NAME

CURLOPT_ACCEPT_ENCODING − automatic decompression of HTTP downloads

SYNOPSIS

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

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_ACCEPT_ENCODING, char *enc);
```

DESCRIPTION

Pass a char pointer argument specifying what encoding you would like.

Sets the contents of the *Accept−Encoding:* header sent in an HTTP request, and enables decoding of a response when a *Content−Encoding:* header is received.

libcurl potentially supports several different compressed encodings depending on what support that has been built−in.

To aid applications not having to bother about what specific algorithms this particular libcurl build supports, libcurl allows a zero−length string to be set ("") to ask for an *Accept−Encoding:* header to be used that contains all built−in supported encodings.

Alternatively, you can specify exactly the encoding or list of encodings you want in the response. The following encodings are supported: *identity*, meaning non−compressed, *deflate* which requests the server to compress its response using the zlib algorithm, *gzip* which requests the gzip algorithm, (since curl 7.57.0) *br* which is brotli and (since curl 7.72.0) *zstd* which is zstd. Provide them in the string as a comma−separated list of accepted encodings, like: **"br, gzip, deflate"**.

Set *CURLOPT_ACCEPT_ENCODING(3)* to NULL to explicitly disable it, which makes libcurl not send an *Accept−Encoding:* header and not decompress received contents automatically.

You can also opt to include the *Accept−Encoding:* header in your request with *CURLOPT_HTTPHEADER(3)* but then there is no automatic decompressing when receiving data.

Setting this option is a request, not an order; the server may or may not do it. It must be set (to any non−NULL value) or else any encoding done by the server is ignored.

Servers might respond with *Content−Encoding:* even without getting a *Accept−Encoding:* in the request. Servers might respond with a different content encoding than what was asked for in the request.

The *Content−Length:* header field servers send for a compressed response is supposed to indicate the length of the compressed content so when auto decoding is enabled it may not match the sum of bytes reported by the write callbacks (although, sending the length of the non−compressed content is a common server mistake).

The application does not have to keep the string around after setting this option.

Using this option multiple times makes the last set string override the previous ones.

WARNING: when decompressing data, even tiny transfers might be expanded and generate a huge amount of bytes. You might want to limit using this option to only known and trusted sites using secure protocols, perhaps in combination with *CURLOPT_MAXFILESIZE_LARGE(3)*.

HISTORY

This option was called CURLOPT_ENCODING before 7.21.6

NOTES

The specific libcurl you are using must have been built with zlib to be able to decompress gzip and deflate responses, with the brotli library to decompress brotli responses and with the zstd library to decompress zstd responses.

DEFAULT

NULL

PROTOCOLS

This functionality affects http only

EXAMPLE

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

    /* enable all supported built-in compressions */
    curl_easy_setopt(curl, CURLOPT_ACCEPT_ENCODING, "");

    /* Perform the request */
    result = curl_easy_perform(curl);
    curl_easy_cleanup(curl);
  }
}
```

AVAILABILITY

Added in curl 7.21.6

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_HTTPHEADER(3), **CURLOPT_HTTP_CONTENT_DECODING**(3), **CURLOPT_TRANSFER_ENCODING**(3)