# FAQ -- Frequently Asked Questions

## Philosophy



## Install Related Problems



## Usage Problems

## Running Problems



## libcurl Issues

## License Issues



## PHP/CURL Issues



---

## 1. Philosophy

#### 1.1 What is cURL?

cURL is the name of the project. The name is a play on 'Client for URLs', originally with URL spelled in uppercase to make it obvious it deals with URLs. The fact it can also be pronounced 'see URL' also helped, it works as an abbreviation for "Client URL Request Library" or why not the recursive version: "curl URL Request Library".

The cURL project produces two products:

libcurl

A free and easy-to-use client-side URL transfer library, supporting DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP.

libcurl supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading, Kerberos, SPNEGO, HTTP form based upload, proxies, cookies, user+password authentication, file transfer resume, http proxy tunneling and more!

libcurl is highly portable, it builds and works identically on numerous platforms, including Solaris, NetBSD, FreeBSD, OpenBSD, Darwin, HP-UX, IRIX, AIX, Tru64, Linux, UnixWare, HURD, Windows, Amiga, OS/2, BeOS, Mac OS X, Ultrix, QNX, OpenVMS, RISC OS, Novell NetWare, DOS, Symbian, OSF, Android, Minix, IBM TPF and more...

libcurl is free, thread-safe, IPv6 compatible, feature rich, well supported and fast.

curl

A command line tool for getting or sending files using URL syntax.

Since curl uses libcurl, curl supports the same wide range of common Internet protocols that libcurl does.

We pronounce curl with an initial k sound. It rhymes with words like girl and earl. This is a short WAV file to help you:

```
https://media.merriam-webster.com/soundc11/c/curl0001.wav
```

There are numerous sub-projects and related projects that also use the word curl in the project names in various combinations, but you should take notice that this FAQ is directed at the command-line tool named curl (and libcurl the library), and may therefore not be valid for other curl-related projects. (There is however a small section for the PHP/CURL in this FAQ.)

**1.2 What is libcurl?**

libcurl is a reliable and portable library which provides you with an easy interface to a range of common Internet protocols.

You can use libcurl for free in your application, be it open source, commercial or closed-source.

libcurl is most probably the most portable, most powerful and most often used C-based multi-platform file transfer library on this planet - be it open source or commercial.

**1.3 What is curl not?**

curl is not a wget clone. That is a common misconception. Never, during curl's development, have we intended curl to replace wget or compete on its market.

curl is targeted at single-shot file transfers.

curl is not a website mirroring program. If you want to use curl to mirror something: fine, go ahead and write a script that wraps around curl or use libcurl to make it reality.

curl is not an FTP site mirroring program. Sure, get and send FTP with curl but if you want systematic and sequential behavior you should write a script (or write a new program that interfaces libcurl) and do it.

curl is not a PHP tool, even though it works perfectly well when used from or with PHP (when using the PHP/CURL module).

curl is not a program for a single operating system. curl exists, compiles, builds and runs under a wide range of operating systems, including all modern Unixes (and a bunch of older ones too), Windows, Amiga, BeOS, OS/2, OS X, QNX etc.

**1.4 When will you make curl do XXXX ?**

We love suggestions of what to change in order to make curl and libcurl better. We do however believe in a few rules when it comes to the future of curl:

curl -- the command line tool -- is to remain a non-graphical command line tool. If you want GUIs or fancy scripting capabilities, you should look for another tool that uses libcurl.

We do not add things to curl that other small and available tools already do very well at the side. curl's output can be piped into another program or redirected to another file for the next program to interpret.

We focus on protocol related issues and improvements. If you want to do more magic with the supported protocols than curl currently does, chances are good we will agree. If you want to add more protocols, we may very well agree.

If you want someone else to do all the work while you wait for us to implement it for you, that is not a very friendly attitude. We spend a considerable time already on maintaining and developing curl. In order to get more out of us, you should consider trading in some of your time and effort in return. Simply go to the GitHub repo which resides at https://github.com/curl/curl, fork the project, and create pull requests with your proposed changes.

If you write the code, chances are better that it will get into curl faster.

**1.5 Who makes curl?**

curl and libcurl are not made by any single individual. Daniel Stenberg is project leader and main developer, but other persons' submissions are important and

crucial. Anyone can contribute and post their changes and improvements and have them inserted in the main sources (of course on the condition that developers agree that the fixes are good).

The full list of all contributors is found in the docs/THANKS file.

curl is developed by a community, with Daniel at the wheel.

**1.6 What do you get for making curl?**

Project cURL is entirely free and open. We do this voluntarily, mostly in our spare time. Companies may pay individual developers to work on curl, but that's up to each company and developer. This is not controlled by nor supervised in any way by the curl project.

We get help from companies. Haxx provides website, bandwidth, mailing lists etc, GitHub hosts the primary git repository and other services like the bug tracker at https://github.com/curl/curl. Also again, some companies have sponsored certain parts of the development in the past and I hope some will continue to do so in the future.

If you want to support our project, consider a donation or a banner-program or even better: by helping us with coding, documenting or testing etc.

See also: https://curl.se/sponsors.html

**1.7 What about CURL from curl.com?**

During the summer of 2001, curl.com was busy advertising their client-side programming language for the web, named CURL.

We are in no way associated with curl.com or their CURL programming language.

Our project name curl has been in effective use since 1998. We were not the first computer related project to use the name "curl" and do not claim any rights to the name.

We recognize that we will be living in parallel with curl.com and wish them every success.

**1.8 I have a problem whom do I mail?**

Please do not mail any single individual unless you really need to. Keep curl-related questions on a suitable mailing list. All available mailing lists are listed in the MANUAL document and online at https://curl.se/mail/

Keeping curl-related questions and discussions on mailing lists allows others to join in and help, to share their ideas, to contribute their suggestions and to spread their wisdom. Keeping discussions on public mailing lists also allows for others to learn from this (both current and future users thanks to the web based archives of the mailing lists), thus saving us from having to repeat ourselves even more. Thanks for respecting this.

