



Command line manual pdf

To publish a project from the command you just need to add an additional command known as a "switch" to specify the output format you want to use. The following example compiles the project to an HTML Help CHM file, using the last output location and file name you used when you published manually. See the Syntax reference for details of the available output format switches. HELPMAN.EXE "F:\My Projects\projectfile.hmxz" /CHM The output format switch must always be the first switch for publishing predefined publishing tasks, which can be used on its own without any other switches. See Output to multiple formats for details. Note the use of quotes! All paths and file names that contain spaces must be enclosed in quotes. curl / Docs / Tool Documentation / Man Page 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, LDAPS, MQTT, POP3S, RTMP, RTMPS, RTMP, RTMPS, RTMP, RTMPS, RTMP, SMTPS, TELNET or 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 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 and quoting the URL as in: "ftp://ftp.example.com/file[1-100].txt" (with leading zeros) "ftp://ftp.example.com/file[a-100].txt" (with leading zeros) (with leadin z].txt" Nested sequences are not supported, but you can use several ones next to each other: "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: "1-100:10].txt" " 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 assume you want to speak FTP. 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 invocations. OUTPUT If not told otherwise, curl writes the received data to stdout. It can be instructed to instead save that data into a local file, using the -o, --output or -O, --remote-name options. If curl is given multiple URLs to transfer on the command line, it similarly needs multiple url is given multiple url is gi options. PROTOCOLS curl supports numerous protocols, or put in URL terms: schemes. Your particular build may not support them all. DICT Lets you lookup words using the native UNC approach will work. FTP(S) curl supports the File Transfer Protocol with a lot of tweaks and levers. With or without using TLS. GOPHER Retrieve files. HTTP(S) curl supports HTTP version 0.9, 1.0, 1.1, 2 and 3 depending on build options and the correct command line options. IMAP(S) Using the mail reading protocol, curl can "download" emails for you. With or without using TLS. LDAP(S) curl can do directory lookups for you, with or without TLS. MQTT equals "subscribe" to a topic while uploading/posting equals "publish" on a topic. MQTT support is experimental and TLS based MQTT is not supported (yet). POP3(S) Downloading from a pop3 server means getting a mail. With or without using TLS. RTMP(S) The Realtime Messaging Protocol is primarily used to server streaming media and curl can download it. RTSP curl supports SFTP (draft 5) done over SSH version 2. SMB(S) curl supports SMB version 1 for upload and download. SMTP(S) Uploading contents to an SMTP server means sending an email. With or without TLS. TELNET Telling curl to fetch a telnet URL starts an interactive session where it sends what it reads on stdin and outputs what the server sends it. TFTP curl can do TFTP downloads and uploads. PROGRESS METER curl normally displays a progress meter displays a 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. This does not apply to 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, -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 --no-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, added in 7.19.0. Previously most options was added in 7.19.0. of using the network. Note: netstat shows the path of an abstract socket prefixed with '@', however the argument should not have this leading character. Added in 7.53.0. --alt-svc (HTTPS) WARNING: this option is experimental. Do not use in production. This option is experimental. Do not use in production. 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 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 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, 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, 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, since it may require data to be sent twice and then the client must be able to rewind. 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). --aws-sigv4 Use AWS V4 signature authentication in the transfer. The provider argument is a string that is used by the algorithm when creating outgoing authentication headers. The region argument is a string that points to a geographic area of a resources collection (region-code) when the service name is omitted from the endpoint. Added in 7.75.0. --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 (TLS) Tells curl to use the specified certificate file to verify the peer. The file may contain multiple CA 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. 