NAME

CURLOPT_DEBUGDATA − pointer passed to the debug callback

SYNOPSIS

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

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

DESCRIPTION

Pass a *pointer* to whatever you want passed in to your *CURLOPT_DEBUGFUNCTION(3)* in the last void * argument. This pointer is not used by libcurl, it is only passed to the callback.

DEFAULT

NULL

PROTOCOLS

This functionality affects all supported protocols

EXAMPLE

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

static int my_trace(CURL *handle, curl_infotype type,
                    char *data, size_t size,
                    void *clientp)
{
  struct data *mine = clientp;
  printf("our ptr: %p\n", mine->custom);

  /* output debug info */
  return 0;
}

int main(void)
{
  CURL *curl;
  CURLcode result;
  struct data my_tracedata;

  curl = curl_easy_init();
  if(curl) {
    curl_easy_setopt(curl, CURLOPT_DEBUGFUNCTION, my_trace);

    curl_easy_setopt(curl, CURLOPT_DEBUGDATA, &my_tracedata);

    /* the DEBUGFUNCTION has no effect until we enable VERBOSE */
    curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L);

    curl_easy_setopt(curl, CURLOPT_URL, "https://example.com/");
    result = curl_easy_perform(curl);

    /* always cleanup */
    curl_easy_cleanup(curl);
  }
  return 0;
}
```

AVAILABILITY

Added in curl 7.9.6

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_DEBUGFUNCTION(3), **CURLOPT_STDERR**(3)