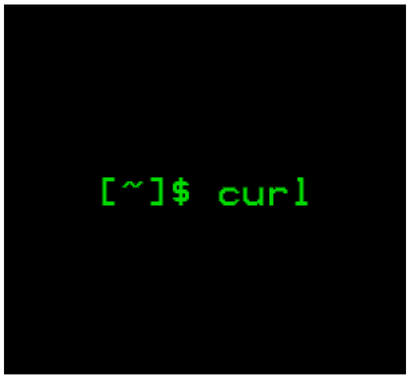
Linux curl command

Updated: 04/26/2017 by Computer Hope



About curl

curl is a tool to transfer data from or to a server, using one of the supported protocols (HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, DICT, TELNET, LDAP or FILE). The command is designed to work without user interaction.

curl offers proxy support, user authentication, FTP uploading, HTTP posting, SSL connections, cookies, file transfer resume, Metalink, and many other features, listed below.

Progress Meter

curl normally displays a progress meter during operations, indicating the amount of transferred data, transfer speeds and estimated time left, etc.

curl displays this data to the terminal by default, so if you invoke **curl** to do an operation and it is about to write data to the terminal, it disables the progress meter as otherwise it would mess up the output mixing progress meter and response data.

If you want a progress meter for HTTP POST or PUT requests, you need to redirect the response output to a file, using shellredirect (**>**), **-o** *[file]* or similar.

It is not the same case for FTP upload as that operation does not spit out any response data to the terminal.

If you prefer a progress "bar" instead of the regular meter, **-#** is your friend.

About the URL

The URL syntax is protocol-dependent. You'll find a detailed description in RFC 3986.

You can specify multiple URLs or parts of URLs by writing part sets within braces as in:

http://site.{one,two,three}.com

or you can get sequences of alphanumeric series by using **[]** as in:

ftp://ftp.numericals.com/file[1-100].txt
ftp://ftp.numericals.com/file[001-100].txt
ftp://ftp.letters.com/file[a-z].txt

Nested sequences are not supported, but you can use several ones next to each other:

http://any.org/archive[1996-1999]/vol[1-4]/part{a,b,c}.html

You can specify any amount of URLs on the command line. They will be fetched in a sequential manner in the specified order.

You can specify a step counter for the ranges to get every Nth number or letter:

http://www.numericals.com/file[1-100:10].txt
http://www.letters.com/file[a-z:2].txt

If you specify URL without **protocol://** prefix, **curl** will attempt to guess what protocol you might want. It will then default to HTTP but try other protocols based on often-used host name prefixes. For example, for host names starting with "ftp." **curl** will assume you want to speak FTP.

curl will do its best to use what you pass to it as a URL. It is not trying to validate it as a syntactically correct URL by any means but is instead very liberal with what it accepts.

curl will attempt to re-use connections for multiple file transfers, so that getting many files from the same server will not do multiple connects / handshakes. This improves speed. Of course this is only done on files specified on a single command line and cannot be used between separate **curl** invokes.

curl syntax

```
curl [options] [URL...]
```

Options

-a, **--append**	(**FTP**/**SFTP**) When used in an FTP upload, this will tell **curl** to append to the target file instead of overwriting it. If the file doesn't exist, it will be created. Note that this option is ignored by some SSH servers, including OpenSSH.
-A, **--user-agent** *<agent string>*	(**HTTP**) Specify the User-Agent string to send to the HTTP server. Some CGIs fail if the agent string is not set to "**Mozilla/4.0**". To encode blanks in the string, surround the string with single quote marks. This value can also be set with the **-H**/**--header** option. If this option is set more than once, the last one will be the one that's used.
--anyauth	(**HTTP**) Tells **curl** to figure out authentication method by itself, and use the most secure method the remote site claims it supports. This is done by first making a request and checking the response-headers, thus possibly inducing an network round-trip. This is used instead of setting a specific authentication method, which you can do with **--basic**, **--digest**, **--ntlm**, and **--negotiate**. Note that using **--anyauth** is not recommended if you do uploads from stdin, since it may require data to be sent twice and then the client must be able to rewind. If the need should arise when uploading from stdin, the upload operation will fail.

-b, **--cookie** *<name=data>*	(**HTTP**) Pass the data to the HTTP server as a cookie. It is expected to be the data previously received from the server in a "**Set-Cookie:**" line. The data should be in the format "**NAME1=VALUE1; NAME2=VALUE2**". If no '**=**' (equals) character is used in the line, it is treated as a filename to use to read previously stored cookie lines from, which should be used in this session if they match. Using this method also activates the "cookie parser" which will make curl record incoming cookies too, which may be handy if you're using this in combination with the **--location** option. The file format of the file to read cookies from should be plain HTTP headers or the Netscape/Mozilla cookie file format. **NOTE:** the file specified with **-b**/**--cookie** is only used as input. No cookies will be stored in the file. To store cookies, use the **-c**/**--cookie-jar** option, or you can save the HTTP headers to a file using **-D**/**--dump-header**. If this option is set more than once, the last occurrence will be the option that's used.
-B, **--use-ascii**	(**FTP/LDAP**) Enable ASCII transfer. For **FTP**, this can also be enforced by using an URL that ends with "**;Type=A**". This option causes data sent to stdout to be in text mode for win32 systems. If this option is used twice, the second one will disable ASCII usage.
--basic	(**HTTP**) Tells **curl** to use HTTP Basic authentication. This is the default and this option is usually pointless, unless you use it to override a previously set option that sets a different authentication method (such as **--ntlm**, **--digest** and **--negotiate**).
--ciphers *<list of ciphers>*	(**SSL**) Specifies which ciphers to use in the connection. The ciphers listed must be valid. You can read up on SSL cipher list details at openssl.org. NSS ciphers are done differently than OpenSSL and GnuTLS. The full list of NSS ciphers is in the NSSCipherSuite entry at this URL: http://git.fedorahosted.org/cgit/mod_nss.git/plain/docs/mod_nss.html#Directives. If this option is used several times, the last one will override the others.
--compressed	(**HTTP**) Request a compressed response using one of the algorithms **curl** supports, and return the uncompressed document. If this option is used and the server sends an unsupported encoding, Curl will report an error.
--connect-timeout *<seconds>*	Maximum time in seconds that the connection to the server may take. This only limits the connection phase; once curl has connected this option no longer applies. Since 7.32.0, this option accepts decimal values, but the actual timeout will decrease in accuracy as the specified timeout increases in decimal precision. See also the **-m**/**--max-time** option. If this option is used several times, the last one will be used.

-c, **--cookie-jar** <*file name*>

(**HTTP**) Specify to which file you want **curl** to write all cookies after a completed operation. Curl writes all cookies previously read from a specified file as well as all cookies received from remote server(s). If no cookies are known, no file will be written. The file will be written using the Netscape cookie file format. If you set the file name to a single dash ("**-**"), the cookies will be written to stdout.

This command line option will activate the cookie engine that makes **curl** record and use cookies. Another way to activate it is to use the **-b**/**--cookie option**.

NOTE: If the cookie jar can't be created or written to, the whole curl operation won't fail or even report an error. If **-v** is specified a warning will be displayed, but that is the only visible feedback you get about this possibly fatal situation.

If this option is used several times, the last specified file name will be used.

-C, **--continue-at** <*offset*>

Continue/Resume a previous file transfer at the given *offset*. The given offset is the exact number of bytes that will be skipped, counted from the beginning of the source file before it is transferred to the destination. If used with uploads, the ftp server command SIZE will not be used by curl.

Use "**-C -**" to tell curl to automatically find out where/how to resume the transfer. It then uses the given output/input files to figure that out.

If this option is used several times, the last one will be used.

--create-dirs

When used in conjunction with the **-o** option, **curl** will create the necessary local directory hierarchy as needed. This option creates the dirs mentioned with the **-o** option, nothing else. If the **-o** file name uses no directory or if the directories it mentions already exist, no directories will be created.

To create remote directories when using FTP or SFTP, try **--ftp-create-dirs**.

--crlf

(**FTP**) Convert **LF** to **CRLF** in upload. Useful for MVS (OS/390).

--crlfile <*file*>

(**HTTPS/FTPS**) Provide a file using PEM format with a Certificate Revocation List that may specify peer certificates that are to be considered revoked.

If this option is used several times, the last one will be used.

(Added in 7.19.7)

-d, **--data** <*data*>

(**HTTP**) Sends the specified data in a **POST** request to the HTTP server, in a way that can emulate as if a user has filled in an HTML form and pressed the submit button. Note that the data is sent exactly as specified with no extra processing (with all newlines cut off). The data is expected to be "url-encoded". This will cause **curl** to pass the data to the server using the content-type **application/x-www-form-urlencoded**. Compare to **-F**/**--form**. If this option is used more than once on the same command line, the data pieces specified will be merged together with a separating "**&**" character. Thus, using **'-d name=daniel -d skill=lousy'** would generate a POST chunk that looks like **'name=daniel&skill=lousy'**.

