NAME

curl_easy_header - get an HTTP header

SYNOPSIS

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

CURLHcode curl_easy_header(CURL *easy,
                           const char *name,
                           size_t index,
                           unsigned int origin,
                           int request,
                           struct curl_header **hout);
```

DESCRIPTION

curl_easy_header(3) returns a pointer to a "curl_header" struct in **hout** with data for the HTTP response header *name*. The case insensitive null-terminated header name should be specified without colon.

index 0 means asking for the first instance of the header. If the returned header struct has **amount** set larger than 1, it means there are more instances of the same header name available to get. Asking for a too big index makes **CURLHE_BADINDEX** get returned.

The *origin* argument is for specifying which headers to receive, as a single HTTP transfer might provide headers from several different places and they may then have different importance to the user and headers using the same name might be used. The *origin* is a bitmask for what header sources you want. See the descriptions below.

The *request* argument tells libcurl from which request you want headers from. A single transfer might consist of a series of HTTP requests and this argument lets you specify which particular individual request you want the headers from. 0 being the first request and then the number increases for further redirects or when multi-state authentication is used. Passing in -1 is a shortcut to "the last" request in the series, independently of the actual amount of requests used.

libcurl stores and provides the actually used "correct" headers. If for example two headers with the same name arrive and the latter overrides the former, then only the latter is provided. If the first header survives the second, then only the first one is provided. An application using this API does not have to bother about multiple headers used wrongly.

The memory for the returned struct is associated with the easy handle and subsequent calls to *curl_easy_header(3)* clobbers the struct used in the previous calls for the same easy handle. Applications need to copy the data if it wants to keep it around. The memory used for the struct gets

freed with calling *curl_easy_cleanup(3)* of the easy handle.

The first line in an HTTP response is called the status line. It is not considered a header by this function. Headers are the "name: value" lines following the status.

This function can be used before (all) headers have been received and is fine to call from within libcurl callbacks. It returns the state of the headers at the time it is called.

The header struct

```
struct curl_header {
  char *name;
  char *value;
  size_t amount;
  size_t index;
  unsigned int origin;
  void *anchor;
};
```

The data **name** field points to, is the same as the requested name, but might have a different case.

The data **value** field points to, comes exactly as delivered over the network but with leading and trailing whitespace and newlines stripped off. The ‘value‘ data is null-terminated. For legacy HTTP/1 "folded headers", this API provides the full single value in an unfolded manner with a single whitespace between the lines.

amount is how many headers using this name that exist, within the origin and request scope asked for.

index is the zero based entry number of this particular header, which in case this header was used more than once in the requested scope can be larger than 0 but is always less than **amount**.

The **origin** field in the "curl_header" struct has one of the origin bits set, indicating where from the header originates. At the time of this writing, there are 5 bits with defined use. The undocumented 27 remaining bits are reserved for future use and must not be assumed to have any particular value.

anchor is a private handle used by libcurl internals. Do not modify.

ORIGINS

CURLH_HEADER

The header arrived as a header from the server.

CURLH_TRAILER
: The header arrived as a trailer. A header that arrives after the body.

CURLH_CONNECT
: The header arrived in a CONNECT response. A CONNECT request is being done to setup a transfer "through" an HTTP(S) proxy.

CURLH_1XX
: The header arrived in an HTTP 1xx response. A 1xx response is an "intermediate" response that might happen before the "real" response.

CURLH_PSEUDO
: The header is an HTTP/2 or HTTP/3 pseudo header

EXAMPLE

```
int main(void)
{
 struct curl_header *type;
 CURL *curl = curl_easy_init();
 if(curl) {
  CURLHcode h;
  curl_easy_setopt(curl, CURLOPT_URL, "https://example.com");
  curl_easy_perform(curl);
  h = curl_easy_header(curl, "Content-Type", 0, CURLH_HEADER, -1, &type);
  curl_easy_cleanup(curl);
 }
}
```

AVAILABILITY

Added in 7.83.0. Officially supported since 7.84.0.

RETURN VALUE

This function returns a CURLHcode indicating success or error.

SEE ALSO

curl_easy_nextheader(3), **curl_easy_perform**(3), **CURLINFO_CONTENT_TYPE**(3), **CURLOPT_HEADERFUNCTION**(3), **libcurl-errors**(3)