If you have found or simply suspect a security problem in curl or libcurl, mail curl-security at haxx.se (closed list of receivers, mails are not disclosed) and tell. Then we can produce a fix in a timely manner before the flaw is announced to the world, thus lessen the impact the problem will have on existing users.

**1.9 Where do I buy commercial support for curl?**

curl is fully open source. It means you can hire any skilled engineer to fix your curl-related problems.

We list available alternatives on the curl website: https://curl.se/support.html

**1.10 How many are using curl?**

It is impossible to tell.

We don't know how many users that knowingly have installed and use curl.

We don't know how many users that use curl without knowing that they are in fact using it.

We don't know how many users that downloaded or installed curl and then never use it.

In 2020, we estimate that curl runs in roughly ten billion installations world wide.

**1.11 Why don't you update ca-bundle.crt**

In the cURL project we've decided not to attempt to keep this file updated (or even present) since deciding what to add to a ca cert bundle is an undertaking we've not been ready to accept, and the one we can get from Mozilla is perfectly fine so there's no need to duplicate that work.

Today, with many services performed over HTTPS, every operating system should come with a default ca cert bundle that can be deemed somewhat trustworthy and that collection (if reasonably updated) should be deemed to be a lot better than a private curl version.

If you want the most recent collection of ca certs that Mozilla Firefox uses, we recommend that you extract the collection yourself from Mozilla Firefox (by

running 'make ca-bundle), or by using our online service setup for this purpose: https://curl.se/docs/caextract.html

**1.12 I have a problem who can I chat with?**

There's a bunch of friendly people hanging out in the #curl channel on the IRC network irc.freenode.net. If you're polite and nice, chances are good that you can get -- or provide -- help instantly.

**1.13 curl's ECCN number?**

The US government restricts exports of software that contains or uses cryptography. When doing so, the Export Control Classification Number (ECCN) is used to identify the level of export control etc.

Apache Software Foundation gives a good explanation of ECCNs athttps://www.apache.org/dev/crypto.html

We believe curl's number might be ECCN 5D002, another possibility is 5D992. It seems necessary to write them (the authority that administers ECCN numbers), asking to confirm.

Comprehensible explanations of the meaning of such numbers and how to obtain them (resp.) are here

https://www.bis.doc.gov/licensing/exportingbasics.htmhttps://www.bis.doc.gov/lice

An incomprehensible description of the two numbers above is herehttps://www.bis.doc.gov/index.php/documents/new-encryption/1653-ccl5-pt2-3

**1.14 How do I submit my patch?**

We strongly encourage you to submit changes and improvements directly as "pull requests" on github: https://github.com/curl/curl/pulls

If you for any reason can't or won't deal with github, send your patch to the curl-library mailing list. We're many subscribers there and there are lots of people who can review patches, comment on them and "receive" them properly.

Lots of more details are found in the CONTRIBUTE.md and INTERNALS.md documents.

**1.15 How do I port libcurl to my OS?**

Here's a rough step-by-step:

1. copy a suitable lib/config-*.h file as a start to lib/config-[youros].h

2. edit lib/config-[youros].h to match your OS and setup

3. edit lib/curl_setup.h to include config-[youros].h when your OS is

```
 detected by the preprocessor, in the style others already exist
```

4. compile lib/*.c and make them into a library

## 2. Install Related Problems

### 2.1 configure fails when using static libraries

You may find that configure fails to properly detect the entire dependency chain of libraries when you provide static versions of the libraries that configure checks for.

The reason why static libraries is much harder to deal with is that for them we don't get any help but the script itself must know or check what more libraries that are needed (with shared libraries, that dependency "chain" is handled automatically). This is a very error-prone process and one that also tends to vary over time depending on the release versions of the involved components and may also differ between operating systems.

For that reason, configure does very little attempts to actually figure this out and you are instead encouraged to set LIBS and LDFLAGS accordingly when you invoke configure, and point out the needed libraries and set the necessary flags yourself.

### 2.2 Does curl work with other SSL libraries?

curl has been written to use a generic SSL function layer internally, and that SSL functionality can then be provided by one out of many different SSL backends.

curl can be built to use one of the following SSL alternatives: OpenSSL, libressl, BoringSSL, GnuTLS, wolfSSL, NSS, mbedTLS, MesaLink, Secure Transport (native iOS/OS X), Schannel (native Windows), GSKit (native IBM i), or BearSSL. They all have their pros and cons, and we try to maintain a comparison of them here: https://curl.se/docs/ssl-compared.html

### 2.4 Does curl support SOCKS (RFC 1928) ?

Yes, SOCKS 4 and 5 are supported.

## 3. Usage problems

### 3.1 curl: (1) SSL is disabled, https: not supported

If you get this output when trying to get anything from a https:// server, it means that the instance of curl/libcurl that you're using was built without support for this protocol.

This could've happened if the configure script that was run at build time couldn't find all libs and include files curl requires for SSL to work. If the configure script fails to find them, curl is simply built without SSL support.

To get the https:// support into a curl that was previously built but that reports that https:// is not supported, you should dig through the document and logs and check out why the configure script doesn't find the SSL libs and/or include files.

Also, check out the other paragraph in this FAQ labeled "configure doesn't find OpenSSL even when it is installed".

**3.2 How do I tell curl to resume a transfer?**

curl supports resumed transfers both ways on both FTP and HTTP. Try the -C option.

**3.3 Why doesn't my posting using -F work?**

You can't arbitrarily use -F or -d, the choice between -F or -d depends on the HTTP operation you need curl to do and what the web server that will receive your post expects.

If the form you're trying to submit uses the type 'multipart/form-data', then and only then you must use the -F type. In all the most common cases, you should use -d which then causes a posting with the type 'application/x-www-form-urlencoded'.

This is described in some detail in the MANUAL and TheArtOfHttpScripting documents, and if you don't understand it the first time, read it again before you post questions about this to the mailing list. Also, try reading through the mailing list archives for old postings and questions regarding this.

**3.4 How do I tell curl to run custom FTP commands?**

You can tell curl to perform optional commands both before and/or after a file transfer. Study the -Q/--quote option.

