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 fourth 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: int curl_multi_timer_callback(CURLM *multi /* multi handle */, long timeout_ms /* timeout in milliseconds */, void *userp /* TIMERDATA */). 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*. The callback should return 0 on success, and -1 on error. 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)

CURLMOPT_MAX_HOST_CONNECTIONS

Pass a long. The set number will be used as the maximum amount of simultaneously open connections to a single host. For each new session to a host, libcurl will open a new connection up to the limit set by CURLMOPT_MAX_HOST_CONNECTIONS. When the limit is reached, the sessions will be pending until there are available connections. If CURLMOPT_PIPELINING is 1, libcurl will try to pipeline if the host is capable of it.

The default value is 0, which means that there is no limit. However, for backwards compatibility, setting it to 0 when CURLMOPT_PIPELINING is 1 will not be treated as unlimited. Instead it will open only 1 connection and try to pipeline on it.

(Added in 7.30.0)

CURLMOPT_MAX_PIPELINE_LENGTH

Pass a long. The set number will be used as the maximum amount of requests in a pipelined connection. When this limit is reached, libcurl will use another connection to the same host (see CURLMOPT_MAX_HOST_CONNECTIONS), or queue the requests until one of the pipelines to the host is ready to accept a request. Thus, the total number of requests in-flight is CURLMOPT_MAX_HOST_CONNECTIONS * CURLMOPT_MAX_PIPELINE_LENGTH. The default value is 5.

(Added in 7.30.0)

CURLMOPT_CONTENT_LENGTH_PENALTY_SIZE

Pass a long. If a pipelined connection is currently processing a request with a Content-Length larger than CURLMOPT_CONTENT_LENGTH_PENALTY_SIZE, that connection will not be considered for additional requests, even if it is shorter than CURLMOPT_MAX_PIPELINE_LENGTH. The default value is 0, which means that the penalization is inactive.

(Added in 7.30.0)

CURLMOPT_CHUNK_LENGTH_PENALTY_SIZE

Pass a long. If a pipelined connection is currently processing a chunked (Transfer-encoding: chunked) request with a current chunk length larger than CURLMOPT_CHUNK_LENGTH_PENALTY_SIZE, that connection will not be considered for additional requests, even if it is shorter than CURLMOPT_MAX_PIPELINE_LENGTH. The default value is 0, which means that the penalization is inactive.

(Added in 7.30.0)

CURLMOPT_PIPELINING_SITE_BL

Pass an array of char *, ending with NULL. This is a list of sites that are blacklisted from pipelining, i.e sites that are known to not support HTTP pipelining. The array is copied by libcurl.

The default value is NULL, which means that there is no blacklist.

Pass a NULL pointer to clear the blacklist.

Example:

```
site_blacklist[] =
{
 "www.haxx.se",
 "www.example.com:1234",
 NULL
};

curl_multi_setopt(m, CURLMOPT_PIPELINE_SITE_BL, site_blacklist);
```

(Added in 7.30.0)

CURLMOPT_PIPELINING_SERVER_BL

Pass an array of char *, ending with NULL. This is a list of server types prefixes (in the Server: HTTP header) that are blacklisted from pipelining, i.e server types that are known to not support HTTP pipelining. The array is copied by libcurl.

Note that the comparison matches if the Server: header begins with the string in the blacklist, i.e "Server: Ninja 1.2.3" and "Server: Ninja 1.4.0" can both be blacklisted by having "Ninja" in the backlist.

The default value is NULL, which means that there is no blacklist.

Pass a NULL pointer to clear the blacklist.

Example:

```
server_blacklist[] =
{
 "Microsoft-IIS/6.0",
 "nginx/0.8.54",
 NULL
};

curl_multi_setopt(m, CURLMOPT_PIPELINE_SERVER_BL, server_blacklist);
```

(Added in 7.30.0)

CURLMOPT_MAX_TOTAL_CONNECTIONS

Pass a long. The set number will be used as the maximum amount of simultaneously open connections in total. For each new session, libcurl will open a new connection up to the limit set by CURLMOPT_MAX_TOTAL_CONNECTIONS. When the limit is reached, the sessions will be pending until there are available connections. If CURLMOPT_PIPELINING is 1, libcurl will try to pipeline if the host is capable of it.

The default value is 0, which means that there is no limit. However, for backwards compatibility, setting it to 0 when CURLMOPT_PIPELINING is 1 will not be treated as unlimited. Instead it will open only 1 connection and try to pipeline on it.

(Added in 7.30.0)

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)