NAME

CURLOPT_URL − provide the URL to use in the request

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 zero terminated string which must be URL-encoded in the following format:

scheme://host:port/path

For a greater explanation of the format please see RFC3986.

libcurl doesn't validate the syntax or use this variable until the transfer is issued. Even if you set a crazy value here, *curl_easy_setopt(3)* will still return *CURLE_OK*.

If the given URL lacks the scheme (such as "http://" or "ftp://" etc) then libcurl will attempt to resolve the protocol based on one of the following given host names: HTTP, FTP, DICT, LDAP, IMAP, POP3 or SMTP

Should the protocol, either that specified by the scheme or deduced by libcurl from the host name, not be supported by libcurl then *CURLE_UNSUPPORTED_PROTOCOL* will be 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(3) can be used to limit what protocols libcurl will 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.

CURLOPT_URL(3) is the only option that **must** be set before a transfer is started.

The host part of the URL contains the address of the server that you want to connect to. This can be the fully qualified domain name of the server, the local network name of the machine on your network or the IP address of the server or machine represented by either an IPv4 or IPv6 address. For example:

http://www.example.com/

http://hostname/

http://192.168.0.1/

http://[2001:1890:1112:1::20]/

It is also possible to specify the user name, password and any supported login options as part of the host, for the following protocols, when connecting to servers that require authentication:

http://user:password@www.example.com

ftp://user:password@ftp.example.com

smb://domain%2fuser:password@server.example.com

imap://user:password;options@mail.example.com

pop3://user:password;options@mail.example.com

smtp://user:password;options@mail.example.com

At present only IMAP, POP3 and SMTP support login options as part of the host. For more information about the login options in URL syntax please see RFC2384, RFC5092 and IETF draft draft-earhart-url-smtp-00.txt (Added in 7.31.0).

The port is optional and when not specified libcurl will use the default port based on the determined or specified protocol: 80 for HTTP, 21 for FTP and 25 for SMTP, etc. The following examples show how to specify the port:

http://www.example.com:8080/ - This will connect to a web server using port 8080 rather than 80.

smtp://mail.example.com:587/ - This will connect to a SMTP server on the alternative mail port.

The path part of the URL is protocol specific and whilst some examples are given below this list is not conclusive:

HTTP The path part of a HTTP request specifies the file to retrieve and from what directory. If the directory is not specified then the web server's root directory is used. If the file is omitted then the default document will be retrieved for either the directory specified or the root directory. The exact resource returned for each URL is entirely dependent on the server's configuration.

http://www.example.com - This gets the main page from the web server.

http://www.example.com/index.html - This returns the main page by explicitly requesting it.

http://www.example.com/contactus/ - This returns the default document from the contactus directory.

FTP The path part of an FTP request specifies the file to retrieve and from what directory. If the file part is omitted then libcurl downloads the directory listing for the directory specified. If the directory is omitted then the directory listing for the root / home directory will be returned.

ftp://ftp.example.com - This retrieves the directory listing for the root directory.

ftp://ftp.example.com/readme.txt - This downloads the file readme.txt from the root directory.

ftp://ftp.example.com/libcurl/readme.txt - This downloads readme.txt from the libcurl directory.

ftp://user:password@ftp.example.com/readme.txt - This retrieves the readme.txt file from the user's home directory. When a username and password is specified, everything that is specified in the path part is relative to the user's home directory. To retrieve files from the root directory or a directory underneath the root directory then the absolute path must be specified by prepending an additional forward slash to the beginning of the path.

ftp://user:password@ftp.example.com//readme.txt - This retrieves the readme.txt from the root directory when logging in as a specified user.

SMTP The path part of a SMTP request specifies the host name to present during communication with the mail server. If the path is omitted then libcurl will attempt to resolve the local computer's host name. However, this may not return the fully qualified domain name that is required by some mail

servers and specifying this path allows you to set an alternative name, such as your machine's fully qualified domain name, which you might have obtained from an external function such as gethostname or getaddrinfo.

smtp://mail.example.com - This connects to the mail server at example.com and sends your local computer's host name in the HELO / EHLO command.

smtp://mail.example.com/client.example.com - This will send client.example.com in the HELO / EHLO command to the mail server at example.com.

