NAME

CURLOPT_CLOSESOCKETDATA − pointer passed to the socket close callback

SYNOPSIS

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

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

DESCRIPTION

Pass a *pointer* that remains untouched by libcurl and passed as the first argument in the closesocket callback set with *CURLOPT_CLOSESOCKETFUNCTION(3)*.

Note that when using multi/share handles, your callback may get invoked even after the easy handle has been cleaned up. The callback and data is inherited by a new connection and that connection may live longer than the transfer itself in the multi/share handle's connection cache.

DEFAULT

NULL

PROTOCOLS

This functionality affects all supported protocols

EXAMPLE

```
struct priv {
  void *custom;
};

static int closesocket(void *clientp, curl_socket_t item)
{
  struct priv *my = clientp;
  printf("our ptr: %p\n", my->custom);

  printf("libcurl wants to close %d now\n", (int)item);
  return 0;
}

int main(void)
{
  struct priv myown;
  CURL *curl = curl_easy_init();
  if(curl) {
    CURLcode result;
    /* call this function to close sockets */
    curl_easy_setopt(curl, CURLOPT_CLOSESOCKETFUNCTION, closesocket);
    curl_easy_setopt(curl, CURLOPT_CLOSESOCKETDATA, &myown);

    result = curl_easy_perform(curl);
    curl_easy_cleanup(curl);
  }
}
```

AVAILABILITY

Added in curl 7.21.7

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)