If you start the data with the "**@**" character, the rest should be a file name to read the data from, or "**-**" (dash) if you want curl to read the data from stdin. The contents of the file must already be url-encoded. Multiple files can also be specified. Posting data from a file named 'foobar' would thus be done with "**--data @foo-bar**".

-d/**--data** is the same as **--data-ascii**. To post data purely binary, you should instead use the **--data-binary** option. To URL-encode the value of a form field you may use **--data-urlencode**.

If this option is used several times, the ones following the first will append data.

--data-ascii *<data>*

(**HTTP**) This is an alias for the **-d**/**--data** option.

If this option is used several times, the ones following the first will append data.

--data-binary *<data>*

(**HTTP**) This posts data exactly as specified with no extra processing whatsoever.

If you start the data with the character **@**, the rest should be a filename. Data is posted in a similar manner as **--data-ascii** does, except that newlines are preserved and conversions are never done.

If this option is used several times, the ones following the first will append data as described in **-d**, **--data**.

--data-urlencode *<data>*

(**HTTP**) This posts data, similar to the other --data options with the exception that this performs URL-encoding. (Added in 7.18.0)

To be CGI-compliant, the <*data*> part should begin with a name followed by a separator and a content specification. The <*data*> part can be passed to **curl** using one of the following syntaxes:

content

This will make curl URL-encode the content and pass that on. Just be careful so that the content doesn't contain any **=** or **@** symbols, as that will then make the syntax match one of the other cases below!

=content

This will make curl URL-encode the content and pass that on. The preceding **=** symbol is not included in the data.

name=content

This will make curl URL-encode the content part and pass that on. Note that the name part is expected to be URL-encoded already.

@filename

This will make curl load data from the given file (including any newlines), URL-encode that data and pass it on in the POST.

name@filename

This will make curl load data from the given file (including any newlines), URL-encode that data and pass it on in the POST. The name part gets an equal sign appended, resulting in **name=urlencoded-file-content**. Note that the name is expected to be URL-encoded already.

--delegation *LEVEL*

Set *LEVEL* to tell the server what it is allowed to delegate when it comes to user credentials. Used with GSS/kerberos.

none

Don't allow any delegation.

policy

Delegates if and only if the OK-AS-DELEGATE flag is set in the Kerberos service ticket, which is a matter of realm policy.

always

Unconditionally allow the server to delegate.

--digest

(**HTTP**) Enables HTTP Digest authentication. This is a authentication that prevents the password from being sent as clear text. Use this in combination with the normal **-u**/**--user** option to set user name and password. See also **--ntlm**, **--negotiate** and **--anyauth** for related options.

If this option is used several times, the following occurrences make no difference.

--disable-eprt

(**FTP**) Tell **curl** to disable the use of the EPRT and LPRT commands when doing active FTP transfers. Curl will normally always first attempt to use EPRT, then LPRT before using PORT, but with this option, it will use PORT right away. EPRT and LPRT are extensions to the original FTP protocol, may not work on all servers but enable more functionality in a better way than the traditional PORT command.

--eprt can be used to explicitly enable EPRT again and **--no-eprt** is an alias for **--disable-eprt**.

Disabling EPRT only changes the active behavior. If you want to switch to passive mode you need to not use **-P**, **--ftp-port** or force it with **--ftp-pasv**.

--disable-epsv	(**FTP**) Tell **curl** to disable the use of the EPSV command when doing passive FTP transfers. Curl will normally always first attempt to use EPSV before PASV, but with this option, it will not try using EPSV. **--epsv** can be used to explicitly enable EPSV again and **--no-epsv** is an alias for **--disable-epsv**. Disabling EPSV only changes the passive behavior. If you want to switch to active mode you need to use **-P**, **--ftp-port**.
-D, **--dump-header** *<file>*	Write the protocol headers to the specified file. This option is handy to use when you want to store the headers that a HTTP site sends to you. Cookies from the headers could then be read in a second **curl** invoke by using the **-b**/**--cookie** option. The **-c**/**--cookie-jar** option is however a better way to store cookies. When used on FTP, the ftp server response lines are considered being "headers" and thus are saved there. If this option is used several times, the last one will be used.
-e, **--referer** *<URL>*	(**HTTP**) Sends the "Referer Page" information to the HTTP server. This can also be set with the **-H**/**--header**. When used with **-L**/**--location** you can append "**;auto**" to the **--referer** URL to make **curl** automatically set the previous URL when it follows a **Location:** header. The "**;auto**" string can be used alone, even if you don't set an initial **--referer**. If this option is used several times, the last one will be used.
--engine *<name>*	Select the OpenSSL crypto engine to use for cipher operations. Use **--engine list** to print a list of build-time supported engines. Note that not all (or none) of the engines may be available at run-time.
--environment	(**RISC OS ONLY**) Sets a range of environment variables, using the names the **-w** option supports, to easier allow extraction of useful information after having run **curl**.
--egd-file *<file>*	(**HTTPS**) Specify the path name to the Entropy Gathering Daemon socket. The socket is used to seed the random engine for SSL connections. See also the **--random-file** option.

-E, **--cert** <*certificate*[**:***password*]>	(**SSL**) Tells **curl** to use the specified client certificate file when getting a file with HTTPS, FTPS or another SSL-based protocol. The certificate must be in PEM format. If the optional password isn't specified, it will be queried for on the terminal. Note that this option assumes a "certificate" file that is the private key and the private certificate concatenated. See **--cert** and **--key** to specify them independently. If **curl** is built against the NSS SSL library then this option can tell **curl** the nickname of the certificate to use within the NSS database defined by the environment variable **SSL_DIR** (or by default **/etc/pki/nssdb**). If the NSS PEM PKCS#11 module (libnsspem.so) is available then PEM files may be loaded. If you want to use a file from the current directory, please precede it with "**./**" prefix, in order to avoid confusion with a nickname. If the nickname contains "**:**", it needs to be preceded by "\" so that it is not recognized as password delimiter. If the nickname contains "\", it needs to be escaped as "\\" so that it is not recognized as an escape character. (**iOS and Mac OS X only**) If **curl** is built against Secure Transport, then the certificate string must match the name of a certificate that's in the system or user keychain. The private key corresponding to the certificate, and certificate chain (if any), must also be present in the keychain. If this option is used several times, the last one will be used.
--cert-type <*type*>	(**SSL**) Tells **curl** what certificate type the provided certificate is in. **PEM**, **DER** and **ENG** are recognized *types*. If not specified, PEM is assumed. If this option is used several times, the last one will be used.
--cacert <*CA certificate*>	(**SSL**) Tells **curl** to use the specified certificate file to verify the peer. The file may contain multiple CA certificates. The certificate(s) must be in PEM format. Normally curl is built to use a default file for this, so this option is typically used to alter that default file. **curl** recognizes the environment variable named '**CURL_CA_BUNDLE**' if that is set, and uses the given path as a path to a CA cert bundle. This option overrides that variable. The Windows version of **curl** will automatically look for a CA certs file named '**curl-ca-bundle.crt**', either in the same directory as **curl.exe**, or in the Current Working Directory, or in any folder along your PATH. If **curl** is built against the NSS SSL library, the NSS PEM PKCS#11 module (libnsspem.so) needs to be available for this option to work properly. If this option is used several times, the last one will be used.
--capath <*CA certificate directory*>	(**SSL**) Tells **curl** to use the specified certificate directory to verify the peer. The certificates must be in PEM format, and the directory must have been processed using the **c_rehash** utility supplied with openssl. Using **--capath** can allow **curl** to make

https connections much more efficiently than using **--cacert** if the **--cacert** file contains many CA certificates.

If this option is used several times, the last one will be used.

-f, **--fail**

(**HTTP**) Fail silently (no output at all) on server errors. This is mostly done to better enable scripts, etc. to better deal with failed attempts. In normal cases when a HTTP server fails to deliver a document, it returns an HTML document stating so (which often also describes why). This flag will prevent **curl** from outputting that and return error 22.

This method is not fail-safe and there are occasions where non-successful response codes will slip through, especially when authentication is involved (response codes 401 and 407).

--ftp-account [*data*]

(**FTP**) When an FTP server asks for "account data" after user name and password has been provided, this data is sent off using the **ACCT** command. (Added in 7.13.0)

If this option is used twice, the second will override the previous use.

--ftp-create-dirs

(**FTP/SFTP**) When an FTP URL/operation uses a path that doesn't currently exist on the server, the standard behavior of curl is to fail. Using this option, **curl** will instead attempt to create missing directories.

--ftp-method [*method*]

(**FTP**) Control what method curl should use to reach a file on an FTP(S) server. The method argument should be one of the following alternatives:

multicwd

curl does a single **CWD** operation for each path part in the given URL. For deep hierarchies this means a lot of commands. This is the default but the slowest behavior.

nocwd

curl does no **CWD** at all. curl will do **SIZE**, **RETR**, **STOR**, etc. and give a full path to the server for all these commands. This is the fastest behavior.

singlecwd

curl does one CWD with the full target directory and then operates on the file "normally" (like in the multicwd case). This is somewhat more standards-compliant than '**nocwd**' but without the full penalty of '**multicwd**'.