Since curl is used for file transfers, you don't normally use curl to perform FTP commands without transferring anything. Therefore you must always specify a URL to transfer to/from even when doing custom FTP commands, or use -I which implies the "no body" option sent to libcurl.

**3.5 How can I disable the Accept: */ * header?**

You can change all internally generated headers by adding a replacement with the -H/--header option. By adding a header with empty contents you safely disable that one. Use -H "Accept:" to disable that specific header.

**3.6 Does curl support ASP, XML, XHTML or HTML version Y?**

To curl, all contents are alike. It doesn't matter how the page was generated. It may be ASP, PHP, Perl, shell-script, SSI or plain HTML files. There's no difference to curl and it doesn't even know what kind of language that generated the page.

See also item 3.14 regarding javascript.

**3.7 Can I use curl to delete/rename a file through FTP?**

Yes. You specify custom FTP commands with -Q/--quote.

One example would be to delete a file after you have downloaded it:

```
curl -O ftp://download.com/coolfile -Q '-DELE coolfile'
```

or rename a file after upload:

```
curl -T infile ftp://upload.com/dir/ -Q "-RNFR infile" -Q "-RNTO
```

**3.8 How do I tell curl to follow HTTP redirects?**

curl does not follow so-called redirects by default. The Location: header that informs the client about this is only interpreted if you're using the -L/--location option. As in:

```
curl -L http://redirector.com
```

Not all redirects are HTTP ones, see 4.14

**3.9 How do I use curl in my favorite programming language?**

Many programming languages have interfaces/bindings that allow you to use curl without having to use the command line tool. If you are fluent in such a language, you may prefer to use one of these interfaces instead.

Find out more about which languages that support curl directly, and how to install and use them, in the libcurl section of the curl website: https://curl.se/libcurl/

All the various bindings to libcurl are made by other projects and people, outside of the cURL project. The cURL project itself only produces libcurl with its plain C API. If you don't find anywhere else to ask you can ask about bindings on the

curl-library list too, but be prepared that people on that list may not know anything about bindings.

In February 2019, there were interfaces available for the following languages: Ada95, Basic, C, C++, Ch, Cocoa, D, Delphi, Dylan, Eiffel, Euphoria, Falcon, Ferite, Gambas, glib/GTK+, Go, Guile, Harbour, Haskell, Java, Julia, Lisp, Lua, Mono, .NET, node.js, Object-Pascal, OCaml, Pascal, Perl, PHP, PostgreSQL, Python, R, Rexx, Ring, RPG, Ruby, Rust, Scheme, Scilab, S-Lang, Smalltalk, SP-Forth, SPL, Tcl, Visual Basic, Visual FoxPro, Q, wxwidgets, XBLite and Xoho. By the time you read this, additional ones may have appeared!

### 3.10 What about SOAP, WebDAV, XML-RPC or similar protocols over HTTP?

curl adheres to the HTTP spec, which basically means you can play with *any* protocol that is built on top of HTTP. Protocols such as SOAP, WEBDAV and XML-RPC are all such ones. You can use -X to set custom requests and -H to set custom headers (or replace internally generated ones).

Using libcurl is of course just as good and you'd just use the proper library options to do the same.

### 3.11 How do I POST with a different Content-Type?

You can always replace the internally generated headers with -H/--header. To make a simple HTTP POST with text/xml as content-type, do something like:

```
curl -d "datatopost" -H "Content-Type: text/xml" [URL]
```

### 3.12 Why do FTP-specific features over HTTP proxy fail?

Because when you use a HTTP proxy, the protocol spoken on the network will be HTTP, even if you specify a FTP URL. This effectively means that you normally can't use FTP-specific features such as FTP upload and FTP quote etc.

There is one exception to this rule, and that is if you can "tunnel through" the given HTTP proxy. Proxy tunneling is enabled with a special option (-p) and is generally not available as proxy admins usually disable tunneling to ports other than 443 (which is used for HTTPS access through proxies).

### 3.13 Why do my single/double quotes fail?

To specify a command line option that includes spaces, you might need to put the entire option within quotes. Like in:

curl -d " with spaces " url.com

or perhaps

curl -d ' with spaces ' url.com

Exactly what kind of quotes and how to do this is entirely up to the shell or command line interpreter that you are using. For most unix shells, you can more or less pick either single (') or double (") quotes. For Windows/DOS prompts I believe you're forced to use double (") quotes.

Please study the documentation for your particular environment. Examples in the curl docs will use a mix of both of these as shown above. You must adjust them to work in your environment.

Remember that curl works and runs on more operating systems than most single individuals have ever tried.

**3.14 Does curl support Javascript or PAC (automated proxy config)?**

Many web pages do magic stuff using embedded Javascript. curl and libcurl have no built-in support for that, so it will be treated just like any other contents.

.pac files are a netscape invention and are sometimes used by organizations to allow them to differentiate which proxies to use. The .pac contents is just a Javascript program that gets invoked by the browser and that returns the name of the proxy to connect to. Since curl doesn't support Javascript, it can't support .pac proxy configuration either.

Some workarounds usually suggested to overcome this Javascript dependency:

Depending on the Javascript complexity, write up a script that translates it to another language and execute that.

Read the Javascript code and rewrite the same logic in another language.

Implement a Javascript interpreter, people have successfully used the Mozilla Javascript engine in the past.

Ask your admins to stop this, for a static proxy setup or similar.

**3.15 Can I do recursive fetches with curl?**

No. curl itself has no code that performs recursive operations, such as those performed by wget and similar tools.

There exists wrapper scripts with that functionality (for example the curlmirror perl script), and you can write programs based on libcurl to do it, but the command line tool curl itself cannot.

**3.16 What certificates do I need when I use SSL?**

There are three different kinds of "certificates" to keep track of when we talk about using SSL-based protocols (HTTPS or FTPS) using curl or libcurl.

CLIENT CERTIFICATE

The server you communicate with may require that you can provide this in order to prove that you actually are who you claim to be. If the server doesn't require this, you don't need a client certificate.

A client certificate is always used together with a private key, and the private key has a pass phrase that protects it.

