NAME

curl_multi_socket - reads/writes available data

SYNOPSIS

```
#include <curl/curl.h>
CURLMcode curl_multi_socket(CURLM *multi_handle, curl_socket_t sockfd,
                            int *running_handles);

CURLMcode curl_multi_socket_all(CURLM *multi_handle,
                                int *running_handles);
```

DESCRIPTION

These functions are deprecated. Do not use. See *curl_multi_socket_action(3)* instead.

At return, the integer **running_handles** points to contains the number of still running easy handles within the multi handle. When this number reaches zero, all transfers are complete/done. Note that when you call *curl_multi_socket_action(3)* on a specific socket and the counter decreases by one, it DOES NOT necessarily mean that this exact socket/transfer is the one that completed. Use *curl_multi_info_read(3)* to figure out which easy handle that completed.

The *curl_multi_socket_action(3)* functions inform the application about updates in the socket (file descriptor) status by doing none, one, or multiple calls to the socket callback function set with the *CURLMOPT_SOCKETFUNCTION(3)* option to *curl_multi_setopt(3)*. They update the status with changes since the previous time the callback was called.

Get the timeout time by setting the *CURLMOPT_TIMERFUNCTION(3)* option with *curl_multi_setopt(3)*. Your application then gets called with information on how long to wait for socket actions at most before doing the timeout action: call the *curl_multi_socket_action(3)* function with the **sockfd** argument set to CURL_SOCKET_TIMEOUT. You can also use the *curl_multi_timeout(3)* function to poll the value at any given time, but for an event-based system using the callback is far better than relying on polling the timeout value.

Usage of *curl_multi_socket(3)* is deprecated, whereas the function is equivalent to *curl_multi_socket_action(3)* with **ev_bitmask** set to 0.

Force libcurl to (re-)check all its internal sockets and transfers instead of just a single one by calling *curl_multi_socket_all(3)*. Note that there should not be any reason to use this function.

EXAMPLE

```
int main(void)
```

```
{
 /* the event-library gets told when there activity on the socket 'fd',
    which we translate to a call to curl_multi_socket_action() */
 int running;
 int rc;
 int fd;
 CURLM *multi;
 rc = curl_multi_socket(multi, fd, &running);
}
```

AVAILABILITY

This function was added in libcurl 7.15.4, and is deemed stable since 7.16.0.

curl_multi_socket(3) is deprecated, use *curl_multi_socket_action(3)* instead!

RETURN VALUE

CURLMcode type, general libcurl multi interface error code.

The return code is for the whole multi stack. Problems still might have occurred on individual transfers even when one of these functions return OK.

SEE ALSO

curl_multi_cleanup(3), **curl_multi_init**(3), **curl_multi_fdset**(3), **curl_multi_info_read**(3), **the**hiperfifo.c**example**