NAME
curl – transfer a URL

SYNOPSIS
curl [options / URLs]

DESCRIPTION
curl is a tool to transfer data from or to a server, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP). The command is designed to work without user interaction.

curl offers a busload of useful tricks like proxy support, user authentication, FTP upload, HTTP post, SSL connections, cookies, file transfer resume, Metalink, and more. As you will see below, the number of features will make your head spin!

curl is powered by libcurl for all transfer-related features. See *libcurl(3)* for details.

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.example.com/file[1-100].txt

  ftp://ftp.example.com/file[001-100].txt   (with leading zeros)

  ftp://ftp.example.com/file[a-z].txt

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

  http://example.com/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 command line options and URLs mixed and in any order on the command line.

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

  http://example.com/file[1-100:10].txt

  http://example.com/file[a-z:2].txt

When using [] or {} sequences when invoked from a command line prompt, you probably have to put the full URL within double quotes to avoid the shell from interfering with it. This also goes for other characters treated special, like for example '&', '?' and '*'.

Provide the IPv6 zone index in the URL with an escaped percentage sign and the interface name. Like in

  http://[fe80::3%25eth0]/

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.

PROGRESS METER

curl normally displays a progress meter during operations, indicating the amount of transferred data, transfer speeds and estimated time left, etc. The progress meter displays number of bytes and the speeds are in bytes per second. The suffixes (k, M, G, T, P) are 1024 based. For example 1k is 1024 bytes. 1M is 1048576 bytes.

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 shell redirect (>), *-o, --output* 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, *-#, --progress-bar* is your friend. You can also disable the progress meter completely with the *-s, --silent* option.

OPTIONS

Options start with one or two dashes. Many of the options require an additional value next to them.

The short "single-dash" form of the options, -d for example, may be used with or without a space between it and its value, although a space is a recommended separator. The long "double-dash" form, *-d, --data* for example, requires a space between it and its value.

Short version options that don't need any additional values can be used immediately next to each other, like for example you can specify all the options -O, -L and -v at once as -OLv.

In general, all boolean options are enabled with --**option** and yet again disabled with --**no-**option. That is, you use the exact same option name but prefix it with "no-". However, in this list we mostly only list and show the --option version of them. (This concept with --no options was added in 7.19.0. Previously most options were toggled on/off on repeated use of the same command line option.)

--abstract-unix-socket <path>
>   (HTTP) Connect through an abstract Unix domain socket, instead of using the network. Note: netstat shows the path of an abstract socket prefixed with '@', however the <path> argument should not have this leading character.

>   Added in 7.53.0.

--alt-svc <file name>
>   (HTTPS) WARNING: this option is experimental. Do not use in production.

>   This option enables the alt-svc parser in curl. If the file name points to an existing alt-svc cache file, that will be used. After a completed transfer, the cache will be saved to the file name again if it has been modified.

Specify a "" file name (zero length) to avoid loading/saving and make curl just handle the cache in memory.

If this option is used several times, curl will load contents from all the files but the the last one will be used for saving.

Added in 7.64.1.

--anyauth

(HTTP) Tells curl to figure out authentication method by itself, and use the most secure one the remote site claims to support. This is done by first doing a request and checking the response-headers, thus possibly inducing an extra network round-trip. This is used instead of setting a specific authentication method, which you can do with *--basic*, *--digest*, *--ntlm*, and *--negotiate*.

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.

Used together with *-u, --user*.

See also *--proxy-anyauth* and *--basic* and *--digest*.

-a, --append

(FTP SFTP) When used in an upload, this makes curl append to the target file instead of overwriting it. If the remote file doesn't exist, it will be created. Note that this flag is ignored by some SFTP servers (including OpenSSH).

--basic   (HTTP) Tells curl to use HTTP Basic authentication with the remote host. 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*, or *--negotiate*).

Used together with *-u, --user*.

See also *--proxy-basic*.

--cacert <file>

(TLS) 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 it 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.

(iOS and macOS only) If curl is built against Secure Transport, then this option is supported for backward compatibility with other SSL engines, but it should not be set. If the option is not set, then curl will use the certificates in the system and user Keychain to verify the peer, which is the preferred method of verifying the peer's certificate chain.

(Schannel only) This option is supported for Schannel in Windows 7 or later with libcurl 7.60 or later. This option is supported for backward compatibility with other SSL engines; instead it is

recommended to use Windows' store of root certificates (the default for Schannel).

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

--capath <dir>
    (TLS) Tells curl to use the specified certificate directory to verify the peer. Multiple paths can be provided by separating them with ":" (e.g. "path1:path2:path3"). The certificates must be in PEM format, and if curl is built against OpenSSL, the directory must have been processed using the c_rehash utility supplied with OpenSSL. Using *--capath* can allow OpenSSL-powered curl to make SSL-connections much more efficiently than using *--cacert* if the --cacert file contains many CA certificates.

    If this option is set, the default capath value will be ignored, and if it is used several times, the last one will be used.

--cert-status
    (TLS) Tells curl to verify the status of the server certificate by using the Certificate Status Request (aka. OCSP stapling) TLS extension.

    If this option is enabled and the server sends an invalid (e.g. expired) response, if the response suggests that the server certificate has been revoked, or no response at all is received, the verification fails.

    This is currently only implemented in the OpenSSL, GnuTLS and NSS backends.

    Added in 7.41.0.

--cert-type <type>
    (TLS) Tells curl what type the provided client certificate is using. PEM, DER, ENG and P12 are recognized types. If not specified, PEM is assumed.

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

    See also *-E, --cert* and *--key* and *--key-type*.

-E, --cert <certificate[:password]>
    (TLS) 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 PKCS#12 format if using Secure Transport, or PEM format if using any other engine. 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 client certificate concatenated! See *-E, --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.

    If curl is built against OpenSSL library, and the engine pkcs11 is available, then a PKCS#11 URI (RFC 7512) can be used to specify a certificate located in a PKCS#11 device. A string beginning with "pkcs11:" will be interpreted as a PKCS#11 URI. If a PKCS#11 URI is provided, then the *--engine* option will be set as "pkcs11" if none was provided and the *--cert-type* option will be set as "ENG" if none was provided.

    (iOS and macOS only) If curl is built against Secure Transport, then the certificate string can either

be the name of a certificate/private key in the system or user keychain, or the path to a PKCS#12-encoded certificate and private key. If you want to use a file from the current directory, please precede it with "./" prefix, in order to avoid confusion with a nickname.

(Schannel only) Client certificates must be specified by a path expression to a certificate store. (Loading PFX is not supported; you can import it to a store first). You can use "<store location>\<store name>\<thumbprint>" to refer to a certificate in the system certificates store, for example, "CurrentUser\MY\934a7ac6f8a5d579285a74fa61e19f23ddfe8d7a". Thumbprint is usually a SHA-1 hex string which you can see in certificate details. Following store locations are supported: CurrentUser, LocalMachine, CurrentService, Services, CurrentUserGroupPolicy, LocalMachineGroupPolicy, LocalMachineEnterprise.

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

See also *--cert-type* and *--key* and *--key-type*.

--ciphers <list of ciphers>
    (TLS) Specifies which ciphers to use in the connection. The list of ciphers must specify valid ciphers. Read up on SSL cipher list details on this URL:

    https://curl.haxx.se/docs/ssl-ciphers.html

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

--compressed-ssh
    (SCP SFTP) Enables built-in SSH compression. This is a request, not an order; the server may or may not do it.

    Added in 7.56.0.

--compressed
    (HTTP) Request a compressed response using one of the algorithms curl supports, and save the uncompressed document. If this option is used and the server sends an unsupported encoding, curl will report an error.

-K, --config <file>

    Specify a text file to read curl arguments from. The command line arguments found in the text file will be used as if they were provided on the command line.

    Options and their parameters must be specified on the same line in the file, separated by whitespace, colon, or the equals sign. Long option names can optionally be given in the config file without the initial double dashes and if so, the colon or equals characters can be used as separators. If the option is specified with one or two dashes, there can be no colon or equals character between the option and its parameter.

    If the parameter contains whitespace (or starts with : or =), the parameter must be enclosed within quotes. Within double quotes, the following escape sequences are available: \\, \", \t, \n, \r and \v. A backslash preceding any other letter is ignored. 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 to *-K, --config* 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 simply writing the URL on its own line. So, it could look similar to this:

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

When curl is invoked, it (unless *-q, --disable* 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 '%USERPROFILE%\Application Data'.

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

```
# --- Example file ---
# this is a comment
url = "example.com"
output = "curlhere.html"
user-agent = "superagent/1.0"

# and fetch another URL too
url = "example.com/docs/manpage.html"
-O
referer = "http://nowhereatall.example.com/"
# --- End of example file ---
```

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

--connect-timeout <seconds>
> Maximum time in seconds that you allow curl's connection to take. This only limits the connection phase, so if curl connects within the given period it will continue - if not it will exit. Since version 7.32.0, this option accepts decimal values.
>
> If this option is used several times, the last one will be used.
>
> See also *-m, --max-time*.

