, long post);

DESCRIPTION

A parameter set to 1 tells libcurl to do a regular HTTP post. This also makes libcurl use a "Content−Type: application/x−www−form−urlencoded" header. This is the most commonly used POST method.

Use one of *CURLOPT_POSTFIELDS(3)* or *CURLOPT_COPYPOSTFIELDS(3)* options to specify what data to post and *CURLOPT_POSTFIELDSIZE(3)* or *CURLOPT_POSTFIELDSIZE_LARGE(3)* to set the data size.

Optionally, you can provide data to POST using the *CURLOPT_READFUNCTION(3)* and *CURLOPT_READDATA(3)* options but then you must make sure to not set *CURLOPT_POSTFIELDS(3)* to anything but NULL. When providing data with a callback, you must transmit it using chunked transfer−encoding or you must set the size of the data with the *CURLOPT_POSTFIELDSIZE(3)* or *CURLOPT_POSTFIELDSIZE_LARGE(3)* options. To enable chunked encoding, pass in the appropriate Transfer−Encoding header, see the post−callback.c example.

You can override the default POST Content−Type: header by setting your own with *CURLOPT_HTTPHEADER(3)*.

Using POST with HTTP 1.1 implies the use of a "Expect: 100−continue" header. You can disable this header with *CURLOPT_HTTPHEADER(3)* as usual.

If you use POST to an HTTP 1.1 server, you can send data without knowing the size before starting the POST if you use chunked encoding. You enable this by adding a header like "Transfer−Encoding: chunked" with *CURLOPT_HTTPHEADER(3)*. With HTTP 1.0 or without chunked transfer, you must specify the size in the request. libcurl automatically uses chunked encoding for POSTs if the size is unknown.

When setting *CURLOPT_POST(3)* to 1, libcurl automatically sets *CURLOPT_NOBODY(3)* and *CURLOPT_HTTPGET(3)* to 0.

If you issue a POST request and then want to make a HEAD or GET using the same reused handle, you must explicitly set the new request type using *CURLOPT_NOBODY(3)* or *CURLOPT_HTTPGET(3)* or similar.

When setting *CURLOPT_POST(3)* to 0, libcurl resets the request type to the default to disable the POST. Typically that means gets reset to GET. Instead you should set a new request type explicitly as described above.

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/foo.bin");
    curl_easy_setopt(curl, CURLOPT_POST, 1L);

    /* set up the read callback with CURLOPT_READFUNCTION */

    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

CURLOPT_HTTPPOST(3), **CURLOPT_POSTFIELDS**(3), **CURLOPT_UPLOAD**(3)