NAME

　　　　CURLOPT_DEBUGFUNCTION – debug callback

SYNOPSIS

　　　　#include <curl/curl.h>

　　　　typedef enum {
　　　　 CURLINFO_TEXT = 0,
　　　　 CURLINFO_HEADER_IN,   /* 1 */
　　　　 CURLINFO_HEADER_OUT, /* 2 */
　　　　 CURLINFO_DATA_IN,   /* 3 */
　　　　 CURLINFO_DATA_OUT,   /* 4 */
　　　　 CURLINFO_SSL_DATA_IN, /* 5 */
　　　　 CURLINFO_SSL_DATA_OUT, /* 6 */
　　　　 CURLINFO_END
　　　　} curl_infotype;

　　　　int debug_callback(CURL *handle,
　　　　　　　　curl_infotype type,
　　　　　　　　char *data,
　　　　　　　　size_t size,
　　　　　　　　void *userptr);

　　　　CURLcode curl_easy_setopt(CURL *handle, CURLOPT_DEBUGFUNCTION,
　　　　　　　　debug_callback);

DESCRIPTION

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

　　　　CURLOPT_DEBUGFUNCTION(3) replaces the standard debug function used when *CURLOPT_VER-BOSE(3)* is in effect. This callback receives debug information, as specified in the *type* argument. This function must return 0. The *data* pointed to by the char * passed to this function WILL NOT be zero terminated, but will be exactly of the *size* as told by the *size* argument.

　　　　The *userptr* argument is the pointer set with *CURLOPT_DEBUGDATA(3)*.

　　　　Available curl_infotype values:

　　　　CURLINFO_TEXT
　　　　　　　　The data is informational text.

　　　　CURLINFO_HEADER_IN
　　　　　　　　The data is header (or header-like) data received from the peer.

　　　　CURLINFO_HEADER_OUT
　　　　　　　　The data is header (or header-like) data sent to the peer.

　　　　CURLINFO_DATA_IN
　　　　　　　　The data is protocol data received from the peer.

　　　　CURLINFO_DATA_OUT
　　　　　　　　The data is protocol data sent to the peer.

　　　　CURLINFO_SSL_DATA_OUT
　　　　　　　　The data is SSL/TLS (binary) data sent to the peer.

　　　　CURLINFO_SSL_DATA_IN
　　　　　　　　The data is SSL/TLS (binary) data received from the peer.

DEFAULT

   NULL

PROTOCOLS

   All

EXAMPLE

```
static
void dump(const char *text,
      FILE *stream, unsigned char *ptr, size_t size)
{
 size_t i;
 size_t c;
 unsigned int width=0x10;

 fprintf(stream, "%s, %10.10ld bytes (0x%8.8lx)0,
      text, (long)size, (long)size);

 for(i=0; i<size; i+= width) {
  fprintf(stream, "%4.4lx: ", (long)i);

  /* show hex to the left */
  for(c = 0; c < width; c++) {
   if(i+c < size)
     fprintf(stream, "%02x ", ptr[i+c]);
   else
     fputs("   ", stream);
  }

  /* show data on the right */
  for(c = 0; (c < width) && (i+c < size); c++)
   fputc(ptr[i+c]>=0x20) && (ptr[i+c]<0x80)?ptr[i+c]:'.', stream);

  fputc('0, stream); /* newline */
 }
}

static
int my_trace(CURL *handle, curl_infotype type,
      char *data, size_t size,
      void *userp)
{
 const char *text;
 (void)handle; /* prevent compiler warning */

 switch (type) {
 case CURLINFO_TEXT:
  fprintf(stderr, "== Info: %s", data);
 default: /* in case a new one is introduced to shock us */
  return 0;

 case CURLINFO_HEADER_OUT:
  text = "=> Send header";
  break;
 case CURLINFO_DATA_OUT:
  text = "=> Send data";
```

```
      break;
    case CURLINFO_SSL_DATA_OUT:
      text = "=> Send SSL data";
      break;
    case CURLINFO_HEADER_IN:
      text = "<= Recv header";
      break;
    case CURLINFO_DATA_IN:
      text = "<= Recv data";
      break;
    case CURLINFO_SSL_DATA_IN:
      text = "<= Recv SSL data";
      break;
    }

    dump(text, stderr, (unsigned char *)data, size);
    return 0;
  }

  int main(void)
  {
    CURL *curl;
    CURLcode res;

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

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

      /* example.com is redirected, so we tell libcurl to follow redirection */
      curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);

      curl_easy_setopt(curl, CURLOPT_URL, "http://example.com/");
      res = curl_easy_perform(curl);
      /* Check for errors */
      if(res != CURLE_OK)
        fprintf(stderr, "curl_easy_perform() failed: %s0,
            curl_easy_strerror(res));

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

AVAILABILITY

Always

RETURN VALUE

Returns CURLE_OK

SEE ALSO

CURLOPT_VERBOSE(3), **CURLOPT_DEBUGDATA**(3),