NAME

CURLOPT_TRAILERFUNCTION − callback for sending trailing headers

SYNOPSIS

```
#include <curl.h>

int curl_trailer_callback(struct curl_slist ** list, void *userdata);

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_TRAILERFUNCTION,
                          curl_trailer_callback *func);
```

DESCRIPTION

Pass a pointer to a callback function.

This callback function is called once right before sending the final CR LF in an HTTP chunked transfer to fill a list of trailing headers to be sent before finishing the HTTP transfer.

You can set the userdata argument with the *CURLOPT_TRAILERDATA(3)* option.

The trailing headers included in the linked list must not be CRLF−terminated, because libcurl adds the appropriate line termination characters after each header item.

If you use *curl_slist_append(3)* to add trailing headers to the *curl_slist* then libcurl duplicates the strings, and frees the *curl_slist* once the trailers have been sent.

If one of the trailing header fields is not formatted correctly it is ignored and an info message is emitted.

The return value can either be **CURL_TRAILERFUNC_OK** or **CURL_TRAILERFUNC_ABORT** which would respectively instruct libcurl to either continue with sending the trailers or to abort the request.

If you set this option to NULL, then the transfer proceeds as usual without any interruptions.

DEFAULT

NULL

PROTOCOLS

This functionality affects http only

EXAMPLE

```
extern size_t read_cb(char *ptr, size_t size,
                      size_t nmemb, void *userdata);

static int trailer_cb(struct curl_slist **tr, void *data)
{
 /* libcurl frees the list */
 *tr = curl_slist_append(*tr, "My-super-awesome-trailer: trailer-stuff");
 return CURL_TRAILERFUNC_OK;
}

int main(void)
{
 CURL *curl = curl_easy_init();
 if(curl) {
  CURLcode result;

  /* Set the URL of the request */
  curl_easy_setopt(curl, CURLOPT_URL, "https://example.com/");
```

```
  /* Now set it as a put */
  curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

  /* Assuming we have a function that returns the data to be pushed
     Let that function be read_cb */
  curl_easy_setopt(curl, CURLOPT_READFUNCTION, read_cb);

  struct curl_slist *headers = NULL;
  headers = curl_slist_append(headers, "Trailer: My-super-awesome-trailer");
  result = curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);

  /* Set the trailers filling callback */
  curl_easy_setopt(curl, CURLOPT_TRAILERFUNCTION, trailer_cb);

  /* Perform the transfer */
  result = curl_easy_perform(curl);

  curl_easy_cleanup(curl);

  curl_slist_free_all(headers);
 }
}
```

AVAILABILITY

Added in curl 7.64.0

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_TRAILERDATA(3), **CURLOPT_WRITEFUNCTION**(3)