--ftp-pasv

(**FTP**) Use PASV when transferring. PASV is the internal default behavior, but using this option can be used to override a previous **--ftp-port** option. (Added in 7.11.0)

	If this option is used several times, the following occurrences make no difference. Undoing an enforced passive really isn't doable but you must then instead enforce the correct **-P**, **--ftp-port** again. Passive mode means that curl will try the EPSV command first and then PASV, unless **--disable-epsv** is used.
--ftp-alternative-to-user*<command>*	(**FTP**) If authenticating with the **USER** and **PASS** commands fail, send this command. When connecting to Tumbleweed's Secure Transport server over FTPS using a client certificate, using "**SITE AUTH**" will tell the server to retrieve the username from the certificate. (Added in 7.15.5)
--ftp-skip-pasv-ip	(**FTP**) Tell **curl** to not use the IP address the server suggests in its response to **curl**'s PASV command when **curl** connects the data connection. Instead **curl** will re-use the same IP address it already uses for the control connection. (Added in 7.14.2) This option has no effect if PORT, EPRT or EPSV is used instead of PASV.
--ftp-pret	(**FTP**) Tell **curl** to send a PRET command before PASV (and EPSV). Certain FTP servers, mainly drftpd, require this non-standard command for directory listings as well as up and downloads in PASV mode. (Added in 7.20.x)
--ftp-ssl	(**FTP**) Try to use SSL/TLS for the FTP connection. Reverts to a non-secure connection if the server doesn't support SSL/TLS. (Added in 7.11.0) If this option is used twice, the second will again disable this.
--ftp-ssl-ccc	(**FTP**) Use CCC (Clear Command Channel) Shuts down the SSL/TLS layer after authenticating. The rest of the control channel communication will be unencrypted. This allows NAT routers to follow the FTP transaction. The default mode is passive. See **--ftp-ssl-ccc-mode** for other modes. (Added in 7.16.1)
--ftp-ssl-ccc-mode [*active/passive*]	(**FTP**) Use CCC (Clear Command Channel) Sets the CCC mode. The passive mode will not initiate the shutdown, but instead wait for the server to do it, and will not reply to the shutdown from the server. The active mode initiates the shutdown and waits for a reply from the server. (Added in 7.16.2)
--ftp-ssl-control	(**FTP**) Require SSL/TLS for the FTP login, clear for transfer. Allows secure authentication, but non-encrypted data transfers for efficiency. Fails the transfer if the server doesn't support SSL/TLS. (Added in 7.16.0)
--ftp-ssl-reqd	(**FTP**) Require SSL/TLS for the FTP connection. Terminates the connection if the server doesn't support SSL/TLS. (Added in 7.15.5)

If this option is used twice, the second will again disable this.

-F, **--form** <*name***=***content*>	(**HTTP**) This lets **curl** emulate a filled-in form in which a user has pressed the submit button. This causes **curl** to **POST** data using the Content-Type **multipart/form-data** according to RFC1867. This enables uploading of binary files etc. To force the 'content' part to be a file, prefix the file name with an "**@**" character. To just get the content part of a file, prefix the file name with the letter "**<**". The difference between "**@**" and "**<**" is that **@** makes a file get attached in the post as a file upload, while the **<** makes a text field and just get the contents for that text field from a file. For example, to send your password file to the server, where 'password' is the name of the form-field to which /etc/passwd will be the input: **curl -F password=@/etc/passwd www.mypasswords.com** To read the file's content from stdin instead of a file, use "**-**" where the file name should've been. This goes for both **@** and **<** constructs. You can also tell curl what Content-Type to use by using '**Type=**', in a manner similar to: **curl -F "web=@index.html;Type=text/html" url.com** or **curl -F "name=daniel;Type=text/foo" url.com** You can also explicitly change the name field of an file upload part by setting filename=, like this: **curl -F "file=@localfile;filename=nameinpost" url.com** If filename/path contains '**,**' or '**;**', it must be quoted by double-quotes like: **curl -F "file=@\"localfile\";filename=\"nameinpost\"" url.com** or **curl -F 'file=@"localfile";filename="nameinpost"' url.com** Note that if a filename/path is quoted by double-quotes, any double-quote or backslash within the filename must be escaped by backslash. This option can be used multiple times.
--form-string <*name=string*>	(**HTTP**) Similar to **--form** except that the value string for the named parameter is used literally. Leading '**@**' and '**<**' characters, and the '**;Type=**' string in the value have no special meaning. Use this in preference to **--form** if there's any possibility that the string value may accidentally trigger the '**@**' or '**<**' features of **--form**.

-g, **--globoff**

This option switches off the "URL globbing parser". When you set this option, you can specify URLs that contain the letters **{}[]** without having them being interpreted by **curl** itself. Note that these letters are not normal legal URL contents but they should be encoded according to the URI standard.

-G, **--get**

When used, this option will make all data specified with **-d**/**--data** or **--data-binary** to be used in a HTTP GET request instead of the POST request that otherwise would be used. The data will be appended to the URL with a '**?**' separator.

If used in combination with **-I**, the POST data will instead be appended to the URL with a HEAD request.

If this option is used several times, only the first one is used. This is because undoing a GET doesn't make sense, but you should then instead enforce the alternative method you prefer.

-H, **--header** *<header>*

(**HTTP**) Extra header to use when getting a web page. You may specify any number of extra headers. Note that if you should add a custom header that has the same name as one of the internal ones curl would use, your externally set header will be used instead of the internal one. This allows you to make even trickier stuff than **curl** would normally do. You should not replace internally set headers without knowing perfectly well what you're doing. Remove an internal header by giving a replacement without content on the right side of the colon, as in: **-H "Host:"**. If you send the custom header with no-value then its header must be terminated with a semicolon, such as **-H "X-Custom-Header;"** to send "**X-Custom-Header:**".

curl will make sure that each header you add/replace get sent with the proper end of line marker, you should thus not add that as a part of the header content: do not add newlines or carriage returns they will only mess things up for you.

See also the **-A**/**--user-agent** and **-e**/**--referer** options.

This option can be used multiple times to add/replace/remove multiple headers.

--hostpubmd5 *<md5>*

(**SCP/SFTP**) Pass a string containing 32 hexadecimal digits. The string should be the 128 bit MD5 checksum of the remote host's public key, curl will refuse the connection with the host unless the md5sums match. (Added in 7.17.1)

--ignore-content-length

(**HTTP**) Ignore the Content-Length header. This is particularly useful for servers running Apache 1.x, which will report incorrect **Content-Length** for files larger than 2 gigabytes.

-i, **--include**

(**HTTP**) Include the HTTP-header in the output. The HTTP-header includes things like server-name, date of the document, HTTP-version and more.

--interface *<name>*	Perform an operation using a specified interface. You can enter interface name, IP address or host name. An example could look like: **curl --interface eth0:1 http://www.netscape.com/** If this option is used several times, the last one will be used.
-I, **--head**	(**HTTP/FTP/FILE**) Fetch the HTTP-header only. HTTP-servers feature the command HEAD which this uses to get nothing but the header of a document. When used on an FTP or FILE file, **curl** displays the file size and last modification time only.
-j, **--junk-session-cookies**	(**HTTP**) When **curl** is told to read cookies from a given file, this option will make it discard all "session cookies". This will basically have the same effect as if a new session is started. Typical browsers always discard session cookies when they're closed down.
-J, **--remote-header-name**	(**HTTP**) This option tells the **-O**, **--remote-name** option to use the server-specified Content-Disposition filename instead of extracting a filename from the URL.
-k, **--insecure**	(**SSL**) This option explicitly allows **curl** to perform "insecure" SSL connections and transfers. All SSL connections are attempted to be made secure by using the CA certificate bundle installed by default. All connections considered "insecure" will fail unless **-k**/**--insecure** is used. See this online resource for more information: http://curl.haxx.se/docs/sslcerts.html.
--key *<key>*	(**SSL/SSH**) Private key file name. Allows you to provide your private key in this separate file. If this option is used several times, the last one will be used.
--key-type *<type>*	(**SSL**) Private key file type. Specify which type your **--key** provided private key is. DER, PEM, and ENG are supported. If not specified, PEM is assumed. If this option is used several times, the last one will be used.
--krb *<level>*	(**FTP**) Enable Kerberos authentication and use. The level must be entered and should be one of 'clear', 'safe', 'confidential', or 'private'. Should you use a level that is not one of these, 'private' will instead be used. This option requires a library built with kerberos4 or GSSAPI (GSS-Negotiate) support. This is not very common. Use **-V**, **--version** to see if your **curl** supports it. If this option is used several times, the last one will be used.

-K, **--config** *<config file>*

Specify which config file to read **curl** arguments from. The config file is a text file in which command line arguments can be written which then will be used as if they were written on the actual command line. Options and their parameters must be specified on the same config file line. If the parameter is to contain white spaces, the parameter must be enclosed within quotes. If the first column of a config line is a '**#**' character, the rest of the line will be treated as a comment. Only write one option per physical line in the config file.

Specify the filename as '**-**' to make curl read the file from stdin.