--connect-to <HOST1:PORT1:HOST2:PORT2>

> For a request to the given HOST1:PORT1 pair, connect to HOST2:PORT2 instead. This option is suitable to direct requests at a specific server, e.g. at a specific cluster node in a cluster of servers. This option is only used to establish the network connection. It does NOT affect the hostname/port that is used for TLS/SSL (e.g. SNI, certificate verification) or for the application protocols. "HOST1" and "PORT1" may be the empty string, meaning "any host/port". "HOST2" and "PORT2" may also be the empty string, meaning "use the request's original host/port".
>
> A "host" specified to this option is compared as a string, so it needs to match the name used in request URL. It can be either numerical such as "127.0.0.1" or the full host name such as "example.org".
>
> This option can be used many times to add many connect rules.
>
> See also *--resolve* and *-H, --header*. Added in 7.49.0.

-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, counting 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.
>
> See also *-r, --range*.

-c, --cookie-jar <filename>

> (HTTP) Specify to which file you want curl to write all cookies after a completed operation. Curl writes all cookies from its in-memory cookie storage to the given file at the end of operations. If no cookies are known, no data 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.
>
> If the cookie jar can't be created or written to, the whole curl operation won't fail or even report an error clearly. Using *-v, --verbose* will get a warning displayed, but that is the only visible feedback you get about this possibly lethal situation.
>
> If this option is used several times, the last specified file name will be used.

-b, --cookie <data|filename>

> (HTTP) Pass the data to the HTTP server in the Cookie header. It is supposedly 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 '=' symbol is used in the argument, it is instead treated as a filename to read previously stored cookie from. This option also activates the cookie engine which will make curl record incoming cookies, which may be handy if you're using this in combination with the *-L, --location* option or do multiple URL transfers on the same invoke. If the file name is exactly a minus ("-"), curl will instead the contents from stdin.
>
> The file format of the file to read cookies from should be plain HTTP headers (Set-Cookie style) or the Netscape/Mozilla cookie file format.
>
> The file specified with *-b, --cookie* is only used as input. No cookies will be written to the file. To store cookies, use the *-c, --cookie-jar* option.
>
> Exercise caution if you are using this option and multiple transfers may occur. If you use the NAME1=VALUE1; format, or in a file use the Set-Cookie format and don't specify a domain, then the cookie is sent for any domain (even after redirects are followed) and cannot be modified by a server-set cookie. If the cookie engine is enabled and a server sets a cookie of the same name then both will be sent on a future transfer to that server, likely not what you intended. To address these issues set a domain in Set-Cookie (doing that will include sub domains) or use the Netscape format.
>
> If this option is used several times, the last one will be used.
>
> Users very often want to both read cookies from a file and write updated cookies back to a file, so using both *-b, --cookie* and *-c, --cookie-jar* in the same command line is common.

--create-dirs
      When used in conjunction with the *-o, --output* option, curl will create the necessary local direc
      tory hierarchy as needed. This option creates the dirs mentioned with the *-o, --output* option, noth
      ing else. If the --output file name uses no dir or if the dirs it mentions already exist, no dir will be
      created.

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

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

      (SMTP added in 7.40.0)

--crlfile <file>
      (TLS) 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.

--data-ascii <data>
      (HTTP) This is just an alias for *-d, --data*.

--data-binary <data>
      (HTTP) This posts data exactly as specified with no extra processing whatsoever.

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

      Like *-d, --data* the default content-type sent to the server is application/x-www-form-urlencoded.
      If you want the data to be treated as arbitrary binary data by the server then set the content-type to
      octet-stream: -H "Content-Type: application/octet-stream".

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

--data-raw <data>
      (HTTP) This posts data similarly to *-d, --data* but without the special interpretation of the @ char
      acter.

      See also *-d, --data*. Added in 7.43.0.

--data-urlencode <data>
      (HTTP) This posts data, similar to the other *-d, --data* options with the exception that this per
      forms URL-encoding.

      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, result-
>   ing in *name=urlencoded-file-content*. Note that the name is expected to be URL-encoded
>   already.

See also *-d, --data* and *--data-raw*. Added in 7.18.0.

-d, --data <data>
>   (HTTP) Sends the specified data in a POST request to the HTTP server, in the same way that a
>   browser does when a user has filled in an HTML form and presses the submit button. This will
>   cause curl to pass the data to the server using the content-type application/x-www-form-urlen-
>   coded. Compare to *-F, --form*.
>
>   *--data-raw* is almost the same but does not have a special interpretation of the @ character. 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 any of these options is used more than once on the same command line, the data pieces specified
>   will be merged together with a separating &-symbol. 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 letter @, the rest should be a file name to read the data from, or - if
>   you want curl to read the data from stdin. Multiple files can also be specified. Posting data from a
>   file named 'foobar' would thus be done with *-d, --data* @foobar. When --data is told to read from
>   a file like that, carriage returns and newlines will be stripped out. If you don't want the @ charac-
>   ter to have a special interpretation use *--data-raw* instead.
>
>   See also *--data-binary* and *--data-urlencode* and *--data-raw*. This option overrides *-F, --form* and
>   *-I, --head* and *-T, --upload-file*.

--delegation <LEVEL>
>   (GSS/kerberos) Set LEVEL to tell the server what it is allowed to delegate when it comes to user
>   credentials.

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 an authentication scheme that prevents the
>   password from being sent over the wire in clear text. Use this in combination with the normal *-u,
>   --user* option to set user name and password.
>
>   If this option is used several times, only the first one is used.
>
>   See also *-u, --user* and *--proxy-digest* and *--anyauth*. This option overrides *--basic* and *--ntlm* and
>   *--negotiate*.

--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, and may not work on all servers, but they 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*.

       If the server is accessed using IPv6, this option will have no effect as EPRT is necessary then.

       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) (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*.

       If the server is an IPv6 host, this option will have no effect as EPSV is necessary then.

       Disabling EPSV only changes the passive behavior. If you want to switch to active mode you need to use *-P, --ftp-port*.

-q, --disable

       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.

--disallow-username-in-url

       (HTTP) This tells curl to exit if passed a url containing a username.

       See also *--proto*. Added in 7.61.0.

--dns-interface <interface>

       (DNS) Tell curl to send outgoing DNS requests through <interface>. This option is a counterpart to *--interface* (which does not affect DNS). The supplied string must be an interface name (not an address).

       See also *--dns-ipv4-addr* and *--dns-ipv6-addr*. *--dns-interface* requires that the underlying libcurl was built to support c-ares. Added in 7.33.0.

--dns-ipv4-addr <address>

       (DNS) Tell curl to bind to <ip-address> when making IPv4 DNS requests, so that the DNS requests originate from this address. The argument should be a single IPv4 address.

       See also *--dns-interface* and *--dns-ipv6-addr*. *--dns-ipv4-addr* requires that the underlying libcurl was built to support c-ares. Added in 7.33.0.

--dns-ipv6-addr <address>

       (DNS) Tell curl to bind to <ip-address> when making IPv6 DNS requests, so that the DNS requests originate from this address. The argument should be a single IPv6 address.

       See also *--dns-interface* and *--dns-ipv4-addr*. *--dns-ipv6-addr* requires that the underlying libcurl was built to support c-ares. Added in 7.33.0.

--dns-servers <addresses>
>   Set the list of DNS servers to be used instead of the system default. The list of IP addresses
>   should be separated with commas. Port numbers may also optionally be given as *:<port-number>*
>   after each IP address.
>
>   *--dns-servers* requires that the underlying libcurl was built to support c-ares. Added in 7.33.0.

--doh-url <URL>
>   (all) Specifies which DNS-over-HTTPS (DOH) server to use to resolve hostnames, instead of us-
>   ing the default name resolver mechanism. The URL must be HTTPS.
>
>   If this option is used several times, the last one will be used.
>
>   Added in 7.62.0.

-D, --dump-header <filename>
>   (HTTP FTP) Write the received protocol headers to the specified file.
>
>   This option is handy to use when you want to store the headers that an HTTP site sends to you.
>   Cookies from the headers could then be read in a second curl invocation by using the *-b, --cookie*
>   option! The *-c, --cookie-jar* option is a better way to store cookies.
>
>   If no headers are received, the use of this option will create an empty file.
>
>   When used in 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.
>
>   See also *-o, --output*.

--egd-file <file>
>   (TLS) Specify the path name to the Entropy Gathering Daemon socket. The socket is used to seed
>   the random engine for SSL connections.
>
>   See also *--random-file*.

--engine <name>
>   (TLS) 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.

--etag-compare <file>
>   (HTTP) This option makes a conditional HTTP request for the specific ETag read from the given
>   file by sending a custom If-None-Match header using the extracted ETag.
>
>   For correct results, make sure that specified file contains only a single line with a desired ETag. An
>   empty file is parsed as an empty ETag.
>
>   Use the option *--etag-save* to first save the ETag from a response, and then use this option to com-
>   pare using the saved ETag in a subsequent request.
>
>   OMPARISON: There are 2 types of comparison or ETags, Weak and Strong. This option expects,
>   and uses a strong comparison.
>
>   Added in 7.68.0.

--etag-save <file>

(HTTP) This option saves an HTTP ETag to the specified file. Etag is usually part of headers returned by a request. When server sends an ETag, it must be enveloped by a double quote. This option extracts the ETag without the double quotes and saves it into the <file>.

A server can send a week ETag which is prefixed by "W/". This identifier is not considered, and only relevant ETag between quotation marks is parsed.

It an ETag wasn't send by the server or it cannot be parsed, and empty file is created.

Added in 7.68.0.

