NAME

CURLMOPT_TIMERFUNCTION − callback to receive timeout values

SYNOPSIS

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

int timer_callback(CURLM *multi,    /* multi handle */
                   long timeout_ms, /* timeout in number of ms */
                   void *clientp);  /* private callback pointer */

CURLMcode curl_multi_setopt(CURLM *handle, CURLMOPT_TIMERFUNCTION, timer_callback);
```

DESCRIPTION

Pass a pointer to your callback function, which should match the prototype shown above.

Certain features, such as timeouts and retries, require you to call libcurl even when there is no activity on the file descriptors.

Your callback function **timer_callback** should install a single non−repeating timer with an expire time of **timeout_ms** milliseconds. When that timer fires, call either *curl_multi_socket_action(3)* or *curl_multi_perform(3)*, depending on which interface you use.

If this callback is called when a timer is already running, this new expire time *replaces* the former timeout. The application should then effectively cancel the old timeout and set a new timeout using this new expire time.

A **timeout_ms** value of −1 passed to this callback means you should delete the timer. All other values are valid expire times in number of milliseconds − including zero milliseconds.

The **timer_callback** is called when the timeout expire time is changed.

The **clientp** pointer is set with *CURLMOPT_TIMERDATA(3)*.

The timer callback should return 0 on success, and −1 on error. If this callback returns error, **all** transfers currently in progress in this multi handle are aborted and made to fail.

This callback can be used instead of, or in addition to, *curl_multi_timeout(3)*.

WARNING: do not call libcurl directly from within the callback itself when the **timeout_ms** value is zero, since it risks triggering an unpleasant recursive behavior that immediately calls another call to the callback with a zero timeout...

DEFAULT

NULL

PROTOCOLS

This functionality affects all supported protocols

EXAMPLE

```
struct priv {
  void *custom;
};

static int timerfunc(CURLM *multi, long timeout_ms, void *clientp)
{
  struct priv *mydata = clientp;
  printf("our ptr: %p\n", mydata->custom);
```

```
 if(timeout_ms >= 0) {
  /* this is the new single timeout to wait for, including zero */
 }
 else {
  /* delete the timeout, nothing to wait for now */
 }
 return 0;
}

int main(void)
{
 struct priv mydata;
 CURLM *multi = curl_multi_init();
 curl_multi_setopt(multi, CURLMOPT_TIMERFUNCTION, timerfunc);
 curl_multi_setopt(multi, CURLMOPT_TIMERDATA, &mydata);
}
```

AVAILABILITY

Added in curl 7.16.0

RETURN VALUE

curl_multi_setopt(3) returns a CURLMcode indicating success or error.

CURLM_OK (0) means everything was OK, non−zero means an error occurred, see *libcurl−errors(3)*.

SEE ALSO

CURLMOPT_SOCKETFUNCTION(3), **CURLMOPT_TIMERDATA**(3)