Note that to be able to specify a URL in the config file, you need to specify it using the **--url** option, and not by writing the URL on its own line. So, it could look similar to this:

url = "http://curl.haxx.se/docs/"

Long option names can optionally be given in the config file without the initial double dashes.

When **curl** is invoked, it always (unless **-q** is used) checks for a default config file and uses it if found. The default config file is checked for in the following places in this order:

1) **curl** tries to find the "home dir": It first checks for the **CURL_HOME** and then the **HOME** environment variables. Failing that, it uses **getpwuid()** on unix-like systems (which returns the home dir given the current user in your system). On Windows, it then checks for the **APPDATA** variable, or as a last resort the '**%USER-PROFILE%\Application Data**'.

2) On windows, if there is no **_curlrc** file in the home dir, it checks for one in the same dir the executable **curl** is placed. On unix-like systems, it will try to load **.curlrc** from the determined home dir.

This option can be used multiple times to load multiple config files.

--keepalive-time *<seconds>*

This option sets the time a connection needs to remain idle before sending keepalive probes and the time between individual keepalive probes. It is currently effective on operating systems offering the TCP_KEEPIDLE and TCP_KEEPINTVL socket options (meaning Linux, recent AIX, HP-UX, and more). This option has no effect if **--no-keepalive** is used. (Added in 7.18.0)

If this option is used several times, the last one will be used. If unspecified, the option defaults to 60 seconds.

--limit-rate *<speed>*

Specify the maximum transfer rate you want **curl** to use. This feature is useful if you have a limited pipe and you'd like your transfer not use your entire bandwidth.

The given speed is measured in bytes/second, unless a suffix is appended. Appending '**k**' or '**K**' will count the number as kilobytes, '**m**' or '**M**' makes it megabytes while '**g**' or '**G**' makes it gigabytes. Examples: **200K**, **3m** and **1G**.

	The given rate is the average speed counted during the entire transfer. It means that **curl** might use higher transfer speeds in short bursts, but over time it uses no more than the given rate. If you are also using the **-Y**/**--speed-limit** option, that option will take precedence and might cripple the rate-limiting slightly, to help keep the speed-limit logic working. If this option is used several times, the last one will be used.
-l/**--list-only**	(**FTP**) When listing an FTP directory, this switch forces a name-only view. Especially useful if you want to machine-parse the contents of an FTP directory since the normal directory view doesn't use a standard look or format. This option causes an FTP NLST command to be sent. Some FTP servers list only files in their response to NLST; they do not include subdirectories and symbolic links.
--local-port <*num*>[**-***num*]	Set a preferred number or range of local port numbers to use for the connection(s). Note that port numbers by nature are a scarce resource that will be busy at times so setting this range to something too narrow might cause unnecessary connection setup failures. (Added in 7.15.2)
-L, **--location**	(**HTTP/HTTPS**) If the server reports that the requested page has moved to a different location (indicated with a **Location:** header and a 3XX response code) this option will make **curl** redo the request on the new place. If used together with **-i**/**--include** or **-I**/**--head**, headers from all requested pages will be shown. When authentication is used, **curl** only sends its credentials to the initial host. If a redirect takes **curl** to a different host, it won't be able to intercept the user+password. See also **--location-trusted** on how to change this. You can limit the amount of redirects to follow by using the **--max-redirs** option. When **curl** follows a redirect and the request is not a plain GET (for example POST or PUT), it will do the following request with a GET if the HTTP response was 301, 302, or 303. If the response code was any other 3xx code, **curl** will re-send the following request using the same unmodified method.
--libcurl <*file*>	Append this option to any ordinary **curl** command line, and you will get a libcurl-using C source code written to the file that does the equivalent of what your command-line operation does! It should be noted that this option is extremely awesome. If this option is used several times, the last given file name will be used. (Added in 7.16.1)
--location-trusted	(**HTTP/HTTPS**) Similar to **-L**/**--location**, but will allow sending the name + password to all hosts that the site may redirect to. This may or may not introduce a security breach if the site redirects you do a site to which you'll send your authentication info (which is plaintext in the case of HTTP Basic authentication).

--max-filesize *<bytes>*

Specify the maximum size (in bytes) of a file to download. If the file requested is larger than this value, the transfer will not start and **curl** will return with exit code 63.

NOTE: The file size is not always known prior to download, and for such files this option has no effect even if the file transfer ends up being larger than this given limit. This concerns both FTP and HTTP transfers.

-m, **--max-time** *<seconds>*

Maximum time in seconds that you allow the whole operation to take. This is useful for preventing your batch jobs from hanging for hours due to slow networks or links going down. See also the **--connect-timeout** option.

If this option is used several times, the last one will be used.

--mail-auth *<address>*

(**SMTP**) Specify a single address. This will be used to specify the authentication address (identity) of a submitted message that is being relayed to another server.

(Added in 7.25.0)

--mail-from *<address>*

(**SMTP**) Specify a single address that the given mail should get sent from.

(Added in 7.20.0)

--mail-rcpt *<address>*

(**SMTP**) Specify a single address that the given mail should get sent to. This option can be used multiple times to specify many recipients.

(Added in 7.20.0)

--metalink

This option can tell **curl** to parse and process a given URI as Metalink file (both version 3 and 4 (RFC 5854) are supported) and make use of the mirrors listed within for failover if there are errors (such as the file or server not being available). It will also verify the hash of the file after the download completes. The Metalink file itself is downloaded and processed in memory and not stored in the local file system.

Example to use a remote Metalink file:

curl --metalink http://www.example.com/example.metalink

To use a Metalink file in the local file system, use FILE protocol (file://):

curl --metalink file://example.metalink

Please note that if FILE protocol is disabled, there is no way to use a local Metalink file at the time of this writing. Also note that if **--metalink** and **--include** are used together, **--include** will be ignored. This is because including headers in the response will break Metalink parser and if the headers are included in the file described in Metalink file, hash check will fail.

(Added in 7.27.0, if built against the libmetalink library.)

-n, **--netrc**

Makes curl scan the **.netrc** file in the user's home directory for login name and password. This is typically used for **ftp** on unix. If used with http, **curl** will enable user authentication. See **netrc(4)** or ftp documentation for details on the file format. **Curl** will not complain if that file hasn't the right permissions (it should not be world nor group readable). The environment variable "**HOME**" is used to find the home directory.

A quick and very simple example of how to setup a **.netrc** to allow **curl** to ftp to the machine host.domain.com with user name 'myself' and password 'secret' should look similar to:

machine host.domain.com login myself password secret

If this option is used twice, the second will again disable netrc usage.

--negotiate

(**HTTP**) Enables GSS-Negotiate authentication. The GSS-Negotiate method was designed by Microsoft and is used in their web applications. It is primarily meant as a support for Kerberos5 authentication but may be also used along with another authentication methods.

This option requires that the curl library was built with GSSAPI support. This is not very common. Use **-V**/**--version** to see if your version supports GSS-Negotiate.

When using this option, you must also provide a fake **-u**/**--user** option to activate the authentication code properly. Sending a '**-u :**' is enough as the user name and password from the **-u** option aren't actually used.

If this option is used several times, the following occurrences make no difference.

--no-keepalive

Disables the use of keepalive messages on the TCP connection, as by default **curl** enables them.

Note that this is the negated option name documented. You can thus use **--keepalive** to enforce keepalive.

--no-sessionid

(**SSL**) Disable **curl**'s use of SSL session-ID caching. By default all transfers are done using the cache. Note that while nothing should ever get hurt by attempting to reuse SSL session-IDs, there seem to be broken SSL implementations in the wild that may require you to disable this in order for you to succeed. (Added in 7.16.0)

Note that this is the negated option name documented. You can thus use **--sessionid** to enforce session-ID caching.

--noproxy *<no-proxy-list>*

Comma-separated list of hosts which do not use a proxy, if one is specified. The only wildcard is a single ***** character, which matches all hosts, and effectively disables the proxy. Each name in this list is matched as either a domain which contains the hostname, or the hostname itself. For example, **local.com** would match **local.com**, **local.com:80**, and **www.local.com**, but not **www.notlocal.com**. (Added in 7.19.4).

-N, **--no-buffer**

Disables the buffering of the output stream. In normal work situations, **curl** will use a standard buffered output stream that will have the effect that it will output the data in chunks, not necessarily exactly when the data arrives. Using this option will disable that buffering.

Note that this is the negated option name documented. You can thus use **--buffer** to enforce the buffering.

--netrc-file

This option is similar to **--netrc**, except that you provide the path (absolute or relative) to the netrc file that **curl** should use. You can only specify one netrc file per invocation. If several **--netrc-file** options are provided, only the last one will be used. (Added in 7.21.5)

This option overrides any use of **--netrc** as they are mutually exclusive. It will also abide by **--netrc-optional** if specified.

--netrc-optional

Very similar to **--netrc**, but this option makes the .netrc usage optional and not mandatory as the **--netrc** option does.

