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_action(3)* function informs 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_action(3)* and *curl_multi_perform(3)*) - to allow libcurl to keep timeouts and retries etc to work. A timeout value of -1 means that there is no timeout at all, and 0 means that the timeout is already reached. Libcurl attempts to limit calling this only when the fixed future timeout time actually changes. 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)

CURLMOPT_MAXCONNECTS

Pass a long. The set number will be used as the maximum amount of simultaneously open connections that libcurl may cache. Default is 10, and libcurl will enlarge the size for each added easy handle to make it fit 4 times the number of added easy handles.

By setting this option, you can prevent the cache size from growing beyond the limit set by you.

When the cache is full, curl closes the oldest one in the cache to prevent the number of open connections from increasing.

This option is for the multi handle's use only, when using the easy interface you should instead use the *CURLOPT_MAXCONNECTS* option.

(Added in 7.16.3)

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)