SERVER CERTIFICATE

The server you communicate with has a server certificate. You can and should verify this certificate to make sure that you are truly talking to the real server and not a server impersonating it.

CERTIFICATE AUTHORITY CERTIFICATE ("CA cert")

You often have several CA certs in a CA cert bundle that can be used to verify a server certificate that was signed by one of the authorities in the bundle. curl does not come with a CA cert bundle but most curl installs provide one. You can also override the default.

The server certificate verification process is made by using a Certificate Authority certificate ("CA cert") that was used to sign the server certificate. Server certificate verification is enabled by default in curl and libcurl and is often the reason for problems as explained in FAQ entry 4.12 and the SSLCERTS document (https://curl.se/docs/sslcerts.html). Server certificates that are "self-signed" or otherwise signed by a CA that you do not have a CA cert for, cannot be verified. If the verification during a connect fails, you are refused access. You then need to explicitly disable the verification to connect to the server.

**3.17 How do I list the root dir of an FTP server?**

There are two ways. The way defined in the RFC is to use an encoded slash in the first path part. List the "/tmp" dir like this:

```
curl ftp://ftp.sunet.se/%2ftmp/
```

or the not-quite-kosher-but-more-readable way, by simply starting the path section of the URL with a slash:

```
curl ftp://ftp.sunet.se//tmp/
```

**3.18 Can I use curl to send a POST/PUT and not wait for a response?**

No.

But you could easily write your own program using libcurl to do such stunts.

### 3.19 How do I get HTTP from a host using a specific IP address?

For example, you may be trying out a website installation that isn't yet in the DNS. Or you have a site using multiple IP addresses for a given host name and you want to address a specific one out of the set.

Set a custom Host: header that identifies the server name you want to reach but use the target IP address in the URL:

curl --header "Host: www.example.com" http://127.0.0.1/

You can also opt to add faked host name entries to curl with the --resolve option. That has the added benefit that things like redirects will also work properly. The above operation would instead be done as:

curl --resolve www.example.com:80:127.0.0.1 http://www.example.com/

### 3.20 How to SFTP from my user's home directory?

Contrary to how FTP works, SFTP and SCP URLs specify the exact directory to work with. It means that if you don't specify that you want the user's home directory, you get the actual root directory.

To specify a file in your user's home directory, you need to use the correct URL syntax which for SFTP might look similar to:

curl -O -u user:password sftp://example.com/~/file.txt

and for SCP it is just a different protocol prefix:

curl -O -u user:password scp://example.com/~/file.txt

### 3.21 Protocol xxx not supported or disabled in libcurl

When passing on a URL to curl to use, it may respond that the particular protocol is not supported or disabled. The particular way this error message is phrased is because curl doesn't make a distinction internally of whether a particular protocol is not supported (i.e. never got any code added that knows how to speak that protocol) or if it was explicitly disabled. curl can be built to only support a given set of protocols, and the rest would then be disabled or not supported.

Note that this error will also occur if you pass a wrongly spelled protocol part as in "htpt://example.com" or as in the less evident case if you prefix the protocol part with a space as in " http://example.com/".

### 3.22 curl -X gives me HTTP problems

In normal circumstances, -X should hardly ever be used.

By default you use curl without explicitly saying which request method to use when the URL identifies a HTTP transfer. If you just pass in a URL like "curl http://example.com" it will use GET. If you use -d or -F curl will use POST, -I will cause a HEAD and -T will make it a PUT.

If for whatever reason you're not happy with these default choices that curl does for you, you can override those request methods by specifying -X [WHATEVER]. This way you can for example send a DELETE by doing "curl -X DELETE [URL]".

It is thus pointless to do "curl -XGET [URL]" as GET would be used anyway. In the same vein it is pointless to do "curl -X POST -d data [URL]"... But you can make a fun and somewhat rare request that sends a request-body in a GET request with something like "curl -X GET -d data [URL]"

Note that -X doesn't actually change curl's behavior as it only modifies the actual string sent in the request, but that may of course trigger a different set of events.

Accordingly, by using -XPOST on a command line that for example would follow a 303 redirect, you will effectively prevent curl from behaving correctly. Be aware.

## 4. Running Problems

### 4.2 Why do I get problems when I use & or % in the URL?

In general unix shells, the & symbol is treated specially and when used, it runs the specified command in the background. To safely send the & as a part of a URL, you should quote the entire URL by using single (') or double (") quotes around it. Similar problems can also occur on some shells with other characters, including ?*!$~(){}<>\|;`. When in doubt, quote the URL.

An example that would invoke a remote CGI that uses &-symbols could be:

```
curl 'http://www.altavista.com/cgi-bin/query?text=yes&q=curl'
```

In Windows, the standard DOS shell treats the percent sign specially and you need to use TWO percent signs for each single one you want to use in the URL.

If you want a literal percent sign to be part of the data you pass in a POST using -d/--data you must encode it as '%25' (which then also needs the percent sign doubled on Windows machines).

#### 4.3 How can I use {, }, [ or ] to specify multiple URLs?

Because those letters have a special meaning to the shell, to be used in a URL specified to curl you must quote them.

An example that downloads two URLs (sequentially) would be:

curl '{curl,www}.haxx.se'

To be able to use those characters as actual parts of the URL (without using them for the curl URL "globbing" system), use the -g/--globoff option:

curl -g 'www.site.com/weirdname[].html'

#### 4.4 Why do I get downloaded data even though the web page doesn't exist?

curl asks remote servers for the page you specify. If the page doesn't exist at the server, the HTTP protocol defines how the server should respond and that means that headers and a "page" will be returned. That's simply how HTTP works.

By using the --fail option you can tell curl explicitly to not get any data if the HTTP return code doesn't say success.

#### 4.5 Why do I get return code XXX from a HTTP server?

RFC2616 clearly explains the return codes. This is a short transcript. Go read the RFC for exact details:

##### 4.5.1 "400 Bad Request"

The request could not be understood by the server due to malformed syntax. The client SHOULD NOT repeat the request without modifications.

