

**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 \* to a *cookie* string.

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

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)*

**DEFAULT**

NULL

**PROTOCOLS**

HTTP

**EXAMPLE**

TODO

**AVAILABILITY**

ALL was added in 7.14.1

SESS was added in 7.15.4

FLUSH was added in 7.17.1

**RETURN VALUE**

Returns CURLE\_OK if the option is supported, CURLE\_UNKNOWN\_OPTION if not, or CURLE\_OUT\_OF\_MEMORY if there was insufficient heap space.

**SEE ALSO**

CURLOPT\_COOKIEFILE(3), CURLOPT\_COOKIEJAR(3),

**NAME**

curl\_getdate - Convert a date string to number of seconds

**SYNOPSIS**

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

```
time_t curl_getdate(char *datestring, time_t *now );
```

**DESCRIPTION**

*curl\_getdate(3)* returns the number of seconds since the Epoch, January 1st 1970 00:00:00 in the UTC time zone, for the date and time that the *datestring* parameter specifies. The *now* parameter is not used, pass a NULL there.

**PARSING DATES AND TIMES**

A "date" is a string containing several items separated by whitespace. The order of the items is immaterial. A date string may contain many flavors of items:

**calendar date items**

Can be specified several ways. Month names can only be three-letter english abbreviations, numbers can be zero-prefixed and the year may use 2 or 4 digits. Examples: 06 Nov 1994, 06-Nov-94 and Nov-94 6.

**time of the day items**

This string specifies the time on a given day. You must specify it with 6 digits with two colons: HH:MM:SS. To not include the time in a date string, will make the function assume 00:00:00. Example: 18:19:21.

**time zone items**

Specifies international time zone. There are a few acronyms supported, but in general you should instead use the specific relative time compared to UTC. Supported formats include: -1200, MST, +0100.

**day of the week items**

Specifies a day of the week. Days of the week may be spelled out in full (using english): 'Sunday', 'Monday', etc or they may be abbreviated to their first three letters. This is usually not info that adds anything.

**pure numbers**

If a decimal number of the form YYYYMMDD appears, then YYYY is read as the year, MM as the month number and DD as the day of the month, for the specified calendar date.

**EXAMPLES**

```
Sun, 06 Nov 1994 08:49:37 GMT
Sunday, 06-Nov-94 08:49:37 GMT
Sun Nov 6 08:49:37 1994
06 Nov 1994 08:49:37 GMT
06-Nov-94 08:49:37 GMT
Nov 6 08:49:37 1994
06 Nov 1994 08:49:37
06-Nov-94 08:49:37
1994 Nov 6 08:49:37
GMT 08:49:37 06-Nov-94 Sunday
94 6 Nov 08:49:37
1994 Nov 6
06-Nov-94
Sun Nov 6 94
1994.Nov.6
Sun/Nov/6/94/GMT
Sun, 06 Nov 1994 08:49:37 CET
06 Nov 1994 08:49:37 EST
```

Sun, 12 Sep 2004 15:05:58 -0700  
Sat, 11 Sep 2004 21:32:11 +0200  
20040912 15:05:58 -0700  
20040911 +0200

## STANDARDS

This parser was written to handle date formats specified in RFC 822 (including the update in RFC 1123) using time zone name or time zone delta and RFC 850 (obsoleted by RFC 1036) and ANSI C's `asctime()` format. These formats are the only ones RFC 7231 says HTTP applications may use.

## RETURN VALUE

This function returns -1 when it fails to parse the date string. Otherwise it returns the number of seconds as described.

If the year is larger than 2037 on systems with 32 bit `time_t`, this function will return 0x7fffffff (since that is the largest possible signed 32 bit number).

Having a 64 bit `time_t` is not a guarantee that dates beyond 03:14:07 UTC, January 19, 2038 will work fine. On systems with a 64 bit `time_t` but with a crippled `mktime()`, *curl\_getdate(3)* will return -1 in this case.

## SEE ALSO

`curl_easy_escape(3)`, `curl_easy_unescape(3)`, `CURLOPT_TIMECONDITION(3)`, `CURLOPT_TIMEVALUE(3)`