NAME

CURLOPT_WRITEFUNCTION − set callback for writing received data

SYNOPSIS

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

size_t write_callback(char *ptr, size_t size, size_t nmemb, void *userdata);

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_WRITEFUNCTION, write_callback);
```

DESCRIPTION

Pass a pointer to your callback function, which should match the prototype shown above.

This callback function gets called by libcurl as soon as there is data received that needs to be saved. *ptr* points to the delivered data, and the size of that data is *size* multiplied with *nmemb*.

The callback function will be passed as much data as possible in all invokes, but you must not make any assumptions. It may be one byte, it may be thousands. The maximum amount of body data that will be passed to the write callback is defined in the curl.h header file: *CURL_MAX_WRITE_SIZE* (the usual default is 16K). If *CURLOPT_HEADER(3)* is enabled, which makes header data get passed to the write callback, you can get up to *CURL_MAX_HTTP_HEADER* bytes of header data passed into it. This usually means 100K.

This function may be called with zero bytes data if the transferred file is empty.

The data passed to this function will not be zero terminated!

Set the *userdata* argument with the *CURLOPT_WRITEDATA(3)* option.

Your callback should return the number of bytes actually taken care of. If that amount differs from the amount passed to your callback function, it'll signal an error condition to the library. This will cause the transfer to get aborted and the libcurl function used will return *CURLE_WRITE_ERROR*.

If your callback function returns CURL_WRITEFUNC_PAUSE it will cause this transfer to become paused. See *curl_easy_pause(3)* for further details.

Set this option to NULL to get the internal default function used instead of your callback. The internal default function will write the data to the FILE * given with *CURLOPT_WRITEDATA(3)*.

DEFAULT

libcurl will use 'fwrite' as a callback by default.

PROTOCOLS

For all protocols

AVAILABILITY

Support for the CURL_WRITEFUNC_PAUSE return code was added in version 7.18.0.

RETURN VALUE

This will return CURLE_OK.

EXAMPLE

A common technique is to use this callback to store the incoming data into a dynamically growing allocated buffer. Like in the getinmemory example: http://curl.haxx.se/libcurl/c/getinmemory.html

SEE ALSO

CURLOPT_WRITEDATA(3), **CURLOPT_READFUNCTION**(3),