##### 4.5.2 "401 Unauthorized"

The request requires user authentication.

##### 4.5.3 "403 Forbidden"

The server understood the request, but is refusing to fulfill it. Authorization will not help and the request SHOULD NOT be repeated.

##### 4.5.4 "404 Not Found"

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent.

##### 4.5.5 "405 Method Not Allowed"

The method specified in the Request-Line is not allowed for the resource identified by the Request-URI. The response MUST include an Allow header containing a list of valid methods for the requested resource.

#### 4.5.6 "301 Moved Permanently"

If you get this return code and an HTML output similar to this:

```
<H1>Moved Permanently</H1> The document has moved <A
HREF="http://same_url_now_with_a_trailing_slash/">here</A>.
```

it might be because you requested a directory URL but without the trailing slash. Try the same operation again *with* the trailing URL, or use the -L/--location option to follow the redirection.

### 4.6 Can you tell me what error code 142 means?

All curl error codes are described at the end of the man page, in the section called "EXIT CODES".

Error codes that are larger than the highest documented error code means that curl has exited due to a crash. This is a serious error, and we appreciate a detailed bug report from you that describes how we could go ahead and repeat this!

### 4.7 How do I keep user names and passwords secret in curl command lines?

This problem has two sides:

The first part is to avoid having clear-text passwords in the command line so that they don't appear in 'ps' outputs and similar. That is easily avoided by using the "-K" option to tell curl to read parameters from a file or stdin to which you can pass the secret info. curl itself will also attempt to "hide" the given password by blanking out the option - this doesn't work on all platforms.

To keep the passwords in your account secret from the rest of the world is not a task that curl addresses. You could of course encrypt them somehow to at least hide them from being read by human eyes, but that is not what anyone would call security.

Also note that regular HTTP (using Basic authentication) and FTP passwords are sent as cleartext across the network. All it takes for anyone to fetch them is to listen on the network. Eavesdropping is very easy. Use more secure authentication methods (like Digest, Negotiate or even NTLM) or consider the SSL-based alternatives HTTPS and FTPS.

### 4.8 I found a bug!

It is not a bug if the behavior is documented. Read the docs first. Especially check out the KNOWN_BUGS file, it may be a documented bug!

If it is a problem with a binary you've downloaded or a package for your particular platform, try contacting the person who built the package/archive you have.

If there is a bug, read the BUGS document first. Then report it as described in there.

**4.9 curl can't authenticate to the server that requires NTLM?**

NTLM support requires OpenSSL, GnuTLS, mbedTLS, NSS, Secure Transport, or Microsoft Windows libraries at build-time to provide this functionality.

NTLM is a Microsoft proprietary protocol. Proprietary formats are evil. You should not use such ones.

**4.10 My HTTP request using HEAD, PUT or DELETE doesn't work!**

Many web servers allow or demand that the administrator configures the server properly for these requests to work on the web server.

Some servers seem to support HEAD only on certain kinds of URLs.

To fully grasp this, try the documentation for the particular server software you're trying to interact with. This is not anything curl can do anything about.

**4.11 Why do my HTTP range requests return the full document?**

Because the range may not be supported by the server, or the server may choose to ignore it and return the full document anyway.

**4.12 Why do I get "certificate verify failed" ?**

When you invoke curl and get an error 60 error back it means that curl couldn't verify that the server's certificate was good. curl verifies the certificate using the CA cert bundle and verifying for which names the certficiate has been granted.

To completely disable the certficiate verification, use -k. This does however enable man-in-the-middle attacks and makes the transfer INSECURE. We strongly advice against doing this for more than experiments.

If you get this failure with a CA cert bundle installed and used, the server's certificate might not be signed by one of the CA's in yout CA store. It might for example be self-signed. You then correct this problem by obtaining a valid CA cert for the server. Or again, decrease the security by disabling this check.

At times, you find that the verification works in your favorite browser but fails in curl. When this happens, the reason is usually that the server sends an incomplete cert chain. The server is mandated to send all "intermediate certificates" but doesn't. This typically works with browsers anyway since they A) cache such certs and B) supports AIA which downloads such missing certificates on demand. This is a server misconfiguration. A good way to figure out if this is the case it to use the SSL Labs server test and check the certificate chain: https://www.ssllabs.com/ssltest/

Details are also in the SSLCERTS.md document, found online here:https://curl.se/docs/sslcerts.html

**4.13 Why is curl -R on Windows one hour off?**

Since curl 7.53.0 this issue should be fixed as long as curl was built with any modern compiler that allows for a 64-bit curl_off_t type. For older compilers or prior curl versions it may set a time that appears one hour off. This happens due to a flaw in how Windows stores and uses file modification times and it is not easily worked around. For more details read this:https://www.codeproject.com/Articles/1144/Beating-the-Daylight-Savings-Time-bug-and-getting

**4.14 Redirects work in browser but not with curl!**

curl supports HTTP redirects well (see item 3.8). Browsers generally support at least two other ways to perform redirects that curl does not:

Meta tags. You can write a HTML tag that will cause the browser to redirect to another given URL after a certain time.

Javascript. You can write a Javascript program embedded in a HTML page that redirects the browser to another given URL.

There is no way to make curl follow these redirects. You must either manually figure out what the page is set to do, or write a script that parses the results and fetches the new URL.

**4.15 FTPS doesn't work**

curl supports FTPS (sometimes known as FTP-SSL) both implicit and explicit mode.

When a URL is used that starts with FTPS://, curl assumes implicit SSL on the control connection and will therefore immediately connect and try to speak SSL. FTPS:// connections default to port 990.

To use explicit FTPS, you use a FTP:// URL and the --ftp-ssl option (or one of its related flavors). This is the most common method, and the one mandated by

RFC4217. This kind of connection will then of course use the standard FTP port 21 by default.

**4.16 My HTTP POST or PUT requests are slow!**

libcurl makes all POST and PUT requests (except for POST requests with a very tiny request body) use the "Expect: 100-continue" header. This header allows the server to deny the operation early so that libcurl can bail out before having to send any data. This is useful in authentication cases and others.