--ntlm

(**HTTP**) Enables NTLM authentication. The NTLM authentication method was designed by Microsoft and is used by IIS web servers. It is a proprietary protocol, reverse-engineered by clever people and implemented in **curl** based on their efforts. This kind of behavior should not be endorsed, you should encourage everyone who uses NTLM to switch to a public and documented authentication method instead, such as Digest.

If you want to enable NTLM for your proxy authentication, then use **--proxy-ntlm**.

This option requires a library built with SSL support. Use **-V**, **--version** to see if your **curl** supports NTLM.

If this option is used several times, only the first one is used.

-o, **--output** *<file>*

Write output to *<file>* instead of stdout. If you are using **{}** or **[]** to fetch multiple documents, you can use '**#**' followed by a number in the *<file>* specifier. That variable will be replaced with the current string for the URL being fetched. Like in:

```
curl http://{one,two}.site.com -o "file_#1.txt"
```

or use several variables like:

```
curl http://{site,host}.host[1-5].com -o "#1_#2"
```

You may use this option as many times as you have number of URLs.

See also the **--create-dirs** option to create the local directories dynamically. Specifying the output as '**-**' (a single dash) will force the output to be done to stdout.

-O, **--remote-name**

Write output to a local file named like the remote file we get. (Only the file part of the remote file is used, the path is cut off.)

The remote file name to use for saving is extracted from the given URL, nothing else.

Consequentially, the file will be saved in the current working directory. If you want the file saved in a different directory, make sure you change current working directory before you invoke **curl** with the **-O**, **--remote-name** flag!

You may use this option as many times as you have number of URLs.

--pass *<phrase>*

(**SSL/SSH**) Pass phrase for the private key.

If this option is used several times, the last one will be used.

--post301

(**HTTP**) Tells **curl** to respect RFC 2616/10.3.2 and not convert POST requests into GET requests when following a 301 redirection. The non-RFC behaviour is ubiquitous in web browsers, so **curl** does the conversion by default to maintain consistency. However, a server may require a POST to remain a POST after such a redirection. This option is meaningful only when using **-L**, **--location** (Added in 7.17.1)

--post302

(**HTTP**) Tells **curl** to respect RFC 2616/10.3.2 and not convert POST requests into GET requests when following a 302 redirection. The non-RFC behaviour is ubiquitous in web browsers, so **curl** does the conversion by default to maintain consistency. However, a server may require a POST to remain a POST after such a redirection. This option is meaningful only when using **-L**, **--location** (Added in 7.19.1)

--post303

(**HTTP**) Tells **curl** to respect RFC 2616/10.3.2 and not convert POST requests into GET requests when following a 303 redirection. The non-RFC behaviour is ubiquitous in web browsers, so **curl** does the conversion by default to maintain consistency. However, a server may require a POST to remain a POST after such a redirection. This option is meaningful only when using **-L**, **--location** (Added in 7.26.0)

--proto *<protocols>*

Tells **curl** to use the listed protocols for its initial retrieval. Protocols are evaluated left to right, are comma separated, and are each a protocol name or '**all**', optionally prefixed by zero or more modifiers. Available modifiers are:

+

Permit this protocol in addition to protocols already permitted (this is the default if no modifier is used).

-

Deny this protocol, removing it from the list of protocols already permitted.

=

Permit only this protocol (ignoring the list already permitted), though subject to later modification by subsequent entries in the comma separated list.

For example:

--proto -ftps uses the default protocols, but disables ftps

--proto -all,https,+http only enables http and https

--proto =http,https also only enables http and https

Unknown protocols produce a warning. This allows scripts to safely rely on being able to disable potentially dangerous protocols, without relying upon support for that protocol being built into **curl** to avoid an error.

This option can be used multiple times, in which case the effect is the same as concatenating the protocols into one instance of the option. (Added in 7.20.2)

--proto-redir <*protocols*>	Tells **curl** to use the listed protocols after a redirect. See **--proto** for how protocols are represented. (Added in 7.20.2)
--proxy-anyauth	Tells **curl** to pick a suitable authentication method when communicating with the given proxy. This will cause an extra request/response round-trip. (Added in 7.13.2)
--proxy-basic	Tells **curl** to use HTTP Basic authentication when communicating with the given proxy. Use **--basic** for enabling HTTP Basic with a remote host. Basic is the default authentication method **curl** uses with proxies.
--proxy-digest	Tells **curl** to use HTTP Digest authentication when communicating with the given proxy. Use **--digest** for enabling HTTP Digest with a remote host.
--proxy-ntlm	Tells **curl** to use HTTP NTLM authentication when communicating with the given proxy. Use **--ntlm** for enabling NTLM with a remote host.
--proxy1.0 <*proxyhost*[**:***port*]>	Use the specified HTTP 1.0 proxy. If the port number is not specified, it is assumed at port 1080.

	The only difference between this and the HTTP proxy option (**-x**, **--proxy**), is that attempts to use CONNECT through the proxy will specify an HTTP 1.0 protocol instead of the default HTTP 1.1.
--pubkey *<key>*	(**SSH**) Public key file name. Allows you to provide your public key in this separate file. If this option is used several times, the last one will be used.
-p, **--proxytunnel**	When an HTTP proxy is used (**-x**, **--proxy**), this option will cause non-HTTP protocols to attempt to tunnel through the proxy instead of merely using it to do HTTP-like operations. The tunnel approach is made with the HTTP proxy CONNECT request and requires that the proxy allows direct connect to the remote port number **curl** wants to tunnel through to.
-P, **--ftp-port** *<address>*	(**FTP**) Reverses the default initiator/listener roles when connecting with FTP. This switch makes **curl** use active mode. In practice, **curl** then tells the server to connect back to the client's specified address and port, while passive mode asks the server to setup an IP address and port for it to connect to. *<address>* should be one of: **interface** i.e "**eth0**" to specify which interface's IP address you want to use (Unix only) **IP address** i.e "**192.168.10.1**" to specify the exact IP address **host name** i.e "**my.host.domain**" to specify the machine **-** make **curl** pick the same IP address that is already used for the control connection. If this option is used several times, the last one will be used. Disable the use of PORT with **--ftp-pasv**. Disable the attempt to use the EPRT command instead of PORT by using **--disable-eprt**. EPRT is really PORT++. Starting in 7.19.5, you can append "**:[start]-[end]**" to the right of the address, to tell curl what TCP port range to use. That means you specify a port range, from a lower to a higher number. A single number works as well, but do note that it increases the risk of failure since the port may not be available.
-q	If used as the first parameter on the command line, the **curlrc** config file will not be read and used. See the **-K**, **--config** for details on the default config file search path.

-Q, **--quote** *<command>*

(FTP/SFTP) Send an arbitrary command to the remote FTP or SFTP server. Quote commands are sent BEFORE the transfer is taking place (just after the initial PWD command to be exact). To make commands take place after a successful transfer, prefix them with a dash '**-**'. To make commands get sent after libcurl has changed working directory, just before the transfer command(s), prefix the command with '**+**' (this is only supported for FTP). You may specify any amount of commands. If the server returns failure for one of the commands, the entire operation will be aborted. You must send syntactically correct FTP commands as RFC959 defines to FTP servers, or one of the commands listed below to SFTP servers.

This option can be used multiple times. When speaking to an FTP server, prefix the command with an asterisk (*****) to make **curl** continue even if the command fails as by default **curl** will stop at first failure.

SFTP is a binary protocol. Unlike for FTP, **curl** interprets SFTP quote commands itself before sending them to the server. File names may be quoted shell-style to embed spaces or special characters. Following is the list of all supported SFTP quote commands:

chgrp group file

The **chgrp** command sets the group ID of the file named by the file operand to the group ID specified by the group operand. The group operand is a decimal integer group ID.

chmod mode file

The **chmod** command modifies the file mode bits of the specified file. The mode operand is an octal integer mode number.

chown user file

The **chown** command sets the owner of the file named by the file operand to the user ID specified by the user operand. The user operand is a decimal integer user ID.

ln source_file target_file

The **ln** and **symlink** commands create a symbolic link at the target_file location pointing to the source_file location.

mkdir directory_name

The **mkdir** command creates the directory named by the directory_name operand.

pwd

The **pwd** command returns the absolute pathname of the current working directory.

rename source target

The **rename** command renames the file or directory named by the source operand to the destination path named by the target operand.

rm file

The **rm** command removes the file specified by the file operand.

rmdir directory

The **rmdir** command removes the directory entry specified by the directory operand, provided it is empty.

symlink source_file target_file

See **ln**.

--random-file *<file>*

(**SSL**) Specify the path name to file containing what will be considered as random data. The data is used to seed the random engine for SSL connections. See also the **--egd-file** option.

--raw

(**HTTP**) When used, it disables all internal HTTP decoding of content or transfer encodings and instead makes them passed on unaltered, raw. (Added in 7.16.2)

--remote-name-all

This option changes the default action for all given URLs to be dealt with as if **-O**, **--remote-name** were used for each one. So if you want to disable that for a specific URL after **--remote-name-all** has been used, you must use "**-o -**" or **--no-remote-name**. (Added in 7.19.0)

--resolve *<host***:***port***:***address>*

Provide a custom address for a specific host and port pair. Using this, you can make the **curl** requests(s) use a specified address and prevent the otherwise normally resolved address to be used. Consider it a sort of **/etc/hosts** alternative provided on the command line. The port number should be the number used for the specific protocol the host will be used for. It means you need several entries if you want to provide address for the same host but different ports.