POP3 The path part of a POP3 request specifies the message ID to retrieve. If the ID is not specified then a list of waiting messages is returned instead.

pop3://user:password@mail.example.com - This lists the available messages for the user

pop3://user:password@mail.example.com/1 - This retrieves the first message for the user

IMAP The path part of an IMAP request not only specifies the mailbox to list (Added in 7.30.0) or select, but can also be used to check the UIDVALIDITY of the mailbox, to specify the UID, SECTION (Added in 7.30.0) and PARTIAL octets (Added in 7.37.0) of the message to fetch and to specify what messages to search for (Added in 7.37.0).

imap://user:password@mail.example.com - Performs a top level folder list

imap://user:password@mail.example.com/INBOX - Performs a folder list on the user's inbox

imap://user:password@mail.example.com/INBOX/;UID=1 - Selects the user's inbox and fetches message 1

imap://user:password@mail.example.com/INBOX;UIDVALIDITY=50/;UID=2 - Selects the user's inbox, checks the UIDVALIDITY of the mailbox is 50 and fetches message 2 if it is

imap://user:password@mail.example.com/INBOX/;UID=3/;SECTION=TEXT - Selects the user's inbox and fetches the text portion of message 3

imap://user:password@mail.example.com/INBOX/;UID=4/;PARTIAL=0.1024 - Selects the user's inbox and fetches the first 1024 octets of message 4

imap://user:password@mail.example.com/INBOX?NEW - Selects the user's inbox and checks for NEW messages

imap://user:password@mail.example.com/INBOX?SUBJECT%20shadows - Selects the user's inbox and searches for messages containing "shadows" in the subject line

For more information about the individual components of an IMAP URL please see RFC5092.

SCP The path part of a SCP request specifies the file to retrieve and from what directory. The file part may not be omitted. The file is taken as an absolute path from the root directory on the server. To specify a path relative to the user's home directory on the server, prepend ~/ to the path portion. If the user name is not embedded in the URL, it can be set with the *CURLOPT_USERPWD(3)* or *CURLOPT_USERNAME(3)* option.

scp://user@example.com/etc/issue - This specifies the file /etc/issue

scp://example.com/~/my-file - This specifies the file my-file in the user's home directory on the server

SFTP The path part of a SFTP request specifies the file to retrieve and from what directory. If the file part is omitted then libcurl downloads the directory listing for the directory specified. If the path ends in a / then a directory listing is returned instead of a file. If the path is omitted entirely then the directory listing for the root / home directory will be returned. If the user name is not embedded in the URL, it can be set with the *CURLOPT_USERPWD(3)* or *CURLOPT_USERNAME(3)* option.

sftp://user:password@example.com/etc/issue - This specifies the file /etc/issue

sftp://user@example.com/~/my-file - This specifies the file my-file in the user's home directory

sftp://ssh.example.com/~/Documents/ - This requests a directory listing of the Documents directory under the user's home directory

SMB The path part of a SMB request specifies the file to retrieve and from what share and directory or the share to upload to and as such, may not be omitted. If the user name is not embedded in the URL, it can be set with the *CURLOPT_USERPWD(3)* or *CURLOPT_USERNAME(3)* option. If the user name is embedded in the URL then it must contain the domain name and as such, the backslash must be URL encoded as %2f.

smb://server.example.com/files/issue - This specifies the file "issue" located in the root of the "files" share

smb://server.example.com/files/ -T issue - This specifies the file "issue" will be uploaded to the root of the "files" share.

LDAP The path part of a LDAP request can be used to specify the: Distinguished Name, Attributes, Scope, Filter and Extension for a LDAP search. Each field is separated by a question mark and when that field is not required an empty string with the question mark separator should be included.

ldap://ldap.example.com/o=My%20Organisation - This will perform a LDAP search with the DN as My Organisation.

ldap://ldap.example.com/o=My%20Organisation?postalAddress - This will perform the same search but will only return postalAddress attributes.

ldap://ldap.example.com/?rootDomainNamingContext - This specifies an empty DN and requests information about the rootDomainNamingContext attribute for an Active Directory server.

For more information about the individual components of a LDAP URL please see RFC4516.

RTMP There's no official URL spec for RTMP so libcurl uses the URL syntax supported by the underlying librtmp library. It has a syntax where it wants a traditional URL, followed by a space and a series of space-separated name=value pairs.

While space is not typically a "legal" letter, libcurl accepts them. When a user wants to pass in a '#' (hash) character it will be treated as a fragment and get cut off by libcurl if provided literally. You will instead have to escape it by providing it as backslash and its ASCII value in hexadecimal: "\23".

DEFAULT

There is no default URL. If this option isn't 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 will bring reasons for 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 very 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(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(3)* settings.

PROTOCOLS

All

EXAMPLE

```
CURL *curl = curl_easy_init();
if(curl) {
  curl_easy_setopt(curl, CURLOPT_URL, "http://example.com");

  curl_easy_perform(curl);
}
```

AVAILABILITY

POP3 and SMTP were added in 7.31.0

RETURN VALUE

Returns CURLE_OK on success or CURLE_OUT_OF_MEMORY if there was insufficient heap space.

Note that *curl_easy_setopt(3)* won't actually parse the given string so given a bad URL, it will not be detected until *curl_easy_perform(3)* or similar is called.

SEE ALSO

CURLOPT_VERBOSE(3), **CURLOPT_PROTOCOLS**(3), **CURLOPT_FORBID_REUSE**(3), **CURLOPT_FRESH_CONNECT**(3), **curl_easy_perform**(3)