NAME

CURLOPT_OPENSOCKETDATA − pointer passed to open socket callback

SYNOPSIS

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

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_OPENSOCKETDATA, void *pointer);
```

DESCRIPTION

Pass a *pointer* that is untouched by libcurl and passed as the first argument in the open socket callback set with *CURLOPT_OPENSOCKETFUNCTION(3)*.

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;
    extern int sockfd; /* the already connected one */

    /* 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 7.17.1

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_CLOSESOCKETFUNCTION(3), **CURLOPT_OPENSOCKETFUNCTION**(3), **CURLOPT_SOCKOPTFUNCTION**(3)