NAME

curl_multi_add_handle - add an easy handle to a multi session

SYNOPSIS

#include <curl/curl.h>

CURLMcode curl_multi_add_handle(CURLM *multi_handle, CURL *easy_handle);

DESCRIPTION

Adds a standard easy handle to the multi stack. This function call will make this *multi_handle* control the specified *easy_handle*.

While an easy handle is added to a multi stack, you cannot and you must not use *curl_easy_perform(3)* on that handle. After having removed the easy handle from the multi stack again, it is perfectly fine to use it with the easy interface again.

If the easy handle is not set to use a shared (*CURLOPT_SHARE(3)*) or global DNS cache (*CURLOPT_DNS_USE_GLOBAL_CACHE(3)*), it will be made to use the DNS cache that is shared between all easy handles within the multi handle when *curl_multi_add_handle(3)* is called.

When an easy interface is added to a multi handle, it will use a shared connection cache owned by the multi handle. Removing and adding new easy handles will not affect the pool of connections or the ability to do connection re-use.

If you have CURLMOPT_TIMERFUNCTION set in the multi handle (and you really should if you're working event-based with *curl_multi_socket_action(3)* and friends), that callback will be called from within this function to ask for an updated timer so that your main event loop will get the activity on this handle to get started.

The easy handle will remain added to the multi handle until you remove it again with *curl_multi_remove_handle(3)* - even when a transfer with that specific easy handle is completed.

You should remove the easy handle from the multi stack before you terminate first the easy handle and then the multi handle:

1 - *curl_multi_remove_handle(3)*

2 - *curl_easy_cleanup(3)*

3 - *curl_multi_cleanup(3)*

RETURN VALUE

CURLMcode type, general libcurl multi interface error code.

SEE ALSO

curl_multi_cleanup(3),**curl_multi_init**(3), **curl_multi_setopt**(3), **curl_multi_socket_action**(3)