## NAME

CURLOPT_HTTPHEADER − set custom HTTP headers

## SYNOPSIS

#include <curl/curl.h>

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_HTTPHEADER, struct curl_slist *headers);

## DESCRIPTION

Pass a pointer to a linked list of HTTP headers to pass to the server and/or proxy in your HTTP request. The same list can be used for both host and proxy requests!

The linked list should be a fully valid list of **struct curl_slist** structs properly filled in. Use *curl_slist_append(3)* to create the list and *curl_slist_free_all(3)* to clean up an entire list. If you add a header that is otherwise generated and used by libcurl internally, your added one will be used instead. If you add a header with no content as in 'Accept:' (no data on the right side of the colon), the internally used header will get disabled. With this option you can add new headers, replace internal headers and remove internal headers. To add a header with no content (nothing to the right side of the colon), use the form 'MyHeader;' (note the ending semicolon).

The headers included in the linked list **must not** be CRLF-terminated, because libcurl adds CRLF after each header item. Failure to comply with this will result in strange bugs because the server will most likely ignore part of the headers you specified.

The first line in a request (containing the method, usually a GET or POST) is not a header and cannot be replaced using this option. Only the lines following the request-line are headers. Adding this method line in this list of headers will only cause your request to send an invalid header. Use *CURLOPT_CUSTOMREQUEST(3)* to change the method.

When this option is passed to *curl_easy_setopt(3)*, libcurl will not copy the entire list so you **must** keep it around until you no longer use this *handle* for a transfer before you call *curl_slist_free_all(3)* on the list.

Pass a NULL to this option to reset back to no custom headers.

The most commonly replaced headers have "shortcuts" in the options *CURLOPT_COOKIE(3)*, *CURLOPT_USERAGENT(3)* and *CURLOPT_REFERER(3)*. We recommend using those.

There's an alternative option that sets or replaces headers only for requests that are sent with CONNECT to a proxy: *CURLOPT_PROXYHEADER(3)*. Use *CURLOPT_HEADEROPT(3)* to control the behavior.

## SECURITY CONCERNS

By default, this option makes libcurl send the given headers in all HTTP requests done by this handle. You should therefore use this option with caution if you for example connect to the remote site using a proxy and a CONNECT request, you should to consider if that proxy is supposed to also get the headers. They may be private or otherwise sensitive to leak.

Use *CURLOPT_HEADEROPT(3)* to make the headers only get sent to where you intend them to get sent.

## DEFAULT

NULL

## PROTOCOLS

HTTP

## EXAMPLE

```
CURL *curl = curl_easy_init();

struct curl_slist *list = NULL;
```

```
if(curl) {
  curl_easy_setopt(curl, CURLOPT_URL, "http://example.com");

  list = curl_slist_append(list, "Shoesize: 10");
  list = curl_slist_append(list, "Accept:");

  curl_easy_setopt(curl, CURLOPT_HTTPHEADER, list);

  curl_easy_perform(curl);

  curl_slist_free_all(list); /* free the list again */
}
```

## AVAILABILITY

As long as HTTP is enabled

## RETURN VALUE

Returns CURLE_OK if HTTP is supported, and CURLE_UNKNOWN_OPTION if not.

## SEE ALSO

**CURLOPT_CUSTOMREQUEST**(3), **CURLOPT_HEADEROPT**(3), **CURLOPT_PROXYHEADER**(3), **CURLOPT_HEADER**(3)