NAME

CURLOPT_HEADERFUNCTION − callback that receives header data

SYNOPSIS

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

size_t header_callback(char *buffer,
                       size_t size,
                       size_t nitems,
                       void *userdata);

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_HEADERFUNCTION,
                          header_callback);
```

DESCRIPTION

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

This callback function gets invoked by libcurl as soon as it has received header data. The header callback is called once for each header and only complete header lines are passed on to the callback. Parsing headers is easy to do using this callback. *buffer* points to the delivered data, and the size of that data is *nitems*; *size* is always 1. The provided header line is not null−terminated. Do not modify the passed in buffer.

The pointer named *userdata* is the one you set with the *CURLOPT_HEADERDATA(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 signals an error condition to the library. This causes the transfer to get aborted and the libcurl function used returns *CURLE_WRITE_ERROR*.

You can also abort the transfer by returning CURL_WRITEFUNC_ERROR. (7.87.0)

A complete HTTP header that is passed to this function can be up to *CURL_MAX_HTTP_HEADER* (100K) bytes and includes the final line terminator.

If this option is not set, or if it is set to NULL, but *CURLOPT_HEADERDATA(3)* is set to anything but NULL, the function used to accept response data is used instead. That is the function specified with *CURLOPT_WRITEFUNCTION(3)*, or if it is not specified or NULL − the default, stream−writing function.

It is important to note that the callback is invoked for the headers of all responses received after initiating a request and not the final response. This includes all responses which occur during authentication negotiation. If you need to operate on only the headers from the final response, you need to collect headers in the callback yourself and use HTTP status lines, for example, to delimit response boundaries.

For an HTTP transfer, the status line and the blank line preceding the response body are both included as headers and passed to this function.

When a server sends a chunked encoded transfer, it may contain a trailer. That trailer is identical to an HTTP header and if such a trailer is received it is passed to the application using this callback as well. There are several ways to detect it being a trailer and not an ordinary header: 1) it comes after the response−body. 2) it comes after the final header line (CR LF) 3) a Trailer: header among the regular response−headers mention what header(s) to expect in the trailer.

For non−HTTP protocols like FTP, POP3, IMAP and SMTP this function gets called with the server responses to the commands that libcurl sends.

A more convenient way to get HTTP headers might be to use *curl_easy_header(3)*.

LIMITATIONS

libcurl does not unfold HTTP "folded headers" (deprecated since RFC 7230). A folded header is a header that continues on a subsequent line and starts with a whitespace. Such folds are passed to the header callback as separate ones, although strictly they are continuations of the previous lines.

DEFAULT

Nothing.

PROTOCOLS

This functionality affects ftp, http, imap, pop3 and smtp

EXAMPLE

```
static size_t header_callback(char *buffer, size_t size,
                              size_t nitems, void *userdata)
{
  /* received header is 'nitems' bytes in 'buffer' NOT NULL-TERMINATED */
  /* 'userdata' is set with CURLOPT_HEADERDATA */
  return nitems;
}

int main(void)
{
  CURL *curl = curl_easy_init();
  if(curl) {
    CURLcode result;
    curl_easy_setopt(curl, CURLOPT_URL, "https://example.com");

    curl_easy_setopt(curl, CURLOPT_HEADERFUNCTION, header_callback);

    result = curl_easy_perform(curl);

    curl_easy_cleanup(curl);
  }
}
```

AVAILABILITY

Added in curl 7.7.2

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_HEADERDATA(3), **CURLOPT_WRITEFUNCTION**(3), **curl_easy_header**(3)