NAME

CURLOPT_POSTFIELDS − data to POST to server

SYNOPSIS

#include <curl/curl.h>

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_POSTFIELDS, char *postdata);

DESCRIPTION

Pass a char pointer as parameter, pointing to the data buffer to use in an HTTP POST operation or an MQTT subscribe. The data must be formatted and encoded the way you want the server to receive it. libcurl does not convert or encode it in any way. For example, a web server may assume that this data is URL encoded.

The data pointed to is NOT copied by the library: as a consequence, it must be preserved by the calling application until the associated transfer finishes. This behavior can be changed (so libcurl does copy the data) by instead using the *CURLOPT_COPYPOSTFIELDS(3)* option.

This POST is a normal **application/x−www−form−urlencoded** kind (and libcurl sets that Content−Type by default when this option is used), which is commonly used by HTML forms. Change Content−Type with *CURLOPT_HTTPHEADER(3)*.

You can use *curl_easy_escape(3)* to URL encode your data, if necessary. It returns a pointer to an encoded string that can be passed as *postdata*.

Using *CURLOPT_POSTFIELDS(3)* implies setting *CURLOPT_POST(3)* to 1.

If *CURLOPT_POSTFIELDS(3)* is explicitly set to NULL then libcurl gets the POST data from the read callback. To send a zero−length (empty) POST, set *CURLOPT_POSTFIELDS(3)* to an empty string, or set *CURLOPT_POST(3)* to 1 and *CURLOPT_POSTFIELDSIZE(3)* to 0.

libcurl assumes this option points to a null−terminated string unless you also set *CURLOPT_POSTFIELDSIZE(3)* to specify the length of the provided data, which then is strictly required if you want to send off null bytes included in the data.

Using POST with HTTP 1.1 implies the use of a "Expect: 100−continue" header, and libcurl adds that header automatically if the POST is either known to be larger than 1MB or if the expected size is unknown. You can disable this header with *CURLOPT_HTTPHEADER(3)* as usual.

To make **multipart/formdata** posts, check out the *CURLOPT_MIMEPOST(3)* option combined with *curl_mime_init(3)*.

Using this option multiple times makes the last set pointer override the previous ones. Set it to NULL to disable its use again.

DEFAULT

NULL

PROTOCOLS

This functionality affects http and mqtt

EXAMPLE

```
/* send an application/x-www-form-urlencoded POST */
int main(void)
{
  CURLcode result = CURLE_OK;
  CURL *curl = curl_easy_init();
```

```
 if(curl) {
  const char *data = "data to send";

  curl_easy_setopt(curl, CURLOPT_URL, "https://example.com");

  /* size of the POST data if strlen() is not good enough */
  curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, 12L);

  /* pass in a pointer to the data - libcurl does not copy */
  curl_easy_setopt(curl, CURLOPT_POSTFIELDS, data);

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

 /* send an application/json POST */
 curl = curl_easy_init();
 if(curl) {
  const char *json = "{\"name\": \"daniel\"}";
  struct curl_slist *slist1 = NULL;
  slist1 = curl_slist_append(slist1, "Content-Type: application/json");
  slist1 = curl_slist_append(slist1, "Accept: application/json");

  /* set custom headers */
  curl_easy_setopt(curl, CURLOPT_HTTPHEADER, slist1);

  curl_easy_setopt(curl, CURLOPT_URL, "https://example.com");

  /* pass in a pointer to the data - libcurl does not copy */
  curl_easy_setopt(curl, CURLOPT_POSTFIELDS, json);

  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_COPYPOSTFIELDS(3), **CURLOPT_MIMEPOST**(3), **CURLOPT_POSTFIELD-SIZE**(3), **CURLOPT_READFUNCTION**(3), **CURLOPT_UPLOAD**(3)