This option can be used many times to add many host names to resolve.

(Added in 7.21.3)

-r, **--range** *<range>*

(**HTTP/FTP**) Retrieve a byte range (i.e a partial document) from a HTTP/1.1 or FTP server. Ranges can be specified in a number of ways.

0-499 specifies the first 500 bytes;

500-999 specifies the second 500 bytes;

-500 specifies the last 500 bytes;

9500- specifies the bytes from offset 9500 and forward;

0-0,-1 specifies the first and last byte only(*)(H);

500-700,600-799 specifies 300 bytes from offset 500(H);

100-199,500-599 specifies two separate 100 bytes ranges(*)(H).

(*) = **NOTE** that this will cause the server to reply with a multipart response!

You should also be aware that many HTTP/1.1 servers do not have this feature enabled, so that when you attempt to get a range, you'll instead get the whole document.

FTP range downloads only support the simple syntax '**start-stop**' (optionally with one of the numbers omitted). It depends on the non-RFC command SIZE.

Only digit characters (0-9) are valid in the '**start**' and '**stop**' fields of the '**start-stop**' range syntax. If a non-digit character is given in the range, the server's response will be unspecified, depending on the server's configuration.

If this option is used several times, the last one will be used.

-R, **--remote-time**	When used, this will make libcurl attempt to figure out the timestamp of the remote file, and if that is available make the local file get that same timestamp.
--retry *<num>*	If a transient error is returned when **curl** tries to perform a transfer, it will retry this number of times before giving up. Setting the number to **0** makes **curl** do no retries (which is the default). Transient error means either: a timeout, an FTP 5xx response code or an HTTP 5xx response code. When **curl** is about to retry a transfer, it will first wait one second and then for all forthcoming retries it will double the waiting time until it reaches 10 minutes which then will be the delay between the rest of the retries. By using **--retry-delay** you disable this exponential backoff algorithm. See also **--retry-max-time** to limit the total time allowed for retries. (Added in 7.12.3) If this option is used multiple times, the last occurrence decide the amount.
--retry-delay *<seconds>*	Make **curl** sleep this amount of time between each retry when a transfer has failed with a transient error (it changes the default backoff time algorithm between retries). This option is only interesting if **--retry** is also used. Setting this delay to zero will make curl use the default backoff time. (Added in 7.12.3) If this option is used multiple times, the last occurrence decide the amount.

--retry-max-time *<seconds>*	The retry timer is reset before the first transfer attempt. Retries will be done as usual (see **--retry**) as long as the timer hasn't reached this given limit. Notice that if the timer hasn't reached the limit, the request will be made and while performing, it may take longer than this given time period. To limit a single request's maximum time, use **-m**, **--max-time**. Set this option to zero to not timeout retries. (Added in 7.12.3) If this option is used multiple times, the last occurrence decide the amount.
-s, **--silent**	Silent mode. Don't show progress meter or error messages. Makes **curl** mute. It will still output the data you ask for, potentially even to the terminal/stdout unless you redirect it.
--sasl-ir	Enable initial response in SASL authentication. (Added in 7.31.0)
-S, **--show-error**	When used with **-s** it makes **curl** show error message if it fails.
--ssl	(**FTP, POP3, IMAP, SMTP**) Try to use SSL/TLS for the connection. Reverts to a non-secure connection if the server doesn't support SSL/TLS. See also **--ftp-ssl-control** and **--ssl-reqd** for different levels of encryption required. (Added in 7.20.0) This option was formerly known as **--ftp-ssl** (Added in 7.11.0). That option name can still be used but will be removed in a future version.
--ssl-reqd	(**FTP, POP3, IMAP, SMTP**) Require SSL/TLS for the connection. Terminates the connection if the server doesn't support SSL/TLS. (Added in 7.20.0) This option was formerly known as **--ftp-ssl-reqd** (added in 7.15.5). That option name can still be used but will be removed in a future version.
--ssl-allow-beast	(**SSL**) This option tells **curl** to not work around a security flaw in the SSL3 and TLS1.0 protocols known as BEAST. If this option isn't used, the SSL layer may use work-arounds known to cause interoperability problems with some older SSL implementations. **WARNING:** this option loosens the SSL security, and by using this flag you ask for exactly that. (Added in 7.25.0)
--socks4 *<host*[**:***port*]*>*	Use the specified SOCKS4 proxy. If the port number is not specified, it is assumed at port 1080. (Added in 7.15.2) This option overrides any previous use of **-x**, **--proxy**, as they are mutually exclusive. Since 7.21.7, this option is superfluous since you can specify a socks4 proxy with **-x**, **--proxy** using a **socks4://** protocol prefix. If this option is used several times, the last one will be used.

--socks4a <*host*[**:***port*]>

Use the specified SOCKS4a proxy. If the port number is not specified, it is assumed at port 1080. (Added in 7.18.0)

This option overrides any previous use of **-x**, **--proxy**, as they are mutually exclusive.

Since 7.21.7, this option is superfluous since you can specify a socks4a proxy with **-x**, **--proxy** using a **socks4a://** protocol prefix.

If this option is used several times, the last one will be used.

--socks5-hostname <*host*[**:***port*]>

Use the specified SOCKS5 proxy (and let the proxy resolve the host name). If the port number is not specified, it is assumed at port 1080. (Added in 7.18.0)

This option overrides any previous use of **-x**, **--proxy**, as they are mutually exclusive.

Since 7.21.7, this option is superfluous since you can specify a socks5 hostname proxy with **-x**, **--proxy** using a **socks5h://** protocol prefix.

If this option is used several times, the last one will be used. (This option was previously wrongly documented and used as **--socks** without the number appended.)

--socks5 <*host*[**:***port*]>

Use the specified SOCKS5 proxy - but resolve the host name locally. If the port number is not specified, it is assumed at port 1080.

This option overrides any previous use of **-x**, **--proxy**, as they are mutually exclusive.

Since 7.21.7, this option is superfluous since you can specify a socks5 proxy with **-x**, **--proxy** using a **socks5://** protocol prefix.

If this option is used several times, the last one will be used. (This option was previously wrongly documented and used as **--socks** without the number appended.)

This option (as well as **--socks4**) does not work with IPV6, FTPS or LDAP.

--socks5-gssapi-service<*servicename*>

The default service name for a socks server is rcmd/server-fqdn. This option allows you to change it.

Examples:

--socks5 proxy-name --socks5-gssapi-service sockd

would use sockd/proxy-name;

--socks5 proxy-name --socks5-gssapi-service sockd/real-name

would use sockd/real-name for cases where the proxy-name does not match the principal name.

--socks5-gssapi-nec	As part of the gssapi negotiation a protection mode is negotiated. RFC 1961 says in section 4.3/4.4 it should be protected, but the NEC reference implementation does not. The option **--socks5-gssapi-nec** allows the unprotected exchange of the protection mode negotiation. (Added in 7.19.4).
--stderr *<file>*	Redirect all writes to stderr to the specified file instead. If the file name is a plain '**-**', it is instead written to stdout. If this option is used several times, the last one will be used.
--tcp-nodelay	Turn on the **TCP_NODELAY** option. See the **curl_easy_setopt** man page for details about this option. (Added in 7.11.2)
--tftp-blksize *<value>*	(**TFTP**) Set TFTP BLKSIZE option (must be >512). This is the block size that curl will try to use when transferring data to or from a TFTP server. By default 512 bytes will be used. If this option is used several times, the last one will be used. (Added in 7.20.0)
--tlsauthtype *<authtype>*	Set TLS authentication type. Currently, the only supported option is "SRP", for TLS-SRP (RFC 5054). If **--tlsuser** and **--tlspassword** are specified but **--tlsauthtype** is not, then this option defaults to "SRP". (Added in 7.21.4)
--tlsuser *<user>*	Set username for use with the TLS authentication method specified with **--tlsauthtype**. Requires that **--tlspassword** also be set. (Added in 7.21.4)
--tlspassword *<password>*	Set password for use with the TLS authentication method specified with **--tlsauthtype**. Requires that **--tlsuser** also be set. (Added in 7.21.4)
--tr-encoding	(**HTTP**) Request a compressed Transfer-Encoding response using one of the algorithms **curl** supports, and uncompress the data while receiving it. (Added in 7.21.6)
-t, **--telnet-option** *<OPT=val>*	Pass options to the telnet protocol. Supported options are: **TType=<term>** Sets the terminal type. **XDISPLOC=<X display>** Sets the X display location.

NEW_ENV=<var,val> Sets an environment variable.

-T, **--upload-file** *<file>*

This transfers the specified local file to the remote URL. If there is no file part in the specified URL, Curl will append the local file name. **NOTE** that you must use a trailing **/** on the last directory to really prove to **curl** that there is no file name or **curl** will think that your last directory name is the remote file name to use. That will most likely cause the upload operation to fail. If this is used on a http(s) server, the PUT command will be used.