--expect100-timeout <seconds>

(HTTP) Maximum time in seconds that you allow curl to wait for a 100-continue response when curl emits an Expects: 100-continue header in its request. By default curl will wait one second. This option accepts decimal values! When curl stops waiting, it will continue as if the response has been received.

See also *--connect-timeout*. Added in 7.47.0.

--fail-early

Fail and exit on the first detected transfer error.

When curl is used to do multiple transfers on the command line, it will attempt to operate on each given URL, one by one. By default, it will ignore errors if there are more URLs given and the last URL's success will determine the error code curl returns. So early failures will be "hidden" by subsequent successful transfers.

Using this option, curl will instead return an error on the first transfer that fails, independent of the amount of URLs that are given on the command line. This way, no transfer failures go undetected by scripts and similar.

This option is global and does not need to be specified for each use of *-:, --next*.

This option does not imply *-f, --fail*, which causes transfers to fail due to the server's HTTP status code. You can combine the two options, however note *-f, --fail* is not global and is therefore contained by *-:, --next*.

Added in 7.52.0.

-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 an HTTP server fails to deliver a document, it returns an HTML document stating so (which often also describes why and more). 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).

--false-start

(TLS) Tells curl to use false start during the TLS handshake. False start is a mode where a TLS client will start sending application data before verifying the server's Finished message, thus saving a round trip when performing a full handshake.

This is currently only implemented in the NSS and Secure Transport (on iOS 7.0 or later, or OS X 10.9 or later) backends.

Added in 7.42.0.

--form-string <name=string>

(HTTP SMTP IMAP) Similar to *-F, --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 *-F, --form* if there's any possibility that the string value may accidentally trigger the '@' or '<' features of *-F, --form*.

See also *-F, --form*.

-F, --form <name=content>

(HTTP SMTP IMAP) For HTTP protocol family, 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 RFC 2388.

For SMTP and IMAP protocols, this is the mean to compose a multipart mail message to transmit.

This enables uploading of binary files etc. To force the 'content' part to be a file, prefix the file name with an @ sign. To just get the content part from a file, prefix the file name with the symbol <. The difference between @ and < is then 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.

Tell curl to read content from stdin instead of a file by using - as filename. This goes for both @ and < constructs. When stdin is used, the contents is buffered in memory first by curl to determine its size and allow a possible resend. Defining a part's data from a named non-regular file (such as a named pipe or similar) is unfortunately not subject to buffering and will be effectively read at transmission time; since the full size is unknown before the transfer starts, such data is sent as chunks by HTTP and rejected by IMAP.

Example: send an image to an HTTP server, where 'profile' is the name of the form-field to which the file portrait.jpg will be the input:

 curl -F profile=@portrait.jpg https://example.com/upload.cgi

Example: send a your name and shoe size in two text fields to the server:

 curl -F name=John -F shoesize=11 https://example.com/

Example: send a your essay in a text field to the server. Send it as a plain text field, but get the contents for it from a local file:

 curl -F "story=<hugefile.txt" https://example.com/

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" example.com

or

 curl -F "name=daniel;type=text/foo" example.com

You can also explicitly change the name field of a file upload part by setting filename=, like this:

 curl -F "file=@localfile;filename=nameinpost" example.com

If filename/path contains ',' or ';', it must be quoted by double-quotes like:

curl -F "file=@\"localfile\";filename=\"nameinpost\"" example.com

or

curl -F 'file=@"localfile";filename="nameinpost"' example.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.

Quoting must also be applied to non-file data if it contains semicolons, leading/trailing spaces or leading double quotes:

curl -F 'colors="red; green; blue";type=text/x-myapp' example.com

You can add custom headers to the field by setting headers=, like

curl -F "submit=OK;headers=\"X-submit-type: OK\"" example.com

or

curl -F "submit=OK;headers=@headerfile" example.com

The headers= keyword may appear more that once and above notes about quoting apply. When headers are read from a file, Empty lines and lines starting with '#' are comments and ignored; each header can be folded by splitting between two words and starting the continuation line with a space; embedded carriage-returns and trailing spaces are stripped. Here is an example of a header file contents:

This file contain two headers.
X-header-1: this is a header

The following header is folded.
X-header-2: this is
 another header

To support sending multipart mail messages, the syntax is extended as follows:
- name can be omitted: the equal sign is the first character of the argument,
- if data starts with '(', this signals to start a new multipart: it can be followed by a content type specification.
- a multipart can be terminated with a '=)' argument.

Example: the following command sends an SMTP mime e-mail consisting in an inline part in two alternative formats: plain text and HTML. It attaches a text file:

curl -F '=(;type=multipart/alternative' \
     -F '=plain text message' \
     -F '= <body>HTML message</body>;type=text/html' \
   -F '=)' -F '=@textfile.txt' ... smtp://example.com

Data can be encoded for transfer using encoder=. Available encodings are *binary* and *8bit* that do nothing else than adding the corresponding Content-Transfer-Encoding header, *7bit* that only rejects 8-bit characters with a transfer error, *quoted-printable* and *base64* that encodes data according to the corresponding schemes, limiting lines length to 76 characters.

Example: send multipart mail with a quoted-printable text message and a base64 attached file:

```
 curl -F '=text message;encoder=quoted-printable' \
     -F '=@localfile;encoder=base64' ... smtp://example.com
```

See further examples and details in the MANUAL.

This option can be used multiple times.

This option overrides *-d, --data* and *-I, --head* and *-T, --upload-file*.

--ftp-account <data>
>   (FTP) When an FTP server asks for "account data" after user name and password has been pro-
>   vided, this data is sent off using the ACCT command.
>
>   If this option is used several times, the last one will be used.
>
>   Added in 7.13.0.

--ftp-alternative-to-user <command>
>   (FTP) If authenticating with the USER and PASS commands fails, 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-create-dirs
>   (FTP SFTP) When an FTP or SFTP 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.
>
>   See also *--create-dirs*.

--ftp-method <method>
>   (FTP) Control what method curl should use to reach a file on an FTP(S) server. The method argu-
>   ment should be one of the following alternatives:

multicwd
>   curl does a single CWD operation for each path part in the given URL. For deep hierar-
>   chies this means very many commands. This is how RFC 1738 says it should be done.
>   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'.

Added in 7.15.1.

--ftp-pasv
>   (FTP) Use passive mode for the data connection. Passive is the internal default behavior, but using
>   this option can be used to override a previous *-P, --ftp-port* option.
>
>   If this option is used several times, only the first one is used. 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.

See also *--disable-epsv*. Added in 7.11.0.

-P, --ftp-port <address>

(FTP) Reverses the default initiator/listener roles when connecting with FTP. This option makes curl use active mode. 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
e.g. "eth0" to specify which interface's IP address you want to use (Unix only)

IP address
e.g. "192.168.10.1" to specify the exact IP address

host name
e.g. "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++.

Since 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.

See also *--ftp-pasv* and *--disable-eprt*.

--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 7.20.0.

--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.

This option has no effect if PORT, EPRT or EPSV is used instead of PASV.

See also *--ftp-pasv*. Added in 7.14.2.

--ftp-ssl-ccc-mode <active/passive>

(FTP) 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.

See also *--ftp-ssl-ccc*. Added in 7.16.2.

--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 also *--ssl* and *--ftp-ssl-ccc-mode*. Added in 7.16.1.

--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.

-G, --get

When used, this option will make all data specified with *-d, --data*, *--data-binary* or *--data-urlencode* to be used in an 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, --head*, 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.

-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.

--happy-eyeballs-timeout-ms <milliseconds>

Happy eyeballs is an algorithm that attempts to connect to both IPv4 and IPv6 addresses for dual-stack hosts, preferring IPv6 first for the number of milliseconds. If the IPv6 address cannot be connected to within that time then a connection attempt is made to the IPv4 address in parallel. The first connection to be established is the one that is used.

The range of suggested useful values is limited. Happy Eyeballs RFC 6555 says "It is RECOMMENDED that connection attempts be paced 150-250 ms apart to balance human factors against network load." libcurl currently defaults to 200 ms. Firefox and Chrome currently default to 300 ms.

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

Added in 7.59.0.

--haproxy-protocol

(HTTP) Send a HAProxy PROXY protocol v1 header at the beginning of the connection. This is used by some load balancers and reverse proxies to indicate the client's true IP address and port.

This option is primarily useful when sending test requests to a service that expects this header.

Added in 7.60.0.

-I, --head

(HTTP FTP FILE) Fetch the headers 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.

-H, --header <header/@file>

(HTTP) Extra header to include in the request when sending HTTP to a server. 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 is 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.

Starting in 7.55.0, this option can take an argument in @filename style, which then adds a header for each line in the input file. Using @- will make curl read the header file from stdin.

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

Starting in 7.37.0, you need *--proxy-header* to send custom headers intended for a proxy.

Example:

 curl -H "X-First-Name: Joe" http://example.com/

WARNING: headers set with this option will be set in all requests - even after redirects are followed, like when told with *-L, --location*. This can lead to the header being sent to other hosts than the original host, so sensitive headers should be used with caution combined with following redirects.

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

-h, --help
> Usage help. This lists all current command line options with a short description.

--hostpubmd5 <md5>
> (SFTP SCP) 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.

--http0.9
> (HTTP) Tells curl to be fine with HTTP version 0.9 response.

> HTTP/0.9 is a completely headerless response and therefore you can also connect with this to non-HTTP servers and still get a response since curl will simply transparently downgrade - if allowed.

