NAME

curl_easy_getinfo - extract information from a curl handle

SYNOPSIS

#include <curl/curl.h>

CURLcode curl_easy_getinfo(CURL *curl, CURLINFO info, ...);

DESCRIPTION

Request internal information from the curl session with this function. The third argument **MUST** be a pointer to a long, a pointer to a char *, a pointer to a struct curl_slist * or a pointer to a double (as this documentation describes further down). The data pointed-to will be filled in accordingly and can be relied upon only if the function returns CURLE_OK. Use this function AFTER a performed transfer if you want to get transfer- oriented data.

You should not free the memory returned by this function unless it is explicitly mentioned below.

AVAILABLE INFORMATION

The following information can be extracted:

CURLINFO_EFFECTIVE_URL

Pass a pointer to a char pointer to receive the last used effective URL.

CURLINFO_RESPONSE_CODE

Pass a pointer to a long to receive the last received HTTP, FTP or SMTP response code. This option was previously known as CURLINFO_HTTP_CODE in libcurl 7.10.7 and earlier. The value will be zero if no server response code has been received. Note that a proxy's CONNECT response should be read with *CURLINFO_HTTP_CONNECTCODE* and not this.

Support for SMTP responses added in 7.25.0.

CURLINFO_HTTP_CONNECTCODE

Pass a pointer to a long to receive the last received proxy response code to a CONNECT request.

CURLINFO_FILETIME

Pass a pointer to a long to receive the remote time of the retrieved document (in number of seconds since 1 jan 1970 in the GMT/UTC time zone). If you get -1, it can be because of many reasons (unknown, the server hides it or the server doesn't support the command that tells document time etc) and the time of the document is unknown. Note that you must tell the server to collect this information before the transfer is made, by using the CURLOPT_FILETIME option to *curl_easy_setopt(3)* or you will unconditionally get a -1 back. (Added in 7.5)

CURLINFO_TOTAL_TIME

Pass a pointer to a double to receive the total time in seconds for the previous transfer, including name resolving, TCP connect etc.

CURLINFO_NAMELOOKUP_TIME

Pass a pointer to a double to receive the time, in seconds, it took from the start until the name resolving was completed.

CURLINFO_CONNECT_TIME

Pass a pointer to a double to receive the time, in seconds, it took from the start until the connect to the remote host (or proxy) was completed.

CURLINFO_APPCONNECT_TIME

Pass a pointer to a double to receive the time, in seconds, it took from the start until the SSL/SSH connect/handshake to the remote host was completed. This time is most often very near to the PRETRANSFER time, except for cases such as HTTP pippelining where the pretransfer time can be delayed due to waits in line for the pipeline and more. (Added in 7.19.0)

CURLINFO_PRETRANSFER_TIME

Pass a pointer to a double to receive the time, in seconds, it took from the start until the file transfer is just about to begin. This includes all pre-transfer commands and negotiations that are specific to the particular protocol(s) involved. It does *not* involve the sending of the protocol- specific request that triggers a transfer.

CURLINFO_STARTTRANSFER_TIME

Pass a pointer to a double to receive the time, in seconds, it took from the start until the first byte is received by libcurl. This includes CURLINFO_PRETRANSFER_TIME and also the time the server needs to calculate the result.

CURLINFO_REDIRECT_TIME

Pass a pointer to a double to receive the total time, in seconds, it took for all redirection steps include name lookup, connect, pretransfer and transfer before final transaction was started. CURLINFO_REDIRECT_TIME contains the complete execution time for multiple redirections. (Added in 7.9.7)

CURLINFO_REDIRECT_COUNT

Pass a pointer to a long to receive the total number of redirections that were actually followed. (Added in 7.9.7)

CURLINFO_REDIRECT_URL

Pass a pointer to a char pointer to receive the URL a redirect *would* take you to if you would enable CURLOPT_FOLLOWLOCATION. This can come very handy if you think using the built-in libcurl redirect logic isn't good enough for you but you would still prefer to avoid implementing all the magic of figuring out the new URL. (Added in 7.18.2)

CURLINFO_SIZE_UPLOAD

Pass a pointer to a double to receive the total amount of bytes that were uploaded.

CURLINFO_SIZE_DOWNLOAD

Pass a pointer to a double to receive the total amount of bytes that were downloaded. The amount is only for the latest transfer and will be reset again for each new transfer.

CURLINFO_SPEED_DOWNLOAD

Pass a pointer to a double to receive the average download speed that curl measured for the complete download. Measured in bytes/second.

CURLINFO_SPEED_UPLOAD

Pass a pointer to a double to receive the average upload speed that curl measured for the complete upload. Measured in bytes/second.