Use the file name "**-**" (a single dash) to use stdin instead of a given file. Alternately, the file name "**.**" (a single period) may be specified instead of "**-**" to use stdin in non-blocking mode to allow reading server output while stdin is being uploaded.

You can specify one **-T** for each URL on the command line. Each **-T** + **URL** pair specifies what to upload and to where. **curl** also supports "globbing" of the **-T** argument, meaning that you can upload multiple files to a single URL by using the same URL globbing style supported in the URL, like this:

```
curl -T "{file1,file2}" http://www.uploadtothissite.com
```

or even

```
curl -T "img[1-1000].png" ftp://ftp.picturemania.com/upload/
```

--trace *<file>*

Enables a full trace dump of all incoming and outgoing data, including descriptive information, to the given output file. Use "**-**" as filename to have the output sent to stdout.

This option overrides previous uses of **-v**, **--verbose** or **--trace-ascii**.

If this option is used several times, the last one will be used.

--trace-ascii *<file>*

Enables a full trace dump of all incoming and outgoing data, including descriptive information, to the given output file. Use "**-**" as filename to have the output sent to stdout.

This is very similar to **--trace**, but leaves out the hex part and only shows the ASCII part of the dump. It makes smaller output that might be easier to read for untrained humans.

This option overrides previous uses of **-v**, **--verbose** or **--trace**.

If this option is used several times, the last one will be used.

--trace-time

Prepends a time stamp to each trace or verbose line that **curl** displays. (Added in 7.14.0)

-u, **--user** *<user***:***password>*	Specify user and password to use for server authentication. Overrides **-n**, **--netrc** and **--netrc-optional**. If you just give the user name (without entering a colon) **curl** will prompt for a password. If you use an SSPI-enabled **curl** binary and do NTLM autentication, you can force **curl** to pick up the user name and password from your environment by specifying a single colon with this option: "**-u :**". If this option is used several times, the last one will be used.
-U, **--proxy-user** *<user***:***password>*	Specify user and password to use for proxy authentication. If you use an SSPI-enabled **curl** binary and do NTLM autentication, you can force **curl** to pick up the user name and password from your environment by specifying a single colon with this option: "**-U :**". If this option is used several times, the last one will be used.
--url *<URL>*	Specify a URL to fetch. This option is mostly handy when you want to specify URL(s) in a config file. This option may be used any number of times. To control where this URL is written, use the **-o**, **--output** or the **-O**, **--remote-name options**.
-v, **--verbose**	Makes the fetching more verbose/talkative. Mostly usable for debugging. Lines starting with '**>**' means "header data" sent by **curl**, '**<**' means "header data" received by **curl** that is hidden in normal cases and lines starting with '*****' means additional info provided by **curl**. Note that if you only want HTTP headers in the output, **-i**, **--include** might be option you're looking for. If you think this option still doesn't give you enough details, consider using **--trace** or **--trace-ascii** instead. This option overrides previous uses of **--trace-ascii** or **--trace**. Use **-s**, **--silent** to make **curl** quiet.
-w, **--write-out** *<format>*	Defines what to display on stdout after a completed and successful operation. The format is a string that may contain plain text mixed with any number of variables. The string can be specified as "**string**", to get read from a particular file you specify it "**@filename**" and to tell curl to read the format from stdin you write "**@-**".

The variables present in the output format will be substituted by the value or text that **curl** thinks fit, as described below. All variables are specified like **%{variable_name}** and to output a normal **%** you just write them like **%%**. You can output a newline by using **\n**, a carriage return with **\r** and a tab space with **\t**.

NOTE: The **%**-letter is a special letter in the win32-environment, where all occurrences of **%** must be doubled when using this option.

Available variables are at this point:

content_type

The Content-Type of the requested document, if there was any.

filename_effective

The ultimate filename that **curl** writes out to. This is only meaningful if **curl** is told to write to a file with the **--remote-name** or **--output** option. It's most useful in combination with the **--remote-header-name** option. (Added in 7.25.1)

ftp_entry_path

The initial path **curl** ended up in when logging on to the remote FTP server. (Added in 7.15.4)

http_code

The numerical response code that was found in the last retrieved HTTP(S) or FTP(s) transfer. In 7.18.2 the alias **response_code** was added to show the same info.

http_connect

The numerical code that was found in the last response (from a proxy) to a **curl** CONNECT request. (Added in 7.12.4)

local_ip

The IP address of the local end of the most recently done connection - can be either IPv4 or IPv6 (Added in 7.29.0)

local_port

The local port number of the most recently done connection (Added in 7.29.0)

num_connects

Number of new connects made in the recent transfer. (Added in 7.12.3)

num_redirects

Number of redirects that were followed in the request. (Added in 7.12.3)

redirect_url

When an HTTP request was made without **-L** to follow redirects, this variable shows the actual URL a redirect would take you to. (Added in 7.18.2)

remote_ip

The remote IP address of the most recently done connection - can be either IPv4 or IPv6 (Added in 7.29.0)

remote_port

The remote port number of the most recently done connection (Added in 7.29.0)

size_download

The total amount of bytes that were downloaded.

size_header

The total amount of bytes of the downloaded headers.

size_request

The total amount of bytes that were sent in the HTTP request.

size_upload

The total amount of bytes that were uploaded.

speed_download

The average download speed that **curl** measured for the complete download. Bytes per second.

speed_upload

The average upload speed that **curl** measured for the complete upload. Bytes per second.

ssl_verify_result

The result of the SSL peer certificate verification that was requested. 0 means the verification was successful. (Added in 7.19.0)

time_appconnect

The time, in seconds, it took from the start until the SSL/SSH/etc connect/handshake to the remote host was completed. (Added in 7.19.0)

time_connect

The time, in seconds, it took from the start until the TCP connect to the remote host (or proxy) was completed.

time_namelookup

The time, in seconds, it took from the start until the name resolving was completed.

time_pretransfer

The time, in seconds, it took from the start until the file transfer was just about to begin. This includes all pre-transfer commands and negotiations that are specific to the particular protocol(s) involved.

time_redirect

The time, in seconds, it took for all redirection steps include name lookup, connect, pretransfer, and transfer before the final transaction was started. **time_redirect** shows the complete execution time for multiple redirections. (Added in 7.12.3)

time_starttransfer

The time, in seconds, it took from the start until the first byte was just about to be transferred. This includes **time_pretransfer** and also the time the server needed to calculate the result.

time_total

The total time, in seconds, that the full operation lasted. The time will be displayed with millisecond resolution.

url_effective

The URL that was fetched last. This is most meaningful if you've told **curl** to follow **location:** headers.

-x, **--proxy** *<[protocol://][user:password@]proxyhost[:port]>*

Use the specified HTTP proxy. If the port number is not specified, it is assumed at port 1080.

This option overrides existing environment variables that set the proxy to use. If there's an environment variable setting a proxy, you can set proxy to **""** to override it.

All operations that are performed over an HTTP proxy will transparently be converted to HTTP. It means that certain protocol specific operations might not be available. This is not the case if you can tunnel through the proxy, as one with the **-p**, **--proxytunnel** option.

User and password that might be provided in the proxy string are URL decoded by **curl**. This allows you to pass in special characters such as **@** by using %40 or pass in a colon with %3a.

The proxy host can be specified the exact same way as the proxy environment variables, including the protocol prefix (http://) and the embedded user **+** password.

From 7.21.7, the proxy string may be specified with a **protocol://** prefix to specify alternative proxy protocols.
Use **socks4://**, **socks4a://**, **socks5://** or **socks5h://** to request the specific SOCKS version to be used. No protocol specified, **http://** and all others will be treated as HTTP proxies.

If this option is used several times, the last one will be used.

-X, **--request** *<command>*

(**HTTP**) Specifies a custom request method to use when communicating with the HTTP server. The specified request will be used instead of the method otherwise used (which defaults to GET). Read the HTTP 1.1 specification for details and explanations. Common additional HTTP requests include PUT and DELETE, but related technologies like WebDAV offers PROPFIND, COPY, MOVE, and more.

Normally you don't need this option. All sorts of GET, HEAD, POST, and PUT requests are rather invoked by using dedicated command line options.

This option only changes the actual word used in the HTTP request, it does not alter the way curl behaves. So for example if you want to make a proper HEAD request, using **-X** HEAD will not suffice. You need to use the **-I**, **--head** option.

(**FTP**) Specifies a custom FTP command to use instead of LIST when doing file lists with FTP.

If this option is used several times, the last one will be used.

--xattr

When saving output to a file, this option tells **curl** to store certain file metadata in extended file attributes. Currently, the URL is stored in the **xdg.origin.url** attribute and, for HTTP, the content type is stored in the **mime_type** attribute. If the file system does not support extended attributes, a warning is issued.

-y, **--speed-time** *<time>*

If a download is slower than **speed-limit** bytes per second during a **speed-time** period, the download gets aborted. If **speed-time** is used, the default **speed-limit** will be 1 unless set with **-Y**.

This option controls transfers and thus will not affect slow connects etc. If this is a concern for you, try the **--connect-timeout** option.