However, many servers don't implement the Expect: stuff properly and if the server doesn't respond (positively) within 1 second libcurl will continue and send off the data anyway.

You can disable libcurl's use of the Expect: header the same way you disable any header, using -H / CURLOPT_HTTPHEADER, or by forcing it to use HTTP 1.0.

**4.17 Non-functional connect timeouts**

In most Windows setups having a timeout longer than 21 seconds make no difference, as it will only send 3 TCP SYN packets and no more. The second packet sent three seconds after the first and the third six seconds after the second. No more than three packets are sent, no matter how long the timeout is set.

See option TcpMaxConnectRetransmissions on this page:https://support.microsoft.com/en-us/kb/175523/en-us

Also, even on non-Windows systems there may run a firewall or anti-virus software or similar that accepts the connection but does not actually do anything else. This will make (lib)curl to consider the connection connected and thus the connect timeout won't trigger.

**4.18 file:// URLs containing drive letters (Windows, NetWare)**

When using curl to try to download a local file, one might use a URL in this format:

file://D:/blah.txt

You'll find that even if D:\blah.txt does exist, curl returns a 'file not found' error.

According to RFC 1738 (https://www.ietf.org/rfc/rfc1738.txt), file:// URLs must contain a host component, but it is ignored by most implementations. In the above example, 'D:' is treated as the host component, and is taken away. Thus, curl tries to open '/blah.txt'. If your system is installed to drive C:, that will resolve to 'C:\blah.txt', and if that doesn't exist you will get the not found error.

To fix this problem, use file:// URLs with *three* leading slashes:

file:///D:/blah.txt

Alternatively, if it makes more sense, specify 'localhost' as the host component:

file://localhost/D:/blah.txt

In either case, curl should now be looking for the correct file.

**4.19 Why doesn't curl return an error when the network cable is unplugged?**

Unplugging a cable is not an error situation. The TCP/IP protocol stack was designed to be fault tolerant, so even though there may be a physical break somewhere the connection shouldn't be affected, just possibly delayed. Eventually, the physical break will be fixed or the data will be re-routed around the physical problem through another path.

In such cases, the TCP/IP stack is responsible for detecting when the network connection is irrevocably lost. Since with some protocols it is perfectly legal for the client to wait indefinitely for data, the stack may never report a problem, and even when it does, it can take up to 20 minutes for it to detect an issue. The curl option --keepalive-time enables keep-alive support in the TCP/IP stack which makes it periodically probe the connection to make sure it is still available to send data. That should reliably detect any TCP/IP network failure.

But even that won't detect the network going down before the TCP/IP connection is established (e.g. during a DNS lookup) or using protocols that don't use TCP. To handle those situations, curl offers a number of timeouts on its own. --speed-limit/--speed-time will abort if the data transfer rate falls too low, and --connect-timeout and --max-time can be used to put an overall timeout on the connection phase or the entire transfer.

A libcurl-using application running in a known physical environment (e.g. an embedded device with only a single network connection) may want to act immediately if its lone network connection goes down. That can be achieved by having the application monitor the network connection on its own using an OS-specific mechanism, then signaling libcurl to abort (see also item 5.13).

**4.20 curl doesn't return error for HTTP non-200 responses!**

Correct. Unless you use -f (--fail).

When doing HTTP transfers, curl will perform exactly what you're asking it to do and if successful it will not return an error. You can use curl to test your web server's "file not found" page (that gets 404 back), you can use it to check your authentication protected web pages (that gets a 401 back) and so on.

The specific HTTP response code does not constitute a problem or error for curl. It simply sends and delivers HTTP as you asked and if that worked, everything is fine and dandy. The response code is generally providing more higher level error information that curl doesn't care about. The error was not in the HTTP transfer.

If you want your command line to treat error codes in the 400 and up range as errors and thus return a non-zero value and possibly show an error message, curl has a dedicated option for that: -f (CURLOPT_FAILONERROR in libcurl speak).

You can also use the -w option and the variable %{response_code} to extract the exact response code that was returned in the response.

## 5. libcurl Issues

#### 5.1 Is libcurl thread-safe?

Yes.

We have written the libcurl code specifically adjusted for multi-threaded programs. libcurl will use thread-safe functions instead of non-safe ones if your system has such. Note that you must never share the same handle in multiple threads.

There may be some exceptions to thread safety depending on how libcurl was built. Please review the guidelines for thread safety to learn more:https://curl.se/libcurl/c/threadsafe.html

#### 5.2 How can I receive all data into a large memory chunk?

[ See also the examples/getinmemory.c source ]

You are in full control of the callback function that gets called every time there is data received from the remote server. You can make that callback do whatever you want. You do not have to write the received data to a file.

One solution to this problem could be to have a pointer to a struct that you pass to the callback function. You set the pointer using the CURLOPT_WRITEDATA option. Then that pointer will be passed to the callback instead of a FILE * to a file:

```
/* imaginary struct */
struct MemoryStruct {
char *memory;
size_t size;
};
```

```
/* imaginary callback function */
size_t
WriteMemoryCallback(void *ptr, size_t size, size_t nmemb, void *d
{
size_t realsize = size * nmemb;
struct MemoryStruct *mem = (struct MemoryStruct *)data;

mem->memory = (char *)realloc(mem->memory, mem->size + realsize +
if (mem->memory) {
memcpy(&(mem->memory[mem->size]), ptr, realsize);
mem->size += realsize;
mem->memory[mem->size] = 0;
}
return realsize;
}
```

**5.3 How do I fetch multiple files with libcurl?**

libcurl has excellent support for transferring multiple files. You should just repeatedly set new URLs with curl_easy_setopt() and then transfer it with curl_easy_perform(). The handle you get from curl_easy_init() is not only reusable, but you're even encouraged to reuse it if you can, as that will enable libcurl to use persistent connections.

**5.4 Does libcurl do Winsock initialization on win32 systems?**

Yes, if told to in the curl_global_init() call.