> Since curl 7.66.0, HTTP/0.9 is disabled by default.

-0, --http1.0
> (HTTP) Tells curl to use HTTP version 1.0 instead of using its internally preferred HTTP version.

> This option overrides *--http1.1* and *--http2*.

--http1.1
> (HTTP) Tells curl to use HTTP version 1.1.

> This option overrides *-0, --http1.0* and *--http2*. Added in 7.33.0.

--http2-prior-knowledge
> (HTTP) Tells curl to issue its non-TLS HTTP requests using HTTP/2 without HTTP/1.1 Upgrade. It requires prior knowledge that the server supports HTTP/2 straight away. HTTPS requests will

still do HTTP/2 the standard way with negotiated protocol version in the TLS handshake.

--http2-prior-knowledge requires that the underlying libcurl was built to support HTTP/2. This option overrides *--http1.1* and *-0, --http1.0* and *--http2*. Added in 7.49.0.

--http2   (HTTP) Tells curl to use HTTP version 2.

See also *--http1.1* and *--http3*. *--http2* requires that the underlying libcurl was built to support HTTP/2. This option overrides *--http1.1* and *-0, --http1.0* and *--http2-prior-knowledge*. Added in 7.33.0.

--http3   (HTTP) WARNING: this option is experimental. Do not use in production.

Tells curl to use HTTP version 3 directly to the host and port number used in the URL. A normal HTTP/3 transaction will be done to a host and then get redirected via Alt-SVc, but this option allows a user to circumvent that when you know that the target speaks HTTP/3 on the given host and port.

This option will make curl fail if a QUIC connection cannot be established, it cannot fall back to a lower HTTP version on its own.

See also *--http1.1* and *--http2*. *--http3* requires that the underlying libcurl was built to support HTTP/3. This option overrides *--http1.1* and *-0, --http1.0* and *--http2* and *--http2-prior-knowledge*. Added in 7.66.0.

--ignore-content-length
           (FTP HTTP) For 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.

For FTP (since 7.46.0), skip the RETR command to figure out the size before downloading a file.

-i, --include
           Include the HTTP response headers in the output. The HTTP response headers can include things like server name, cookies, date of the document, HTTP version and more...

To view the request headers, consider the *-v, --verbose* option.

See also *-v, --verbose*.

-k, --insecure
           (TLS) By default, every SSL connection curl makes is verified to be secure. This option allows curl to proceed and operate even for server connections otherwise considered insecure.

The server connection is verified by making sure the server's certificate contains the right name and verifies successfully using the cert store.

See this online resource for further details:
 https://curl.haxx.se/docs/sslcerts.html

See also *--proxy-insecure* and *--cacert*.

--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 https://www.example.com/

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

On Linux it can be used to specify a VRF, but the binary needs to either have CAP_NET_RAW or to be run as root. More information about Linux VRF: https://www.kernel.org/doc/Documenta-tion/networking/vrf.txt

See also *--dns-interface*.

-4, --ipv4
> This option tells curl to resolve names to IPv4 addresses only, and not for example try IPv6.
>
> See also *--http1.1* and *--http2*. This option overrides *-6, --ipv6*.

-6, --ipv6
> This option tells curl to resolve names to IPv6 addresses only, and not for example try IPv4.
>
> See also *--http1.1* and *--http2*. This option overrides *-4, --ipv4*.

-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.
>
> See also *-b, --cookie* and *-c, --cookie-jar*.

--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.
>
> If this option is used several times, the last one will be used. If unspecified, the option defaults to 60 seconds.
>
> Added in 7.18.0.

--key-type <type>
> (TLS) 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.

--key <key>
> (TLS SSH) Private key file name. Allows you to provide your private key in this separate file. For SSH, if not specified, curl tries the following candidates in order: '~/.ssh/id_rsa', '~/.ssh/id_dsa', './id_rsa', './id_dsa'.
>
> If curl is built against OpenSSL library, and the engine pkcs11 is available, then a PKCS#11 URI (RFC 7512) can be used to specify a private key located in a PKCS#11 device. A string beginning with "pkcs11:" will be interpreted as a PKCS#11 URI. If a PKCS#11 URI is provided, then the *--engine* option will be set as "pkcs11" if none was provided and the *--key-type* option will be set as "ENG" if none was provided.
>
> 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.

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

--krb requires that the underlying libcurl was built to support Kerberos.

--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!

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

> Added in 7.16.1.

--limit-rate <speed>
> Specify the maximum transfer rate you want curl to use - for both downloads and uploads. This feature is useful if you have a limited pipe and you'd like your transfer not to use your entire bandwidth. To make it slower than it otherwise would be.

> 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.

> If you also use the *-Y, --speed-limit* option, that option will take precedence and might cripple the rate-limiting slightly, to help keeping the speed-limit logic working.

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

-l, --list-only
> (FTP POP3) (FTP) When listing an FTP directory, this switch forces a name-only view. This is especially useful if the user wants to machine-parse the contents of an FTP directory since the normal directory view doesn't use a standard look or format. When used like this, the option causes a NLST command to be sent to the server instead of LIST.

> Note: Some FTP servers list only files in their response to NLST; they do not include sub-directories and symbolic links.

> (POP3) When retrieving a specific email from POP3, this switch forces a LIST command to be performed instead of RETR. This is particularly useful if the user wants to see if a specific message id exists on the server and what size it is.

> Note: When combined with *-X, --request*, this option can be used to send an UIDL command instead, so the user may use the email's unique identifier rather than it's message id to make the request.

> Added in 7.21.5.

--local-port <num/range>
> Set a preferred single number or range (FROM-TO) 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.

--location-trusted
> (HTTP) Like *-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 to a site to which you'll send your authentication info (which is plaintext in the case of HTTP Basic authentication).

See also *-u, --user*.

-L, --location

(HTTP) 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.

You can tell curl to not change the non-GET request method to GET after a 30x response by using the dedicated options for that: *--post301*, *--post302* and *--post303*.

--login-options <options>

(IMAP POP3 SMTP) Specify the login options to use during server authentication.

You can use the login options to specify protocol specific options that may be used during authentication. At present only IMAP, POP3 and SMTP support login options. For more information about the login options please see RFC 2384, RFC 5092 and IETF draft draft-earhart-url-smtp-00.txt

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

Added in 7.34.0.

--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.

See also *--mail-rcpt* and *--mail-from*. Added in 7.25.0.

--mail-from <address>

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

See also *--mail-rcpt* and *--mail-auth*. Added in 7.20.0.

--mail-rcpt <address>

(SMTP) Specify a single address, user name or mailing list name. Repeat this option several times to send to multiple recipients.

When performing a mail transfer, the recipient should specify a valid email address to send the mail to.

When performing an address verification (VRFY command), the recipient should be specified as the user name or user name and domain (as per Section 3.5 of RFC5321). (Added in 7.34.0)

When performing a mailing list expand (EXPN command), the recipient should be specified using the mailing list name, such as "Friends" or "London-Office". (Added in 7.34.0)

Added in 7.20.0.

-M, --manual
    Manual. Display the huge help text.

--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.

    A size modifier may be used. For example, 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. (Added in 7.58.0)

    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.

    See also *--limit-rate*.

--max-redirs <num>
    (HTTP) Set maximum number of redirection-followings allowed. When *-L, --location* is used, is used to prevent curl from following redirections too much. By default, the limit is set to 50 redirections. Set this option to -1 to make it unlimited.

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

-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. 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.

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

    See also *--connect-timeout*.

--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 *-i, --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.

    --metalink requires that the underlying libcurl was built to support metalink. Added in 7.27.0.

--negotiate

        (HTTP) Enables Negotiate (SPNEGO) authentication.

        This option requires a library built with GSS-API or SSPI support. Use *-V, --version* to see if your curl supports GSS-API/SSPI or SPNEGO.

        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, --user* option aren't actually used.

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

        See also *--basic* and *--ntlm* and *--anyauth* and *--proxy-negotiate*.

--netrc-file <filename>

        This option is similar to *-n, --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, the last one will be used.

        It will abide by *--netrc-optional* if specified.

        This option overrides *-n, --netrc*. Added in 7.21.5.

--netrc-optional

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

        See also *--netrc-file*. This option overrides *-n, --netrc*.

-n, --netrc

        Makes curl scan the *.netrc (_netrc* on Windows) 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(5) ftp(1)* for details on the file format. Curl will not complain if that file doesn't have the right permissions (it should not be either world- or 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

-:, --next

        Tells curl to use a separate operation for the following URL and associated options. This allows you to send several URL requests, each with their own specific options, for example, such as different user names or custom requests for each.

        -:, --next will reset all local options and only global ones will have their values survive over to the operation following the *-:, --next* instruction. Global options include *-v, --verbose*, *--trace*, *--trace-ascii* and *--fail-early*.

        For example, you can do both a GET and a POST in a single command line:

        curl www1.example.com --next -d postthis www2.example.com

        Added in 7.36.0.

--no-alpn

    (HTTPS) Disable the ALPN TLS extension. ALPN is enabled by default if libcurl was built with an SSL library that supports ALPN. ALPN is used by a libcurl that supports HTTP/2 to negotiate HTTP/2 support with the server during https sessions.

    See also *--no-npn* and *--http2*. *--no-alpn* requires that the underlying libcurl was built to support TLS. Added in 7.36.0.

