

## NAME

`curl_multi_setopt` – set options for a curl multi handle

## SYNOPSIS

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

```
CURLMcode curl_multi_setopt(CURLM * multi_handle, CURLMoption option, param);
```

## DESCRIPTION

`curl_multi_setopt()` is used to tell a libcurl multi handle how to behave. By using the appropriate options to *curl\_multi\_setopt(3)*, you can change libcurl's behaviour when using that multi handle. All options are set with the *option* followed by the parameter *param*. That parameter can be a **long**, a **function pointer**, an **object pointer** or a **curl\_off\_t** type, depending on what the specific option expects. Read this manual carefully as bad input values may cause libcurl to behave badly! You can only set one option in each function call.

## OPTIONS

### CURLMOPT\_SOCKETFUNCTION

Pass a pointer to a function matching the **curl\_socket\_callback** prototype. The *curl\_multi\_socket(3)* functions inform the application about updates in the socket (file descriptor) status by doing none, one or multiple calls to the `curl_socket_callback` given in the **param** argument. They update the status with changes since the previous time a *curl\_multi\_socket(3)* function was called. If the given callback pointer is NULL, no callback will be called. Set the callback's **userp** argument with *CURLMOPT\_SOCKETDATA*. See *curl\_multi\_socket(3)* for more callback details.

### CURLMOPT\_SOCKETDATA

Pass a pointer to whatever you want passed to the **curl\_socket\_callback**'s forth argument, the **userp** pointer. This is not used by libcurl but only passed-thru as-is. Set the callback pointer with *CURLMOPT\_SOCKETFUNCTION*.

### CURLMOPT\_PIPELINING

Pass a long set to 1 to enable or 0 to disable. Enabling pipelining on a multi handle will make it attempt to perform HTTP Pipelining as far as possible for transfers using this handle. This means that if you add a second request that can use an already existing connection, the second request will be "piped" on the same connection rather than being executed in parallel. (Added in 7.16.0)

### CURLMOPT\_TIMERFUNCTION

Pass a pointer to a function matching the **curl\_multi\_timer\_callback** prototype. This function will then be called when the timeout value changes. The timeout value is at what latest time the application should call one of the "performing" functions of the multi interface (*curl\_multi\_socket(3)*, *curl\_multi\_socket\_all(3)* and *curl\_multi\_perform(3)*) - to allow libcurl to keep timeouts and retries etc to work. Libcurl attempts to limit calling this only when the fixed future timeout time actually change. See also *CURLMOPT\_TIMERDATA*. This callback can be used instead of, or in addition to, *curl\_multi\_timeout(3)*. (Added in 7.16.0)

### CURLMOPT\_TIMERDATA

Pass a pointer to whatever you want passed to the **curl\_multi\_timer\_callback**'s third argument, the **userp** pointer. This is not used by libcurl but only passed-thru as-is. Set the callback pointer with *CURLMOPT\_TIMERFUNCTION*. (Added in 7.16.0)

## RETURNS

The standard CURLMcode for multi interface error codes. Note that it returns a *CURLM\_UNKNOWN\_OPTION* if you try setting an option that this version of libcurl doesn't know of.

## AVAILABILITY

This function was added in libcurl 7.15.4.

**SEE ALSO**

**`curl_multi_cleanup(3)`, `curl_multi_init(3)`, `curl_multi_socket(3)`, `curl_multi_info_read(3)`**