NAME

CURLOPT_UPLOAD − enable data upload

SYNOPSIS

```
#include <curl/curl.h>

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_UPLOAD, long upload);
```

DESCRIPTION

The long parameter *upload* set to 1 tells the library to prepare for and perform an upload. The *CURLOPT_READDATA(3)* and *CURLOPT_INFILESIZE(3)* or *CURLOPT_INFILESIZE_LARGE(3)* options are also interesting for uploads. If the protocol is HTTP, uploading means using the PUT request unless you tell libcurl otherwise.

Using PUT 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 PUT to a HTTP 1.1 server, you can upload data without knowing the size before starting the transfer 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.

DEFAULT

0, default is download

PROTOCOLS

Most

EXAMPLE

```
CURL *curl = curl_easy_init();
if(curl) {
  /* we want to use our own read function */
  curl_easy_setopt(curl, CURLOPT_READFUNCTION, read_callback);

  /* enable uploading */
  curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

  /* specify target */
  curl_easy_setopt(curl, CURLOPT_URL, "ftp://example.com/dir/to/newfile");

  /* now specify which pointer to pass to our callback */
  curl_easy_setopt(curl, CURLOPT_READDATA, hd_src);

  /* Set the size of the file to upload */
  curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE, (curl_off_t)fsize);

  /* Now run off and do what you’ve been told! */
  curl_easy_perform(curl);
}
```

AVAILABILITY

Always

RETURN VALUE

Returns CURLE_OK

SEE ALSO

CURLOPT_PUT(3), **CURLOPT_READFUNCTION**(3), **CURLOPT_INFILESIZE_LARGE**(3),