-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.

--no-keepalive

    Disables the use of keepalive messages on the TCP connection. curl otherwise enables them by default.

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

--no-npn

    (HTTPS) Disable the NPN TLS extension. NPN is enabled by default if libcurl was built with an SSL library that supports NPN. NPN is used by a libcurl that supports HTTP/2 to negotiate HTTP/2 support with the server during https sessions.

    See also *--no-alpn* and *--http2*. *--no-npn* requires that the underlying libcurl was built to support TLS. Added in 7.36.0.

--no-progress-meter

    Option to switch off the progress meter output without muting or otherwise affecting warning and informational messages like *-s, --silent* does.

    Note that this is the negated option name documented. You can thus use --progress-meter to enable the progress meter again.

    See also *-v, --verbose* and *-s, --silent*. Added in 7.67.0.

--no-sessionid

    (TLS) 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.

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

    Added in 7.16.0.

--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.

Since 7.53.0, This option overrides the environment variables that disable the proxy. If there's an environment variable disabling a proxy, you can set noproxy list to "" to override it.

Added in 7.19.4.

--ntlm-wb
(HTTP) Enables NTLM much in the style *--ntlm* does, but hand over the authentication to the separate binary ntlmauth application that is executed when needed.

See also *--ntlm* and *--proxy-ntlm*.

--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*.

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

See also *--proxy-ntlm*. *--ntlm* requires that the underlying libcurl was built to support TLS. This option overrides *--basic* and *--negotiate* and *--digest* and *--anyauth*.

--oauth2-bearer <token>
(IMAP POP3 SMTP) Specify the Bearer Token for OAUTH 2.0 server authentication. The Bearer Token is used in conjunction with the user name which can be specified as part of the *--url* or *-u, --user* options.

The Bearer Token and user name are formatted according to RFC 6750.

If this option is used several times, the last one will be 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}.example.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 the number of URLs you have. For example, if you specify two URLs on the same command line, you can use it like this:

  curl -o aa example.com -o bb example.net

and the order of the -o options and the URLs doesn't matter, just that the first -o is for the first URL and so on, so the above command line can also be written as

  curl example.com example.net -o aa -o bb

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.

See also *-O, --remote-name* and *--remote-name-all* and *-J, --remote-header-name*.

--parallel-immediate

When doing parallel transfers, this option will instruct curl that it should rather prefer opening up more connections in parallel at once rather than waiting to see if new transfers can be added as multiplexed streams on another connection.

See also *-Z, --parallel* and *--parallel-max*. Added in 7.68.0.

--parallel-max

When asked to do parallel transfers, using *-Z, --parallel*, this option controls the maximum amount of transfers to do simultaneously.

The default is 50.

See also *-Z, --parallel*. Added in 7.66.0.

-Z, --parallel

Makes curl perform its transfers in parallel as compared to the regular serial manner.

Added in 7.66.0.

--pass <phrase>

(SSH TLS) Passphrase for the private key

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

--path-as-is

Tell curl to not handle sequences of /../ or /./ in the given URL path. Normally curl will squash or merge them according to standards but with this option set you tell it not to do that.

Added in 7.42.0.

--pinnedpubkey <hashes>

(TLS) Tells curl to use the specified public key file (or hashes) to verify the peer. This can be a path to a file which contains a single public key in PEM or DER format, or any number of base64 encoded sha256 hashes preceded by ´sha256//´ and separated by ´;´

When negotiating a TLS or SSL connection, the server sends a certificate indicating its identity. A public key is extracted from this certificate and if it does not exactly match the public key provided to this option, curl will abort the connection before sending or receiving any data.

PEM/DER support:
  7.39.0: OpenSSL, GnuTLS and GSKit
  7.43.0: NSS and wolfSSL
  7.47.0: mbedtls sha256 support:
  7.44.0: OpenSSL, GnuTLS, NSS and wolfSSL
  7.47.0: mbedtls Other SSL backends not supported.

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

--post301

(HTTP) Tells curl to respect RFC 7231/6.4.2 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*.

See also *--post302* and *--post303* and *-L, --location*. Added in 7.17.1.

--post302

(HTTP) Tells curl to respect RFC 7231/6.4.3 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*.

See also *--post301* and *--post303* and *-L, --location*. Added in 7.19.1.

--post303

(HTTP) Tells curl to violate RFC 7231/6.4.4 and not convert POST requests into GET requests when following 303 redirections. A server may require a POST to remain a POST after a 303 redirection. This option is meaningful only when using *-L, --location*.

See also *--post302* and *--post301* and *-L, --location*. Added in 7.26.0.

--preproxy [protocol://]host[:port]

Use the specified SOCKS proxy before connecting to an HTTP or HTTPS *-x, --proxy*. In such a case curl first connects to the SOCKS proxy and then connects (through SOCKS) to the HTTP or HTTPS proxy. Hence pre proxy.

The pre proxy string should 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 will make curl default to SOCKS4.

If the port number is not specified in the proxy string, it is assumed to be 1080.

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.

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

Added in 7.52.0.

-#, --progress-bar

Make curl display transfer progress as a simple progress bar instead of the standard, more informational, meter.

This progress bar draws a single line of '#' characters across the screen and shows a percentage if the transfer size is known. For transfers without a known size, there will be space ship (-=o=-) that moves back and forth but only while data is being transferred, with a set of flying hash sign symbols on top.

--proto-default <protocol>

Tells curl to use *protocol* for any URL missing a scheme name.

Example:

 curl --proto-default https ftp.mozilla.org

An unknown or unsupported protocol causes error *CURLE_UNSUPPORTED_PROTOCOL* (1).

This option does not change the default proxy protocol (http).

Without this option curl would make a guess based on the host, see *--url* for details.

Added in 7.45.0.

--proto-redir <protocols>
        Tells curl to limit what protocols it may use on redirect. Protocols denied by *--proto* are not over-ridden by this option. See --proto for how protocols are represented.

        Example, allow only HTTP and HTTPS on redirect:

         curl --proto-redir -all,http,https http://example.com

        By default curl will allow HTTP, HTTPS, FTP and FTPS on redirect (7.65.2). Older versions of curl allowed all protocols on redirect except several disabled for security reasons: Since 7.19.4 FILE and SCP are disabled, and since 7.40.0 SMB and SMBS are also disabled. Specifying *all* or *+all* enables all protocols on redirect, including those disabled for security.

        Added in 7.20.2.

--proto <protocols>
        Tells curl to limit what protocols it may use in the transfer. 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.

See also *--proto-redir* and *--proto-default*. Added in 7.20.2.

--proxy-anyauth
        Tells curl to pick a suitable authentication method when communicating with the given HTTP proxy. This might cause an extra request/response round-trip.

        See also *-x, --proxy* and *--proxy-basic* and *--proxy-digest*. 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.

        See also *-x, --proxy* and *--proxy-anyauth* and *--proxy-digest*.

--proxy-cacert <file>
          Same as *--cacert* but used in HTTPS proxy context.

          See also *--proxy-capath* and *--cacert* and *--capath* and *-x, --proxy*. Added in 7.52.0.

--proxy-capath <dir>
          Same as *--capath* but used in HTTPS proxy context.

          See also *--proxy-cacert* and *-x, --proxy* and *--capath*. Added in 7.52.0.

--proxy-cert-type <type>
          Same as *--cert-type* but used in HTTPS proxy context.

          Added in 7.52.0.

--proxy-cert <cert[:passwd]>
          Same as *-E, --cert* but used in HTTPS proxy context.

          Added in 7.52.0.

--proxy-ciphers <list>
          Same as *--ciphers* but used in HTTPS proxy context.

          Added in 7.52.0.

--proxy-crlfile <file>
          Same as *--crlfile* but used in HTTPS proxy context.

          Added in 7.52.0.

--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.

          See also *-x, --proxy* and *--proxy-anyauth* and *--proxy-basic*.

--proxy-header <header/@file>
          (HTTP) Extra header to include in the request when sending HTTP to a proxy. You may specify any number of extra headers. This is the equivalent option to *-H, --header* but is for proxy communication only like in CONNECT requests when you want a separate header sent to the proxy to what is sent to the actual remote host.

          curl will make sure that each header you add/replace is 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.

          Headers specified with this option will not be included in requests that curl knows will not be sent to a proxy.

          Starting in 7.55.0, this option can take an argument in @filename style, which then adds a header for each line in the input file. Using @- will make curl read the header file from stdin.

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

          Added in 7.37.0.

--proxy-insecure
          Same as *-k, --insecure* but used in HTTPS proxy context.

Added in 7.52.0.

--proxy-key-type <type>
        Same as *--key-type* but used in HTTPS proxy context.

        Added in 7.52.0.

--proxy-key <key>
        Same as *--key* but used in HTTPS proxy context.

--proxy-negotiate
        Tells curl to use HTTP Negotiate (SPNEGO) authentication when communicating with the given proxy. Use *--negotiate* for enabling HTTP Negotiate (SPNEGO) with a remote host.

        See also *--proxy-anyauth* and *--proxy-basic*. Added in 7.17.1.

--proxy-ntlm
        Tells curl to use HTTP NTLM authentication when communicating with the given proxy. Use *--ntlm* for enabling NTLM with a remote host.

        See also *--proxy-negotiate* and *--proxy-anyauth*.

--proxy-pass <phrase>
        Same as *--pass* but used in HTTPS proxy context.