CURLINFO_HEADER_SIZE

Pass a pointer to a long to receive the total size of all the headers received. Measured in number of bytes.

CURLINFO_REQUEST_SIZE

Pass a pointer to a long to receive the total size of the issued requests. This is so far only for HTTP requests. Note that this may be more than one request if FOLLOWLOCATION is true.

CURLINFO_SSL_VERIFYRESULT

Pass a pointer to a long to receive the result of the certification verification that was requested (using the CURLOPT_SSL_VERIFYPEER option to *curl_easy_setopt(3)*).

CURLINFO_SSL_ENGINES

Pass the address of a 'struct curl_slist *' to receive a linked-list of OpenSSL crypto-engines supported. Note that engines are normally implemented in separate dynamic libraries. Hence not all the returned engines may be available at run-time. **NOTE:** you must call *curl_slist_free_all(3)* on the list pointer once you're done with it, as libcurl will not free the data for you. (Added in 7.12.3)

CURLINFO_CONTENT_LENGTH_DOWNLOAD
: Pass a pointer to a double to receive the content-length of the download. This is the value read from the Content-Length: field. Since 7.19.4, this returns -1 if the size isn't known.

CURLINFO_CONTENT_LENGTH_UPLOAD
: Pass a pointer to a double to receive the specified size of the upload. Since 7.19.4, this returns -1 if the size isn't known.

CURLINFO_CONTENT_TYPE
: Pass a pointer to a char pointer to receive the content-type of the downloaded object. This is the value read from the Content-Type: field. If you get NULL, it means that the server didn't send a valid Content-Type header or that the protocol used doesn't support this.

CURLINFO_PRIVATE
: Pass a pointer to a char pointer to receive the pointer to the private data associated with the curl handle (set with the CURLOPT_PRIVATE option to *curl_easy_setopt(3)*). Please note that for internal reasons, the value is returned as a char pointer, although effectively being a 'void *'. (Added in 7.10.3)

CURLINFO_HTTPAUTH_AVAIL
: Pass a pointer to a long to receive a bitmask indicating the authentication method(s) available. The meaning of the bits is explained in the CURLOPT_HTTPAUTH option for *curl_easy_setopt(3)*. (Added in 7.10.8)

CURLINFO_PROXYAUTH_AVAIL
: Pass a pointer to a long to receive a bitmask indicating the authentication method(s) available for your proxy authentication. (Added in 7.10.8)

CURLINFO_OS_ERRNO
: Pass a pointer to a long to receive the errno variable from a connect failure. Note that the value is only set on failure, it is not reset upon a successful operation. (Added in 7.12.2)

CURLINFO_NUM_CONNECTS
: Pass a pointer to a long to receive how many new connections libcurl had to create to achieve the previous transfer (only the successful connects are counted). Combined with *CURLINFO_REDIRECT_COUNT* you are able to know how many times libcurl successfully reused existing connection(s) or not. See the Connection Options of *curl_easy_setopt(3)* to see how libcurl tries to make persistent connections to save time. (Added in 7.12.3)

CURLINFO_PRIMARY_IP
: Pass a pointer to a char pointer to receive the pointer to a zero-terminated string holding the IP address of the most recent connection done with this **curl** handle. This string may be IPv6 if that's enabled. Note that you get a pointer to a memory area that will be re-used at next request so you need to copy the string if you want to keep the information. (Added in 7.19.0)

CURLINFO_PRIMARY_PORT
: Pass a pointer to a long to receive the destination port of the most recent connection done with this **curl** handle. (Added in 7.21.0)

CURLINFO_LOCAL_IP
: Pass a pointer to a char pointer to receive the pointer to a zero-terminated string holding the local (source) IP address of the most recent connection done with this **curl** handle. This string may be IPv6 if that's enabled. The same restrictions apply as to *CURLINFO_PRIMARY_IP*. (Added in 7.21.0)

CURLINFO_LOCAL_PORT
: Pass a pointer to a long to receive the local (source) port of the most recent connection done with this **curl** handle. (Added in 7.21.0)

CURLINFO_COOKIELIST
: Pass a pointer to a 'struct curl_slist *' to receive a linked-list of all cookies cURL knows (expired ones, too). Don't forget to *curl_slist_free_all(3)* the list after it has been used. If there are no

cookies (cookies for the handle have not been enabled or simply none have been received) ’struct curl_slist *’ will be set to point to NULL. (Added in 7.14.1)

CURLINFO_LASTSOCKET

Pass a pointer to a long to receive the last socket used by this curl session. If the socket is no longer valid, -1 is returned. When you finish working with the socket, you must call curl_easy_cleanup() as usual and let libcurl close the socket and cleanup other resources associated with the handle. This is typically used in combination with *CURLOPT_CONNECT_ONLY*. (Added in 7.15.2)

NOTE: this API is not really working on win64, since the SOCKET type on win64 is 64 bit large while its ’long’ is only 32 bits.