**5.5 Does CURLOPT_WRITEDATA and CURLOPT_READDATA work on win32 ?**

Yes, but you cannot open a FILE * and pass the pointer to a DLL and have that DLL use the FILE * (as the DLL and the client application cannot access each others' variable memory areas). If you set CURLOPT_WRITEDATA you must also use CURLOPT_WRITEFUNCTION as well to set a function that writes the file, even if that simply writes the data to the specified FILE *. Similarly, if you use CURLOPT_READDATA you must also specify CURLOPT_READFUNCTION.

**5.6 What about Keep-Alive or persistent connections?**

curl and libcurl have excellent support for persistent connections when transferring several files from the same server. curl will attempt to reuse connections for all URLs specified on the same command line/config file, and libcurl will reuse connections for all transfers that are made using the same libcurl handle.

When you use the easy interface the connection cache is kept within the easy handle. If you instead use the multi interface, the connection cache will be kept

within the multi handle and will be shared among all the easy handles that are used within the same multi handle.

**5.7 Link errors when building libcurl on Windows!**

You need to make sure that your project, and all the libraries (both static and dynamic) that it links against, are compiled/linked against the same run time library.

This is determined by the /MD, /ML, /MT (and their corresponding /M?d) options to the command line compiler. /MD (linking against MSVCRT dll) seems to be the most commonly used option.

When building an application that uses the static libcurl library, you must add -DCURL_STATICLIB to your CFLAGS. Otherwise the linker will look for dynamic import symbols. If you're using Visual Studio, you need to instead add CURL_STATICLIB in the "Preprocessor Definitions" section.

If you get linker error like "unknown symbol __imp__curl_easy_init ..." you have linked against the wrong (static) library. If you want to use the libcurl.dll and import lib, you don't need any extra CFLAGS, but use one of the import libraries below. These are the libraries produced by the various lib/Makefile.* files:

```
Target: static lib. import lib for libcurl*.dll.
------------------------------------------------------------
MingW: libcurl.a libcurldll.a
MSVC (release): libcurl.lib libcurl_imp.lib
MSVC (debug): libcurld.lib libcurld_imp.lib
Borland: libcurl.lib libcurl_imp.lib
```

**5.8 libcurl.so.X: open failed: No such file or directory**

This is an error message you might get when you try to run a program linked with a shared version of libcurl and your run-time linker (ld.so) couldn't find the shared library named libcurl.so.X. (Where X is the number of the current libcurl ABI, typically 3 or 4).

You need to make sure that ld.so finds libcurl.so.X. You can do that multiple ways, and it differs somewhat between different operating systems, but they are usually:

* Add an option to the linker command line that specify the hard-coded path

the run-time linker should check for the lib (usually -R)

* Set an environment variable (LD_LIBRARY_PATH for example) where ld.so

should check for libs

* Adjust the system's config to check for libs in the directory where you've

put the dir (like Linux's /etc/ld.so.conf)

'man ld.so' and 'man ld' will tell you more details

**5.9 How does libcurl resolve host names?**

libcurl supports a large a number of different name resolve functions. One of them is picked at build-time and will be used unconditionally. Thus, if you want to change name resolver function you must rebuild libcurl and tell it to use a different function.

- The non-IPv6 resolver that can use one of four different host name resolve calls (depending on what your system supports):

```
A - gethostbyname()
B - gethostbyname_r() with 3 arguments
C - gethostbyname_r() with 5 arguments
D - gethostbyname_r() with 6 arguments
```

- The IPv6-resolver that uses getaddrinfo()

- The c-ares based name resolver that uses the c-ares library for resolves.

Using this offers asynchronous name resolves.

- The threaded resolver (default option on Windows). It uses:

```
A - gethostbyname() on plain IPv4 hosts
B - getaddrinfo() on IPv6 enabled hosts
```

Also note that libcurl never resolves or reverse-lookups addresses given as pure numbers, such as 127.0.0.1 or ::1.

**5.10 How do I prevent libcurl from writing the response to stdout?**

libcurl provides a default built-in write function that writes received data to stdout. Set the CURLOPT_WRITEFUNCTION to receive the data, or possibly set CURLOPT_WRITEDATA to a different FILE * handle.

**5.11 How do I make libcurl not receive the whole HTTP response?**

You make the write callback (or progress callback) return an error and libcurl will then abort the transfer.

**5.12 Can I make libcurl fake or hide my real IP address?**

No. libcurl operates on a higher level. Besides, faking IP address would imply sending IP packets with a made-up source address, and then you normally get a problem with receiving the packet sent back as they would then not be routed to you!

If you use a proxy to access remote sites, the sites will not see your local IP address but instead the address of the proxy.

Also note that on many networks NATs or other IP-munging techniques are used that makes you see and use a different IP address locally than what the remote server will see you coming from. You may also consider usinghttps://www.torproject.org/ .

**5.13 How do I stop an ongoing transfer?**

With the easy interface you make sure to return the correct error code from one of the callbacks, but none of them are instant. There is no function you can call from another thread or similar that will stop it immediately. Instead, you need to make sure that one of the callbacks you use returns an appropriate value that will stop the transfer. Suitable callbacks that you can do this with include the progress callback, the read callback and the write callback.

If you're using the multi interface, you can also stop a transfer by removing the particular easy handle from the multi stack at any moment you think the transfer is done or when you wish to abort the transfer.

**5.14 Using C++ non-static functions for callbacks?**

libcurl is a C library, it doesn't know anything about C++ member functions.

You can overcome this "limitation" with relative ease using a static member function that is passed a pointer to the class:

```
// f is the pointer to your object.
static size_t YourClass::func(void *buffer, size_t sz, size_t n,
{
// Call non-static member function.
static_cast<YourClass*>(f)->nonStaticFunction();
}

// This is how you pass pointer to the static function:
curl_easy_setopt(hcurl, CURLOPT_WRITEFUNCTION, YourClass::func);
curl_easy_setopt(hcurl, CURLOPT_WRITEDATA, this);
```

**5.15 How do I get an FTP directory listing?**

If you end the FTP URL you request with a slash, libcurl will provide you with a directory listing of that given directory. You can also set

CURLOPT_CUSTOMREQUEST to alter what exact listing command libcurl would use to list the files.