        Added in 7.52.0.

--proxy-pinnedpubkey <hashes>
        (TLS) Tells curl to use the specified public key file (or hashes) to verify the proxy. This can be a path to a file which contains a single public key in PEM or DER format, or any number of base64 encoded sha256 hashes preceded by ´sha256//´ and separated by ´;´

        When negotiating a TLS or SSL connection, the server sends a certificate indicating its identity. A public key is extracted from this certificate and if it does not exactly match the public key provided to this option, curl will abort the connection before sending or receiving any data.

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

--proxy-service-name <name>
        This option allows you to change the service name for proxy negotiation.

        Added in 7.43.0.

--proxy-ssl-allow-beast
        Same as *--ssl-allow-beast* but used in HTTPS proxy context.

        Added in 7.52.0.

--proxy-tls13-ciphers <ciphersuite list>
        (TLS) Specifies which cipher suites to use in the connection to your HTTPS proxy when it negotiates TLS 1.3. The list of ciphers suites must specify valid ciphers. Read up on TLS 1.3 cipher suite details on this URL:

        https://curl.haxx.se/docs/ssl-ciphers.html

        This option is currently used only when curl is built to use OpenSSL 1.1.1 or later. If you are using a different SSL backend you can try setting TLS 1.3 cipher suites by using the *--proxy-ciphers* option.

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

--proxy-tlsauthtype <type>
>   Same as *--tlsauthtype* but used in HTTPS proxy context.

>   Added in 7.52.0.

--proxy-tlspassword <string>
>   Same as *--tlspassword* but used in HTTPS proxy context.

>   Added in 7.52.0.

--proxy-tlsuser <name>
>   Same as *--tlsuser* but used in HTTPS proxy context.

>   Added in 7.52.0.

--proxy-tlsv1
>   Same as *-1, --tlsv1* but used in HTTPS proxy context.

>   Added in 7.52.0.

-U, --proxy-user <user:password>
>   Specify the user name and password to use for proxy authentication.

>   If you use a Windows SSPI-enabled curl binary and do either Negotiate or NTLM authentication
>   then you can tell curl to select the user name and password from your environment by specifying a
>   single colon with this option: "-U :".

>   On systems where it works, curl will hide the given option argument from process listings. This is
>   not enough to protect credentials from possibly getting seen by other users on the same system as
>   they will still be visible for a brief moment before cleared. Such sensitive data should be retrieved
>   from a file instead or similar and never used in clear text in a command line.

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

-x, --proxy [protocol://]host[:port]
>   Use the specified proxy.

>   The proxy string can be specified with a protocol:// prefix. No protocol specified or http:// will be
>   treated as HTTP proxy. Use socks4://, socks4a://, socks5:// or socks5h:// to request a specific
>   SOCKS version to be used. (The protocol support was added in curl 7.21.7)

>   HTTPS proxy support via https:// protocol prefix was added in 7.52.0 for OpenSSL, GnuTLS and
>   NSS.

>   Unrecognized and unsupported proxy protocols cause an error since 7.52.0. Prior versions may
>   ignore the protocol and use http:// instead.

>   If the port number is not specified in the proxy string, it is assumed to be 1080.

>   This option overrides existing environment variables that set the proxy to use. If there's an envi-
>   ronment 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.

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

--proxy1.0 <host[: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.

-p, --proxytunnel
When an HTTP proxy is used *-x, --proxy*, this option will make curl tunnel through the proxy. 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.

To suppress proxy CONNECT response headers when curl is set to output headers use *--suppress-connect-headers*.

See also *-x, --proxy*.

--pubkey <key>
(SFTP SCP) 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.

(As of 7.39.0, curl attempts to automatically extract the public key from the private key file, so passing this option is generally not required. Note that this public key extraction requires libcurl to be linked against a copy of libssh2 1.2.8 or higher that is itself linked against OpenSSL.)

-Q, --quote
(FTP SFTP) Send an arbitrary command to the remote FTP or SFTP server. Quote commands are sent BEFORE the transfer takes place (just after the initial PWD command in an FTP transfer, to be exact). To make commands take place after a successful transfer, prefix them with a dash '-'. To make commands be sent after curl has changed the working directory, just before the transfer command(s), prefix the command with a '+' (this is only supported for FTP). You may specify any number 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 RFC 959 defines to FTP servers, or one of the commands listed below to SFTP servers.

Prefix the command with an asterisk (*) to make curl continue even if the command fails as by default curl will stop at first failure.

This option can be used multiple times.

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

-r, --range <range>
> (HTTP FTP SFTP FILE) Retrieve a byte range (i.e. a partial document) from an HTTP/1.1, FTP or SFTP server or a local FILE. 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(*)(HTTP)

> **100-199,500-599**
> > specifies two separate 100-byte ranges(*) (HTTP)

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

> 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.

> 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 and SFTP range downloads only support the simple 'start-stop' syntax (optionally with one of the numbers omitted). FTP use depends on the extended FTP command SIZE.

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

--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.

-e, --referer <URL>
             (HTTP) Sends the "Referrer Page" information to the HTTP server. This can also be set with the
             -H, --header flag of course. When used with *-L, --location* you can append ";auto" to the *-e, --ref-
             erer* 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 *-e, --referer*.

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

             See also *-A, --user-agent* and *-H, --header*.

-J, --remote-header-name
             (HTTP) This option tells the *-O, --remote-name* option to use the server-specified Content-Dispo-
             sition filename instead of extracting a filename from the URL.

             If the server specifies a file name and a file with that name already exists in the current working di-
             rectory it will not be overwritten and an error will occur. If the server doesn't specify a file name
             then this option has no effect.

             There's no attempt to decode %-sequences (yet) in the provided file name, so this option may pro-
             vide you with rather unexpected file names.

             WARNING: Exercise judicious use of this option, especially on Windows. A rogue server could
             send you the name of a DLL or other file that could possibly be loaded automatically by Windows
             or some third party software.

--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.

-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 file will be saved in the current working directory. If you want the file saved in a different di-
             rectory, make sure you change the current working directory before invoking curl with this option.

             The remote file name to use for saving is extracted from the given URL, nothing else, and if it al-
             ready exists it will be overwritten. If you want the server to be able to choose the file name refer to
             -J, --remote-header-name which can be used in addition to this option. If the server chooses a file
             name and that name already exists it will not be overwritten.

             There is no URL decoding done on the file name. If it has %20 or other URL encoded parts of the
             name, they will end up as-is as file name.

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

-R, --remote-time
             When used, this will make curl attempt to figure out the timestamp of the remote file, and if that is
             available make the local file get that same timestamp.

--request-target
>   (HTTP) Tells curl to use an alternative "target" (path) instead of using the path as provided in the
>   URL. Particularly useful when wanting to issue HTTP requests without leading slash or other data
>   that doesn't follow the regular URL pattern, like "OPTIONS *".
>
>   Added in 7.55.0.

-X, --request <command>
>   (HTTP) Specifies a custom request method to use when communicating with the HTTP server.
>   The specified request method 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.
>
>   The method string you set with *-X, --request* will be used for all requests, which if you for exam-
>   ple use *-L, --location* may cause unintended side-effects when curl doesn't change request method
>   according to the HTTP 30x response codes - and similar.
>
>   (FTP) Specifies a custom FTP command to use instead of LIST when doing file lists with FTP.
>
>   (POP3) Specifies a custom POP3 command to use instead of LIST or RETR. (Added in 7.26.0)
>
>   (IMAP) Specifies a custom IMAP command to use instead of LIST. (Added in 7.30.0)
>
>   (SMTP) Specifies a custom SMTP command to use instead of HELP or VRFY. (Added in 7.34.0)
>
>   If this option is used several times, the last one will be used.

--resolve <host:port:address[,address]...>
>   Provide a custom address for a specific host and port pair. Using this, you can make the curl re-
>   quests(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.
>
>   By specifying '*' as host you can tell curl to resolve any host and specific port pair to the specified
>   address. Wildcard is resolved last so any *--resolve* with a specific host and port will be used first.
>
>   The provided address set by this option will be used even if *-4, --ipv4* or *-6, --ipv6* is set to make
>   curl use another IP version.
>
>   Support for providing the IP address within [brackets] was added in 7.57.0.
>
>   Support for providing multiple IP addresses per entry was added in 7.59.0.
>
>   Support for resolving with wildcard was added in 7.64.0.
>
>   This option can be used many times to add many host names to resolve.

Added in 7.21.3.

--retry-connrefused

> In addition to the other conditions, consider ECONNREFUSED as a transient error too for *--retry*. This option is used together with --retry.

Added in 7.52.0.

--retry-delay <seconds>

> Make curl sleep this amount of time before 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.
>
> If this option is used several times, the last one will be used.

Added in 7.12.3.

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

Added in 7.12.3.

--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 4xx response code or an HTTP 408 or 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.
>
> Since curl 7.66.0, curl will comply with the Retry-After: response header if one was present to know when to issue the next retry.
>
> If this option is used several times, the last one will be used.

Added in 7.12.3.

--sasl-authzid

> Use this authorisation identity (authzid), during SASL PLAIN authentication, in addition to the authentication identity (authcid) as specified by *-u, --user*.
>
> If the option isn't specified, the server will derive the authzid from the authcid, but if specified, and depending on the server implementation, it may be used to access another user's inbox, that the user has been granted access to, or a shared mailbox for example.