CURLINFO_FTP_ENTRY_PATH

Pass a pointer to a char pointer to receive a pointer to a string holding the path of the entry path. That is the initial path libcurl ended up in when logging on to the remote FTP server. This stores a NULL as pointer if something is wrong. (Added in 7.15.4)

Also works for SFTP since 7.21.4

CURLINFO_CERTINFO

Pass a pointer to a ’struct curl_certinfo *’ and you’ll get it set to point to struct that holds a number of linked lists with info about the certificate chain, assuming you had CURLOPT_CERTINFO enabled when the previous request was done. The struct reports how many certs it found and then you can extract info for each of those certs by following the linked lists. The info chain is provided in a series of data in the format "name:content" where the content is for the specific named data. See also the certinfo.c example. NOTE: this option is only available in libcurl built with OpenSSL, NSS, GSKit or QsoSSL support. (Added in 7.19.1)

CURLINFO_TLS_SESSION

Pass a pointer to a ’struct curl_tlsinfo *’. The pointer will be initialized to refer to a ’struct curl_tlsinfo *’ that will contain an enum indicating the SSL library used for the handshake and the respective internal TLS session structure of this underlying SSL library.

This may then be used to extract certificate information in a format convenient for further processing, such as manual validation. NOTE: this option may not be available for all SSL backends; unsupported SSL backends will return ’CURLSSLBACKEND_NONE’ to indicate that they are not supported; this does not mean that no SSL backend was used. (Added in 7.34.0)

CURLINFO_CONDITION_UNMET

Pass a pointer to a long to receive the number 1 if the condition provided in the previous request didn’t match (see *CURLOPT_TIMECONDITION*). Alas, if this returns a 1 you know that the reason you didn’t get data in return is because it didn’t fulfill the condition. The long ths argument points to will get a zero stored if the condition instead was met. (Added in 7.19.4)

CURLINFO_RTSP_SESSION_ID

Pass a pointer to a char pointer to receive a pointer to a string holding the most recent RTSP Session ID.

Applications wishing to resume an RTSP session on another connection should retrieve this info before closing the active connection.

CURLINFO_RTSP_CLIENT_CSEQ

Pass a pointer to a long to receive the next CSeq that will be used by the application.

CURLINFO_RTSP_SERVER_CSEQ

Pass a pointer to a long to receive the next server CSeq that will be expected by the application.

(NOTE: listening for server initiated requests is currently unimplemented).

Applications wishing to resume an RTSP session on another connection should retrieve this info before closing the active connection.

CURLINFO_RTSP_CSEQ_RECV

Pass a pointer to a long to receive the most recently received CSeq from the server. If your application encounters a *CURLE_RTSP_CSEQ_ERROR* then you may wish to troubleshoot and/or fix the CSeq mismatch by peeking at this value.

TIMES

An overview of the six time values available from curl_easy_getinfo()

```
curl_easy_perform()
    |
    |--NAMELOOKUP
    |--|--CONNECT
    |--|--|--APPCONNECT
    |--|--|--|--PRETRANSFER
    |--|--|--|--|--STARTTRANSFER
    |--|--|--|--|--|--TOTAL
    |--|--|--|--|--|--REDIRECT
```

NAMELOOKUP

CURLINFO_NAMELOOKUP_TIME. The time it took from the start until the name resolving was completed.

CONNECT

CURLINFO_CONNECT_TIME. The time it took from the start until the connect to the remote host (or proxy) was completed.

APPCONNECT

CURLINFO_APPCONNECT_TIME. The time it took from the start until the SSL connect/handshake with the remote host was completed. (Added in in 7.19.0)

PRETRANSFER

CURLINFO_PRETRANSFER_TIME. The time it took from the start until the file transfer is just about to begin. This includes all pre-transfer commands and negotiations that are specific to the particular protocol(s) involved.

STARTTRANSFER

CURLINFO_STARTTRANSFER_TIME. The time it took from the start until the first byte is received by libcurl.

TOTAL

CURLINFO_TOTAL_TIME. Total time of the previous request.

REDIRECT

CURLINFO_REDIRECT_TIME. The time it took for all redirection steps include name lookup, connect, pretransfer and transfer before final transaction was started. So, this is zero if no redirection took place.

RETURN VALUE

If the operation was successful, CURLE_OK is returned. Otherwise an appropriate error code will be returned.

SEE ALSO

curl_easy_setopt(3)