If this option is used several times, the last one will be used.

-Y, **--speed-limit** *<speed>*	If a download is slower than this given speed, in bytes per second, for **speed-time** seconds it gets aborted. **speed-time** is set with **-y** and is 30 if not set. If this option is used several times, the last one will be used.
-z, **--time-cond** *<date expression>*|*<file>*	(HTTP) Request a file that has been modified later than the given time and date, or one that has been modified before that time. The *<date expression>* can be all sorts of date strings or if it doesn't match any internal ones, it is taken as a filename and tries to get the modification date (mtime) from *<file>* instead. See the **curl_getdate** man pages for date expression details. Start the date expression with a dash (-) to make it request for a document that is older than the given date/time, default is a document that is newer than the specified date/time. If this option is used several times, the last one will be used.
--max-redirs *<num>*	Set maximum number of redirection-followings allowed. If **-L**/**--location** is used, this option can be used to prevent **curl** from following redirections "in absurdum". By default, the limit is set to 50 redirections. Set this option to **-1** to make it limitless. If this option is used several times, the last one will be used.
-0, **--http1.0**	(**HTTP**) Forces **curl** to issue its requests using HTTP 1.0 instead of using its internally preferred: HTTP 1.1.
-1, **--tlsv1**	(**SSL**) Forces **curl** to use TSL version 1 when negotiating with a remote TLS server.
-2, **--sslv2**	(**SSL**) Forces **curl** to use SSL version 2 when negotiating with a remote SSL server.
-3, **--sslv3**	(**SSL**) Forces **curl** to use SSL version 3 when negotiating with a remote SSL server.
--3p-quote	(**FTP**) Specify arbitrary commands to send to the source server. See the **-Q**, **--quote** option for details. (Added in 7.13.0)
--3p-url	(**FTP**) Activates an FTP 3rd party transfer. Specifies the source URL to get a file from, while the "normal" URL will be used as target URL, the file that will be written/created. Note that not all FTP server allow 3rd party transfers. (Added in 7.13.0)
--3p-user	(**FTP**) Specify **user:password** for the source URL transfer. (Added in 7.13.0)

-4, **--ipv4**	If libcurl is capable of resolving an address to multiple IP versions (which it is if it is ipv6-capable), this option tells libcurl to resolve names to IPv4 addresses only.
-6, **--ipv6**	If libcurl is capable of resolving an address to multiple IP versions (which it is if it is ipv6-capable), this option tells libcurl to resolve names to IPv6 addresses only.
-#, **--progress-bar**	Make **curl** display progress information as a progress bar instead of the default statistics. If this option is used twice, the second will again disable the progress bar.

Environment Variables

The environment variables can be specified in lower case or upper case. The lower case version has precedence. **http_proxy** is an exception as it is only available in lower case.

Using an environment variable to set the proxy has the same effect as using the **--proxy** option.

http_proxy [*protocol*:**//**]<*host*>[**:***port*]	Sets the proxy server to use for HTTP.
HTTPS_PROXY [*protocol*:**//**]<*host*>[**:***port*]	Sets the proxy server to use for HTTPS.
[*url-protocol*]**_PROXY** [*protocol*:**//**]<*host*>[**:***port*]	Sets the proxy server to use for [*url-protocol*], where the *protocol* is a protocol that **curl** supports and as specified in a URL: **FTP**, **FTPS**, **POP3**, **IMAP**, **SMTP**, **LDAP** etc.
ALL_PROXY [*protocol*:**//**]<*host*>[**:***port*]	Sets the proxy server to use if no protocol-specific proxy is set.
NO_PROXY<*comma-separated list of hosts*>	list of host names that shouldn't go through any proxy. If set to a asterisk '*****' only, it matches all hosts.

Exit Codes

There are a bunch of different error codes and their corresponding error messages that may appear during bad conditions. At the time of this writing, the exit codes are:

1	Unsupported protocol. This build of curl has no support for this protocol.

2	Failed to initialize.
3	URL malformed. The syntax was not correct.
4	A feature or option that was needed to perform the desired request was not enabled or was explicitly disabled at build-time. To make curl able to do this, you probably need another build of libcurl!
5	Couldn't resolve proxy. The given proxy host could not be resolved.
6	Couldn't resolve host. The given remote host was not resolved.
7	Failed to connect to host.
8	FTP weird server reply. The server sent data curl couldn't parse.
9	FTP access denied. The server denied login or denied access to the particular resource or directory you wanted to reach. Most often you tried to change to a directory that doesn't exist on the server.
11	FTP weird PASS reply. Curl couldn't parse the reply sent to the PASS request.
13	FTP weird PASV reply, Curl couldn't parse the reply sent to the PASV request.
14	FTP weird 227 format. Curl couldn't parse the 227-line the server sent.
15	FTP can't get host. Couldn't resolve the host IP we got in the 227-line.
17	FTP couldn't set binary. Couldn't change transfer

	method to binary.
18	Partial file. Only a part of the file was transferred.
19	FTP couldn't download/access the given file, the RETR (or similar) command failed.
21	FTP quote error. A quote command returned error from the server.
22	HTTP page not retrieved. The requested url was not found or returned another error with the HTTP error code being 400 or above. This return code only appears if **-f**, **--fail** is used.
23	Write error. Curl couldn't write data to a local filesystem or similar.
25	FTP couldn't STOR file. The server denied the STOR operation, used for FTP uploading.
26	Read error. Various reading problems.
27	Out of memory. A memory allocation request failed.
28	Operation timeout. The specified time-out period was reached according to the conditions.
30	FTP PORT failed. The PORT command failed. Not all FTP servers support the PORT command, try doing a transfer using PASV instead!
31	FTP couldn't use REST. The REST command failed. This command is used for resumed FTP transfers.
33	HTTP range error. The range "command" didn't work.

34	HTTP post error. Internal post-request generation error.
35	SSL connect error. The SSL handshaking failed.
36	FTP bad download resume. Couldn't continue an earlier aborted download.
37	FILE couldn't read file. Failed to open the file. Permissions?
38	LDAP cannot bind. LDAP bind operation failed.
39	LDAP search failed.
41	Function not found. A required LDAP function was not found.
42	Aborted by callback. An application told curl to abort the operation.
43	Internal error. A function was called with a bad parameter.
45	Interface error. A specified outgoing interface could not be used.
47	Too many redirects. When following redirects, curl hit the maximum amount.
48	Unknown option specified to libcurl. This indicates that you passed a weird option to curl that was passed on to libcurl and rejected. Read up in the manual!
49	Malformed telnet option.
51	The peer's SSL certificate or SSH MD5

	fingerprint was not OK.
52	The server didn't reply anything, which here is considered an error.
53	SSL crypto engine not found.
54	Cannot set SSL crypto engine as default.
55	Failed sending network data.
56	Failure in receiving network data.
58	Problem with the local certificate.
59	Couldn't use specified SSL cipher.
60	Peer certificate cannot be authenticated with known CA certificates.
61	Unrecognized transfer encoding.
62	Invalid LDAP URL.
63	Maximum file size exceeded.
64	Requested FTP SSL level failed.
65	Sending the data requires a rewind that failed.
66	Failed to initialise SSL Engine.
67	The user name, password, or similar was not accepted and curl failed to log in.
68	File not found on TFTP server.

69	Permission problem on TFTP server.
70	Out of disk space on TFTP server.
71	Illegal TFTP operation.
72	Unknown TFTP transfer ID.
73	File already exists (TFTP).
74	No such user (TFTP).
75	Character conversion failed.
76	Character conversion functions required.
77	Problem with reading the SSL CA cert (path? access rights?).
78	The resource referenced in the URL does not exist.
79	An unspecified error occurred during the SSH session.
80	Failed to shut down the SSL connection.
82	Could not load CRL file, missing or wrong format (added in 7.19.0).
83	Issuer check failed (added in 7.19.0).
84	The FTP PRET command failed
85	RTSP: mismatch of CSeq numbers

86	RTSP: mismatch of Session Identifiers
87	unable to parse FTP file list
88	FTP chunk callback reported error

curl examples

```
curl https://www.computerhope.com/index.htm
```

Fetch the file **index.htm** from **www.computerhope.com** using the **HTTP** protocol, and display it to standard output. This is essentially the same as "viewing the source" of the webpage; the raw HTML will be displayed.

```
curl https://www.computerhope.com/index.htm > index.htm
```

Fetch the same file as above, but redirect the output to a file, **index.htm**, in the current directory.

```
curl -O https://www.computerhope.com/index.htm
```

Fetch the same file as above, and output to a file with the same name (**index.htm**) in the current directory, this time using the curl function **-O**.

```
curl --limit-rate 1234B -O https://www.computerhope.com/index.htm
```

Same as above, but this time, limit the download speed to (an average speed of) 1,234 bytes/second.

```
curl --limit-rate 1234B -O -# https://www.computerhope.com/index.htm
```

Same as above, but this time, display a progress bar (the **-#** option) instead of the numerical progress meter.

Related commands

wget — Download files via HTTP or FTP.