Added in 7.66.0.

--sasl-ir
        Enable initial response in SASL authentication.

        Added in 7.31.0.

--service-name <name>
        This option allows you to change the service name for SPNEGO.

        Examples: *--negotiate --service-name* sockd would use sockd/server-name.

        Added in 7.43.0.

-S, --show-error
        When used with *-s, --silent*, it makes curl show an error message if it fails.

-s, --silent
        Silent or quiet 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.

        Use *-S, --show-error* in addition to this option to disable progress meter but still show error mes-
        sages.

        See also *-v, --verbose* and *--stderr*.

--socks4 <host[:port]>
        Use the specified SOCKS4 proxy. 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 socks4 proxy with *-x, --proxy* using
        a socks4:// protocol prefix.

        Since 7.52.0, *--preproxy* can be used to specify a SOCKS proxy at the same time *-x, --proxy* is
        used with an HTTP/HTTPS proxy. In such a case curl first connects to the SOCKS proxy and then
        connects (through SOCKS) to the HTTP or HTTPS proxy.

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

        Added in 7.15.2.

--socks4a <host[:port]>
        Use the specified SOCKS4a proxy. 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 socks4a proxy with *-x, --proxy* us-
        ing a socks4a:// protocol prefix.

        Since 7.52.0, *--preproxy* can be used to specify a SOCKS proxy at the same time *-x, --proxy* is
        used with an HTTP/HTTPS proxy. In such a case curl first connects to the SOCKS proxy and then
        connects (through SOCKS) to the HTTP or HTTPS proxy.

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

        Added in 7.18.0.

--socks5-basic

      Tells curl to use username/password authentication when connecting to a SOCKS5 proxy. The username/password authentication is enabled by default. Use *--socks5-gssapi* to force GSS-API authentication to SOCKS5 proxies.

      Added in 7.55.0.

--socks5-gssapi-nec

      As part of the GSS-API 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.

--socks5-gssapi-service <name>

      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.

      Added in 7.19.4.

--socks5-gssapi

      Tells curl to use GSS-API authentication when connecting to a SOCKS5 proxy. The GSS-API authentication is enabled by default (if curl is compiled with GSS-API support). Use *--socks5-basic* to force username/password authentication to SOCKS5 proxies.

      Added in 7.55.0.

--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.

      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.

      Since 7.52.0, *--preproxy* can be used to specify a SOCKS proxy at the same time *-x, --proxy* is used with an HTTP/HTTPS proxy. In such a case curl first connects to the SOCKS proxy and then connects (through SOCKS) to the HTTP or HTTPS proxy.

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

      Added in 7.18.0.

--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.

Since 7.52.0, *--preproxy* can be used to specify a SOCKS proxy at the same time *-x, --proxy* is used with an HTTP/HTTPS proxy. In such a case curl first connects to the SOCKS proxy and then connects (through SOCKS) to the HTTP or HTTPS proxy.

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

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

Added in 7.18.0.

-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, --speed-time* and is 30 if not set.
>
> If this option is used several times, the last one will be used.

-y, --speed-time <seconds>
> 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, --speed-limit*.
>
> 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.

--ssl-allow-beast
> 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 workarounds 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.

--ssl-no-revoke
> (Schannel) This option tells curl to disable certificate revocation checks. WARNING: this option loosens the SSL security, and by using this flag you ask for exactly that.
>
> Added in 7.44.0.

--ssl-reqd
> (FTP IMAP POP3 SMTP) Require SSL/TLS for the connection. Terminates the connection if the server doesn't support SSL/TLS.
>
> This option was formerly known as --ftp-ssl-reqd.
>
> Added in 7.20.0.

--ssl   (FTP IMAP POP3 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.

> 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.
>
> Added in 7.20.0.

-2, --sslv2
>   (SSL) Forces curl to use SSL version 2 when negotiating with a remote SSL server. Sometimes
>   curl is built without SSLv2 support. SSLv2 is widely considered insecure (see RFC 6176).
>
>   See also *--http1.1* and *--http2*. *-2, --sslv2* requires that the underlying libcurl was built to support
>   TLS. This option overrides *-3, --sslv3* and *-1, --tlsv1* and *--tlsv1.1* and *--tlsv1.2*.

-3, --sslv3
>   (SSL) Forces curl to use SSL version 3 when negotiating with a remote SSL server. Sometimes
>   curl is built without SSLv3 support. SSLv3 is widely considered insecure (see RFC 7568).
>
>   See also *--http1.1* and *--http2*. *-3, --sslv3* requires that the underlying libcurl was built to support
>   TLS. This option overrides *-2, --sslv2* and *-1, --tlsv1* and *--tlsv1.1* and *--tlsv1.2*.

--stderr 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.
>
>   See also *-v, --verbose* and *-s, --silent*.

--styled-output
>   Enables the automatic use of bold font styles when writing HTTP headers to the terminal. Use
>   --no-styled-output to switch them off.
>
>   Added in 7.61.0.

--suppress-connect-headers
>   When *-p, --proxytunnel* is used and a CONNECT request is made don't output proxy CONNECT
>   response headers. This option is meant to be used with *-D, --dump-header* or *-i, --include* which
>   are used to show protocol headers in the output. It has no effect on debug options such as *-v, --verbose* or *--trace*, or any statistics.
>
>   See also *-D, --dump-header* and *-i, --include* and *-p, --proxytunnel*.

--tcp-fastopen
>   Enable use of TCP Fast Open (RFC7413).
>
>   Added in 7.49.0.

--tcp-nodelay
>   Turn on the TCP_NODELAY option. See the *curl_easy_setopt(3)* man page for details about this
>   option.
>
>   Since 7.50.2, curl sets this option by default and you need to explicitly switch it off if you don't
>   want it on.
>
>   Added in 7.11.2.

-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.

--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.

--tftp-no-options
:   (TFTP) Tells curl not to send TFTP options requests.

    This option improves interop with some legacy servers that do not acknowledge or properly implement TFTP options. When this option is used *--tftp-blksize* is ignored.

    Added in 7.48.0.

-z, --time-cond <time>
:   (HTTP FTP) 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(3)* 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.

--tls-max <VERSION>
:   (SSL) VERSION defines maximum supported TLS version. The minimum acceptable version is set by tlsv1.0, tlsv1.1, tlsv1.2 or tlsv1.3.

    default   Use up to recommended TLS version.

    1.0   Use up to TLSv1.0.

    1.1   Use up to TLSv1.1.

    1.2   Use up to TLSv1.2.

    1.3   Use up to TLSv1.3.

See also *--tlsv1.0* and *--tlsv1.1* and *--tlsv1.2* and *--tlsv1.3*. *--tls-max* requires that the underlying libcurl was built to support TLS. Added in 7.54.0.

--tls13-ciphers <list of TLS 1.3 ciphersuites>
:   (TLS) Specifies which cipher suites to use in the connection if it negotiates TLS 1.3. The list of ciphers suites must specify valid ciphers. Read up on TLS 1.3 cipher suite details on this URL:

     https://curl.haxx.se/docs/ssl-ciphers.html

    This option is currently used only when curl is built to use OpenSSL 1.1.1 or later. If you are using a different SSL backend you can try setting TLS 1.3 cipher suites by using the *--ciphers* option.

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

--tlsauthtype <type>
:   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". This option works only if the underlying libcurl is built with TLS-SRP support, which

requires OpenSSL or GnuTLS with TLS-SRP support.

Added in 7.21.4.

--tlspassword

Set password for use with the TLS authentication method specified with *--tlsauthtype*. Requires that *--tlsuser* also be set.

This doesn't work with TLS 1.3.

Added in 7.21.4.

--tlsuser <name>

Set username for use with the TLS authentication method specified with *--tlsauthtype*. Requires that *--tlspassword* also is set.

This doesn't work with TLS 1.3.

Added in 7.21.4.

--tlsv1.0

(TLS) Forces curl to use TLS version 1.0 or later when connecting to a remote TLS server.

In old versions of curl this option was documented to allow _only_ TLS 1.0, but behavior was inconsistent depending on the TLS library. Use *--tls-max* if you want to set a maximum TLS version.

Added in 7.34.0.

--tlsv1.1

(TLS) Forces curl to use TLS version 1.1 or later when connecting to a remote TLS server.

In old versions of curl this option was documented to allow _only_ TLS 1.1, but behavior was inconsistent depending on the TLS library. Use *--tls-max* if you want to set a maximum TLS version.

Added in 7.34.0.

--tlsv1.2

(TLS) Forces curl to use TLS version 1.2 or later when connecting to a remote TLS server.

In old versions of curl this option was documented to allow _only_ TLS 1.2, but behavior was inconsistent depending on the TLS library. Use *--tls-max* if you want to set a maximum TLS version.

Added in 7.34.0.

--tlsv1.3

(TLS) Forces curl to use TLS version 1.3 or later when connecting to a remote TLS server.

Note that TLS 1.3 is only supported by a subset of TLS backends. At the time of this writing, they are BoringSSL, NSS, and Secure Transport (on iOS 11 or later, and macOS 10.13 or later).

Added in 7.52.0.

-1, --tlsv1

(SSL) Tells curl to use at least TLS version 1.x when negotiating with a remote TLS server. That means TLS version 1.0 or higher

See also *--http1.1* and *--http2*. *-1, --tlsv1* requires that the underlying libcurl was built to support TLS. This option overrides *--tlsv1.1* and *--tlsv1.2* and *--tlsv1.3*.

