NAME

CURLOPT_URL − URL for this transfer

SYNOPSIS

#include <curl/curl.h>

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_URL, char *URL);

DESCRIPTION

Pass in a pointer to the *URL* to work with. The parameter should be a char * to a null−terminated string which must be URL−encoded in the following format:

scheme://host:port/path

For a greater explanation of the format please see RFC 3986.

libcurl does not validate the syntax or use the URL until the transfer is started. Even if you set a crazy value here, *curl_easy_setopt(3)* might still return *CURLE_OK*.

If the given URL is missing a scheme name (such as "http://" or "ftp://" etc) then libcurl guesses based on the host. If the outermost subdomain name matches DICT, FTP, IMAP, LDAP, POP3 or SMTP then that protocol gets used, otherwise HTTP is used. Scheme guessing can be disabled by setting a default protocol, see *CURLOPT_DEFAULT_PROTOCOL(3)* for details.

Should the protocol, either as specified by the URL scheme or deduced by libcurl from the hostname, not be supported by libcurl then *CURLE_UNSUPPORTED_PROTOCOL* is returned from either the *curl_easy_perform(3)* or *curl_multi_perform(3)* functions when you call them. Use *curl_version_info(3)* for detailed information of which protocols are supported by the build of libcurl you are using.

CURLOPT_PROTOCOLS_STR(3) can be used to limit what protocols libcurl may use for this transfer, independent of what libcurl has been compiled to support. That may be useful if you accept the URL from an external source and want to limit the accessibility.

The *CURLOPT_URL(3)* string is ignored if *CURLOPT_CURLU(3)* is set.

Either *CURLOPT_URL(3)* or *CURLOPT_CURLU(3)* must be set before a transfer is started.

The application does not have to keep the string around after setting this option.

Using this option multiple times makes the last set string override the previous ones. Set it to NULL to disable its use again. Note however that libcurl needs a URL set to be able to performed a transfer.

The parser used for handling the URL set with *CURLOPT_URL(3)* is the same that *curl_url_set(3)* uses.

ENCODING

The string pointed to in the *CURLOPT_URL(3)* argument is generally expected to be a sequence of characters using an ASCII compatible encoding.

If libcurl is built with IDN support, the server name part of the URL can use an "international name" by using the current encoding (according to locale) or UTF−8 (when WinIDN is used; or a Windows Unicode build using libidn2).

If libcurl is built without IDN support, the server name is used exactly as specified when passed to the name resolver functions.

DEFAULT

NULL. If this option is not set, no transfer can be performed.

SECURITY CONCERNS

Applications may at times find it convenient to allow users to specify URLs for various purposes and that string would then end up fed to this option.

Getting a URL from an external untrusted party brings several security concerns:

If you have an application that runs as or in a server application, getting an unfiltered URL can easily trick your application to access a local resource instead of a remote. Protecting yourself against localhost accesses is hard when accepting user provided URLs.

Such custom URLs can also access other ports than you planned as port numbers are part of the regular URL format. The combination of a local host and a custom port number can allow external users to play tricks with your local services.

Accepting external URLs may also use other protocols than http:// or other common ones. Restrict what accept with *CURLOPT_PROTOCOLS_STR(3)*.

User provided URLs can also be made to point to sites that redirect further on (possibly to other protocols too). Consider your *CURLOPT_FOLLOWLOCATION(3)* and *CURLOPT_REDIR_PROTOCOLS_STR(3)* settings.

PROTOCOLS

This functionality affects all supported protocols

EXAMPLE

```
int main(void)
{
  CURL *curl = curl_easy_init();
  if(curl) {
    CURLcode result;
    curl_easy_setopt(curl, CURLOPT_URL, "https://example.com");

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

AVAILABILITY

Added in curl 7.1

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)*.

Note that *curl_easy_setopt(3)* does not parse the given string so given a bad URL, it is not detected until *curl_easy_perform(3)* or similar is called.

SEE ALSO

CURLINFO_REDIRECT_URL(3), **CURLOPT_CURLU**(3), **CURLOPT_FORBID_REUSE**(3), **CURLOPT_FRESH_CONNECT**(3), **CURLOPT_PATH_AS_IS**(3), **CURLOPT_PROTOCOLS_STR**(3), **curl_easy_perform**(3), **curl_url_get**(3), **curl_url_set**(3)