The follow-up question tends to be how is a program supposed to parse the directory listing. How does it know what's a file and what's a dir and what's a symlink etc. If the FTP server supports the MLSD command then it will return data in a machine-readable format that can be parsed for type. The types are specified by RFC3659 section 7.5.1. If MLSD is not supported then you have to work with what you're given. The LIST output format is entirely at the server's own liking and the NLST output doesn't reveal any types and in many cases doesn't even include all the directory entries. Also, both LIST and NLST tend to hide unix-style hidden files (those that start with a dot) by default so you need to do "LIST -a" or similar to see them.

Example - List only directories. ftp.funet.fi supports MLSD and ftp.kernel.org does not:

```
curl -s ftp.funet.fi/pub/ -X MLSD |  perl -lne 'print if s/(?:^|;

curl -s ftp.kernel.org/pub/linux/kernel/ |  perl -lne 'print if s
```

If you need to parse LIST output in libcurl one such existing list parser is available at https://cr.yp.to/ftpparse.html Versions of libcurl since 7.21.0 also provide the ability to specify a wildcard to download multiple files from one FTP directory.

**5.16 I want a different time-out!**

Time and time again users realize that CURLOPT_TIMEOUT and CURLOPT_CONNECTIMEOUT are not sufficiently advanced or flexible to cover all the various use cases and scenarios applications end up with.

libcurl offers many more ways to time-out operations. A common alternative is to use the CURLOPT_LOW_SPEED_LIMIT and CURLOPT_LOW_SPEED_TIME options to specify the lowest possible speed to accept before to consider the transfer timed out.

The most flexible way is by writing your own time-out logic and using CURLOPT_XFERINFOFUNCTION (perhaps in combination with other callbacks) and use that to figure out exactly when the right condition is met when the transfer should get stopped.

**5.17 Can I write a server with libcurl?**

No. libcurl offers no functions or building blocks to build any kind of internet protocol server. libcurl is only a client-side library. For server libraries, you need to continue your search elsewhere but there exist many good open source ones out there for most protocols you could possibly want a server for. And there are

really good stand-alone ones that have been tested and proven for many years. There's no need for you to reinvent them!

#### 5.18 Does libcurl use threads?

Put simply: no, libcurl will execute in the same thread you call it in. All callbacks will be called in the same thread as the one you call libcurl in.

If you want to avoid your thread to be blocked by the libcurl call, you make sure you use the non-blocking API which will do transfers asynchronously - but still in the same single thread.

libcurl will potentially internally use threads for name resolving, if it was built to work like that, but in those cases it'll create the child threads by itself and they will only be used and then killed internally by libcurl and never exposed to the outside.

## 6. License Issues

curl and libcurl are released under a MIT/X derivate license. The license is very liberal and should not impose a problem for your project. This section is just a brief summary for the cases we get the most questions. (Parts of this section was much enhanced by Bjorn Reese.)

We are not lawyers and this is not legal advice. You should probably consult one if you want true and accurate legal insights without our prejudice. Note especially that this section concerns the libcurl license only; compiling in features of libcurl that depend on other libraries (e.g. OpenSSL) may affect the licensing obligations of your application.

#### 6.1 I have a GPL program, can I use the libcurl library?

Yes!

Since libcurl may be distributed under the MIT/X derivate license, it can be used together with GPL in any software.

#### 6.2 I have a closed-source program, can I use the libcurl library?

Yes!

libcurl does not put any restrictions on the program that uses the library.

#### 6.3 I have a BSD licensed program, can I use the libcurl library?

Yes!

libcurl does not put any restrictions on the program that uses the library.

**6.4 I have a program that uses LGPL libraries, can I use libcurl?**

Yes!

The LGPL license doesn't clash with other licenses.

**6.5 Can I modify curl/libcurl for my program and keep the changes secret?**

Yes!

The MIT/X derivate license practically allows you to do almost anything with the sources, on the condition that the copyright texts in the sources are left intact.

**6.6 Can you please change the curl/libcurl license to XXXX?**

No.

We have carefully picked this license after years of development and discussions and a large amount of people have contributed with source code knowing that this is the license we use. This license puts the restrictions we want on curl/libcurl and it does not spread to other programs or libraries that use it. It should be possible for everyone to use libcurl or curl in their projects, no matter what license they already have in use.

**6.7 What are my obligations when using libcurl in my commercial apps?**

Next to none. All you need to adhere to is the MIT-style license (stated in the COPYING file) which basically says you have to include the copyright notice in "all copies" and that you may not use the copyright holder's name when promoting your software.

You do not have to release any of your source code.

You do not have to reveal or make public any changes to the libcurl source code.

You do not have to broadcast to the world that you are using libcurl within your app.

All we ask is that you disclose "the copyright notice and this permission notice" somewhere. Most probably like in the documentation or in the section where other third party dependencies already are mentioned and acknowledged.

As can be seen here: https://curl.se/docs/companies.html and elsewhere, more and more companies are discovering the power of libcurl and take advantage of it even in commercial environments.

## 7. PHP/CURL Issues

### 7.1 What is PHP/CURL?

The module for PHP that makes it possible for PHP programs to access curl-functions from within PHP.

In the cURL project we call this module PHP/CURL to differentiate it from curl the command line tool and libcurl the library. The PHP team however does not refer to it like this (for unknown reasons). They call it plain CURL (often using all caps) or sometimes ext/curl, but both cause much confusion to users which in turn gives us a higher question load.

### 7.2 Who wrote PHP/CURL?

PHP/CURL was initially written by Sterling Hughes.

### 7.3 Can I perform multiple requests using the same handle?

Yes - at least in PHP version 4.3.8 and later (this has been known to not work in earlier versions, but the exact version when it started to work is unknown to me).

After a transfer, you just set new options in the handle and make another transfer. This will make libcurl re-use the same connection if it can.

### 7.4 Does PHP/CURL have dependencies?

PHP/CURL is a module that comes with the regular PHP package. It depends on and uses libcurl, so you need to have libcurl installed properly before PHP/CURL can be used.