--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.

--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.
>
> If this option is used several times, the last one will be used.
>
> This option overrides *--trace* and *-v, --verbose*.

--trace-time
> Prepends a time stamp to each trace or verbose line that curl displays.
>
> Added in 7.14.0.

--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. Use "%" as filename to have the output sent to stderr.
>
> If this option is used several times, the last one will be used.
>
> This option overrides *-v, --verbose* and *--trace-ascii*.

--unix-socket <path>
> (HTTP) Connect through this Unix domain socket, instead of using the network.
>
> Added in 7.40.0.

-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 an 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, --upload-file* for each URL on the command line. Each *-T, --upload-file* + URL pair specifies what to upload and to where. curl also supports "globbing" of the *-T, --upload-file* 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 --upload-file "{file1,file2}" http://www.example.com
>
> or even
>
> curl -T "img[1-1000].png" ftp://ftp.example.com/upload/

When uploading to an SMTP server: the uploaded data is assumed to be RFC 5322 formatted. It has to feature the necessary set of headers and mail body formatted correctly by the user as curl will not transcode nor encode it further in any way.

--url <url>

Specify a URL to fetch. This option is mostly handy when you want to specify URL(s) in a config file.

If the given URL is missing a scheme name (such as "http://" or "ftp://" etc) then curl will make a guess based on the host. If the outermost sub-domain name matches DICT, FTP, IMAP, LDAP, POP3 or SMTP then that protocol will be used, otherwise HTTP will be used. Since 7.45.0 guessing can be disabled by setting a default protocol, see *--proto-default* for details.

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.

-B, --use-ascii

(FTP LDAP) Enable ASCII transfer. For FTP, this can also be enforced by using a URL that ends with ";type=A". This option causes data sent to stdout to be in text mode for win32 systems.

-A, --user-agent <name>

(HTTP) Specify the User-Agent string to send to the HTTP server. To encode blanks in the string, surround the string with single quote marks. This header can also be set with the *-H, --header* or the *--proxy-header* options.

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

-u, --user <user:password>

Specify the user name and password to use for server authentication. Overrides *-n, --netrc* and *--netrc-optional*.

If you simply specify the user name, curl will prompt for a password.

The user name and passwords are split up on the first colon, which makes it impossible to use a colon in the user name with this option. The password can, still.

On systems where it works, curl will hide the given option argument from process listings. This is not enough to protect credentials from possibly getting seen by other users on the same system as they will still be visible for a brief moment before cleared. Such sensitive data should be retrieved from a file instead or similar and never used in clear text in a command line.

When using Kerberos V5 with a Windows based server you should include the Windows domain name in the user name, in order for the server to successfully obtain a Kerberos Ticket. If you don't then the initial authentication handshake may fail.

When using NTLM, the user name can be specified simply as the user name, without the domain, if there is a single domain and forest in your setup for example.

To specify the domain name use either Down-Level Logon Name or UPN (User Principal Name) formats. For example, EXAMPLE\user and user@example.com respectively.

If you use a Windows SSPI-enabled curl binary and perform Kerberos V5, Negotiate, NTLM or Digest authentication then you can tell curl to select 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.

-v, --verbose

      Makes curl verbose during the operation. Useful for debugging and seeing what's going on "under the hood". A line starting with '>' means "header data" sent by curl, '<' means "header data" received by curl that is hidden in normal cases, and a line starting with '*' means additional info provided by curl.

      If you only want HTTP headers in the output, *-i, --include* might be the option you're looking for.

      If you think this option still doesn't give you enough details, consider using *--trace* or *--trace-ascii* instead.

      Use *-s, --silent* to make curl really quiet.

      See also *-i, --include*. This option overrides *--trace* and *--trace-ascii*.

-V, --version

      Displays information about curl and the libcurl version it uses.

      The first line includes the full version of curl, libcurl and other 3rd party libraries linked with the executable.

      The second line (starts with "Protocols:") shows all protocols that libcurl reports to support.

      The third line (starts with "Features:") shows specific features libcurl reports to offer. Available features include:

IPv6   You can use IPv6 with this.

krb4   Krb4 for FTP is supported.

SSL   SSL versions of various protocols are supported, such as HTTPS, FTPS, POP3S and so on.

libz   Automatic decompression of compressed files over HTTP is supported.

NTLM   NTLM authentication is supported.

Debug   This curl uses a libcurl built with Debug. This enables more error-tracking and memory debugging etc. For curl-developers only!

AsynchDNS

      This curl uses asynchronous name resolves. Asynchronous name resolves can be done using either the c-ares or the threaded resolver backends.

SPNEGO

      SPNEGO authentication is supported.

Largefile

      This curl supports transfers of large files, files larger than 2GB.

IDN   This curl supports IDN - international domain names.

GSS-API

      GSS-API is supported.

SSPI   SSPI is supported.

TLS-SRP

      SRP (Secure Remote Password) authentication is supported for TLS.

HTTP2   HTTP/2 support has been built-in.

UnixSockets
> Unix sockets support is provided.

HTTPS-proxy
> This curl is built to support HTTPS proxy.

Metalink
> This curl supports Metalink (both version 3 and 4 (RFC 5854)), which describes mirrors and hashes. curl will use mirrors for failover if there are errors (such as the file or server not being available).

PSL   PSL is short for Public Suffix List and means that this curl has been built with knowledge about "public suffixes".

MultiSSL
> This curl supports multiple TLS backends.

-w, --write-out <format>
> Make curl display information on stdout after a completed transfer. The format is a string that may contain plain text mixed with any number of variables. The format can be specified as a literal "string", or you can have curl read the format from a file with "@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 as %{variable_name} and to output a normal % you just write them as %%. You can output a newline by using \n, a carriage return with \r and a tab space with \t.
>
> The output will be written to standard output, but this can be switched to standard error by using %{stderr}.
>
> **NOTE:** The %-symbol is a special symbol in the win32-environment, where all occurrences of % must be doubled when using this option.
>
> The variables available are:

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 *-O, --remote-name* or *-o, --output* option. It's most useful in combination with the *-J, --remote-header-name* option. (Added in 7.26.0)

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)

http_version   The http version that was effectively used. (Added in 7.50.0)

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)

proxy_ssl_verify_result
The result of the HTTPS proxy's SSL peer certificate verification that was requested. 0 means the verification was successful. (Added in 7.52.0)

redirect_url   When an HTTP request was made without *-L, --location* to follow redirects (or when --max-redir is met), this variable will show the actual URL a redirect *would* have gone 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)

scheme   The URL scheme (sometimes called protocol) that was effectively used (Added in 7.52.0)

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)

stderr   From this point on, the *-w, --write-out* output will be written to standard error. (Added in 7.63.0)

stdout   From this point on, the *-w, --write-out* output will be written to standard output. This is the default, but can be used to switch back after switching to stderr. (Added in 7.63.0)

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 including 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.

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

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.

FILES

˜/.curlrc

> Default config file, see *-K, --config* for details.

ENVIRONMENT

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 *-x, --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/domains>

> list of host names that shouldn't go through any proxy. If set to an asterisk '*' only, it matches all hosts. Each name in this list is matched as either a domain name which contains the hostname, or the hostname itself.

> This environment variable disables use of the proxy even when specified with the *-x, --proxy* option. That is **NO_PROXY=direct.example.com curl -x http://proxy.example.com http://direct.example.com** accesses the target URL directly, and **NO_PROXY=direct.example.com curl -x http://proxy.example.com http://somewhere.example.com** accesses the target URL through the proxy.

> The list of host names can also be include numerical IP addresses, and IPv6 versions should then be given without enclosing brackets.

PROXY PROTOCOL PREFIXES

Since curl version 7.21.7, the proxy string may be specified with a protocol:// prefix to specify alternative proxy protocols.

If no protocol is specified in the proxy string or if the string doesn't match a supported one, the proxy will be treated as an HTTP proxy.

The supported proxy protocol prefixes are as follows:

http://   Makes it use it as an HTTP proxy. The default if no scheme prefix is used.

https://   Makes it treated as an **HTTPS** proxy.

socks4://
            Makes it the equivalent of *--socks4*

socks4a://
            Makes it the equivalent of *--socks4a*

socks5://
            Makes it the equivalent of *--socks5*

socks5h://
            Makes it the equivalent of *--socks5-hostname*

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   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.

10   FTP accept failed. While waiting for the server to connect back when an active FTP session is used, an error code was sent over the control connection or similar.

11   FTP weird PASS reply. Curl couldn't parse the reply sent to the PASS request.

12   During an active FTP session while waiting for the server to connect back to curl, the timeout expired.

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.

16   HTTP/2 error. A problem was detected in the HTTP2 framing layer. This is somewhat generic and can be one out of several problems, see the error message for details.

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   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
89	No connection available, the session will be queued
90	SSL public key does not matched pinned public key
91	Invalid SSL certificate status.
92	Stream error in HTTP/2 framing layer.
XX	More error codes will appear here in future releases. The existing ones are meant to never change.

AUTHORS / CONTRIBUTORS

Daniel Stenberg is the main author, but the whole list of contributors is found in the separate THANKS file.

WWW

https://curl.haxx.se

SEE ALSO

ftp(1), **wget**(1)