NAME

CURLOPT_COOKIELIST − add to or manipulate cookies held in memory

SYNOPSIS

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

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_COOKIELIST,
                          char *cookie);
```

DESCRIPTION

Pass a char pointer to a *cookie* string.

Such a cookie can be either a single line in Netscape / Mozilla format or regular HTTP−style header (*Set−Cookie:*) format. This option also enables the cookie engine. This adds that single cookie to the internal cookie store.

We strongly advice against loading cookies from an HTTP header file, as that is an inferior data exchange format.

Exercise caution if you are using this option and multiple transfers may occur. If you use the *Set−Cookie* format and the string does not specify a domain, then the cookie is sent for any domain (even after redirects are followed) and cannot be modified by a server−set cookie. If a server sets a cookie of the same name (or maybe you have imported one) then both are sent on future transfers to that server, likely not what you intended. To address these issues set a domain in *Set−Cookie* (doing that includes subdomains) or much better: use the Netscape file format.

Additionally, there are commands available that perform actions if you pass in these exact strings:

ALL erases all cookies held in memory

SESS erases all session cookies held in memory

FLUSH
: writes all known cookies to the file specified by *CURLOPT_COOKIEJAR(3)*

RELOAD
: loads all cookies from the files specified by *CURLOPT_COOKIEFILE(3)*

DEFAULT

NULL

PROTOCOLS

This functionality affects http only

EXAMPLE

```
/* an inline import of a cookie in Netscape format. */

#define SEP "\t"  /* Tab separates the fields */

int main(void)
{
  const char *my_cookie =
    "example.com"    /* Hostname */
    SEP "FALSE"      /* Include subdomains */
    SEP "/"          /* Path */
    SEP "FALSE"      /* Secure */
    SEP "0"          /* Expiry in epoch time format. 0 == Session */
    SEP "foo"        /* Name */
    SEP "bar";       /* Value */
```

```
 CURL *curl = curl_easy_init();
 if(curl) {
  CURLcode result;
  /* my_cookie is imported immediately via CURLOPT_COOKIELIST. */
  curl_easy_setopt(curl, CURLOPT_COOKIELIST, my_cookie);

  /* The list of cookies in cookies.txt are not be imported until right
     before a transfer is performed. Cookies in the list that have the same
     hostname, path and name as in my_cookie are skipped. That is because
     libcurl has already imported my_cookie and it is considered a "live"
     cookie. A live cookie is not replaced by one read from a file.
  */
  curl_easy_setopt(curl, CURLOPT_COOKIEFILE, "cookies.txt");  /* import */

  /* Cookies are exported after curl_easy_cleanup is called. The server
     may have added, deleted or modified cookies by then. The cookies that
     were skipped on import are not exported.
  */
  curl_easy_setopt(curl, CURLOPT_COOKIEJAR, "cookies.txt");  /* export */

  result = curl_easy_perform(curl); /* cookies imported from cookies.txt */

  curl_easy_cleanup(curl);  /* cookies exported to cookies.txt */
 }
}
```

Cookie file format

The cookie file format and general cookie concepts in curl are described online here: https://curl.se/docs/http−cookies.html

HISTORY

ALL was added in 7.14.1

SESS was added in 7.15.4

FLUSH was added in 7.17.1

RELOAD was added in 7.39.0

AVAILABILITY

Added in curl 7.14.1

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

CURLINFO_COOKIELIST(3), **CURLOPT_COOKIE**(3), **CURLOPT_COOKIEFILE**(3), **CURLOPT_COOKIEJAR**(3)