NAME

CURLOPT_SOCKOPTFUNCTION − callback for setting socket options

SYNOPSIS

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

typedef enum {
  CURLSOCKTYPE_IPCXN,  /* socket created for a specific IP connection */
  CURLSOCKTYPE_ACCEPT, /* socket created by accept() call */
  CURLSOCKTYPE_LAST    /* never use */
} curlsocktype;

#define CURL_SOCKOPT_OK 0
#define CURL_SOCKOPT_ERROR 1 /* causes libcurl to abort and return
                        CURLE_ABORTED_BY_CALLBACK */
#define CURL_SOCKOPT_ALREADY_CONNECTED 2

int sockopt_callback(void *clientp,
                     curl_socket_t curlfd,
                     curlsocktype purpose);

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_SOCKOPTFUNCTION, sockopt_callback);
```

DESCRIPTION

Pass a pointer to your callback function, which should match the prototype shown above.

When set, this callback function gets called by libcurl when the socket has been created, but before the connect call to allow applications to change specific socket options. The callback's *purpose* argument identifies the exact purpose for this particular socket:

CURLSOCKTYPE_IPCXN for actively created connections or *CURLSOCKTYPE_ACCEPT* for FTP when the connection was setup with PORT/EPSV (in earlier versions these sockets were not passed to this callback).

Future versions of libcurl may support more purposes. libcurl passes the newly created socket descriptor to the callback in the *curlfd* parameter so additional setsockopt() calls can be done at the user's discretion.

The *clientp* pointer contains whatever user−defined value set using the *CURLOPT_SOCKOPTDATA(3)* function.

Return *CURL_SOCKOPT_OK* from the callback on success. Return *CURL_SOCKOPT_ERROR* from the callback function to signal an unrecoverable error to the library and it closes the socket and returns *CURLE_COULDNT_CONNECT* for *CURLSOCKTYPE_IPCXN* and *CURLE_ABORTED_BY_CALLBACK* for *CURLSOCKTYPE_ACCEPT*.

The callback function may return *CURL_SOCKOPT_ALREADY_CONNECTED* for *CURLSOCKTYPE_IPCXN*, to tell libcurl that the socket is already connected and then libcurl does no attempt to connect. This allows an application to pass in an already connected socket with *CURLOPT_OPENSOCKETFUNCTION(3)* and then have this function make libcurl not attempt to connect (again).

DEFAULT

NULL

PROTOCOLS

This functionality affects all supported protocols

EXAMPLE

```
/* make libcurl use the already established socket 'sockfd' */

static curl_socket_t opensocket(void *clientp,
                                curlsocktype purpose,
                                struct curl_sockaddr *address)
{
  curl_socket_t sockfd;
  sockfd = *(curl_socket_t *)clientp;
  /* the actual externally set socket is passed in via the OPENSOCKETDATA
     option */
  return sockfd;
}

static int sockopt_callback(void *clientp, curl_socket_t curlfd,
                            curlsocktype purpose)
{
  return CURL_SOCKOPT_ALREADY_CONNECTED;
}

int main(void)
{
  CURL *curl = curl_easy_init();
  if(curl) {
    CURLcode result;
    int sockfd; /* our custom file descriptor */
    /* libcurl thinks that you connect to the host
     * and port that you specify in the URL option. */
    curl_easy_setopt(curl, CURLOPT_URL, "http://99.99.99.99:9999");
    /* call this function to get a socket */
    curl_easy_setopt(curl, CURLOPT_OPENSOCKETFUNCTION, opensocket);
    curl_easy_setopt(curl, CURLOPT_OPENSOCKETDATA, &sockfd);

    /* call this function to set options for the socket */
    curl_easy_setopt(curl, CURLOPT_SOCKOPTFUNCTION, sockopt_callback);

    result = curl_easy_perform(curl);

    curl_easy_cleanup(curl);
  }
}
```

AVAILABILITY

Added in curl 7.16.0

RETURN VALUE

curl_easy_setopt(3) returns a CURLcode indicating success or error.

CURLE_OK (0) means everything was OK, non−zero means an error occurred, see *libcurl−errors(3)*.

SEE ALSO

CURLOPT_OPENSOCKETFUNCTION(3), **CURLOPT_SEEKFUNCTION**(3), **CURLOPT_SOCKOPTDATA**(3)