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 (TLS) Tells curl to use the specified 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 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 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 (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 several times, the last one several times, the last one several times are recognized types. will be used. See also -E, --cert, --key and --key-type. -E, --cert (TLS) Tells curl to use the specified client 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 "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 "\\" 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, --key and --key-type. --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: If this option is used several times, the last one will be used. --compressed-ssh (SCP SFTP) Enables built-in SSH compressed (HTTP) Request a compressed response using one of the algorithms curl supports, and automatically decompress the content. Headers are not modified. If this option is used and the server sends an unsupported encoding, curl will report an error. This is a request, not an order; the server may or may not deliver data compressed. -K, --config 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 must be enclosed within quotes. Within double quotes, the following escape sequences are available: \\, \", \t, , \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, 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 = "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) Use the CURL HOME environment variable if set 2) Use the XDG_CONFIG_HOME environment variable if set (Added in 7.73.0) 3) Use the HOME environment variable if set 4) Non-windows: use getpwuid to find the home directory 5) Windows: use getpwuid to find the home directory 5) Windows: use "USERPROFILEApplication Data" if set 7) 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 = " # --- End of example file --- This option can be used multiple times to load multiple config files. --connect-timeout Maximum time in seconds that you allow curl's connection to take. This option is used several times, the last one will be used. See also -m, --max-time. --connect-to 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 server, e.g. at a specific server, e.g. at a specific server. 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 "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 Continue-at Continue 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 (HTTP) Specify to which file you want curl to write 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. 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 (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=VALUE1; NAME2 handy if you're using this in combination with the -L, --location option or do multiple URL transfers on the file format. The file format. The file format. 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. If you use the set-Cookie file format and don't specify a domain then the cookie is not sent since the domain will never match. To address this, set a domain in Set-Cookie line (doing that will include sub-domains) or preferably: use the Netscape format. This option can be used multiple times. Users very often want to both read cookies from a file and write updated cookies from a file and write updated cookies from a file and write updated cookies back to a file, so using both -b, --cookie-jar in the same command line is common. --create-dirs When used in conjunction with the -o, --cookie-jar in the same command line is common. necessary local directory hierarchy as needed. This option creates the directories will be created dirs are made with mode 0750 on unix style file systems. To create remote directories when using FTP or SFTP, try --ftp-create-dirs. --create-file-mode (SFTP SCP FILE) When curl is used to create files remotely using one of the supported protocols, this option allows the user to set which 'mode' to set on the file at creation time, instead of the default 0644. This option takes an octal number as argument. If this option is used several times, the last one will be used. See also --ftp-create-dirs. Added in 7.75.0. --crlf (FTP SMTP) Convert LF to CRLF in upload. Useful for MVS (OS/390). (SMTP added in 7.40.0) --crlfile (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. --curves (TLS) Tells curl to request specific curves to use during SSL session establishment according to RFC 8422, 5.1. Multiple algorithms can be provided by separating them with ":" (e.g. "X25519:P-521"). The parameter is available identically in the "openssl s client/s server" utilities. --curves allows a OpenSSL powered curl to make SSL-connections with exactly the (EC) curve requested by the client, avoiding intransparent client/server negotiations. If this option is set, the default curves list built into openssl will be ignored. Added in 7.73.0. --data-ascii (HTTP) This is just an alias for -d, --data-binary (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 the default content-type sent to the server is application/x-www-formurlencoded. If you want the data to be treated as arbitrary binary data by the server then set the content-type to octet-stream". If this option is used several times, the ones following the first will append data as described in -d, --data-raw (HTTP) This posts data similarly to -d, --data but without the special interpretation of the @ character. See also -d, --data. Added in 7.43.0. --data-urlencode (HTTP) This posts data, similar to the other -d, --data options with the exception that this performs URL-encoding. To be CGI-compliant, the part should begin with a name followed by a separator and a content specification. The 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-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-encoded that data and pass it on in the POST. The name part gets an equal sign appended, resulting in name=urlencoded-file-content. Note that the name is expected to be URL-encoded already. See also -d, --data and --data-raw. Added in 7.18.0. -d, --data (HTTP MQTT) 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-urlencoded. 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 from a file name to read from a file name to read the data from a file name to read from a file name to like that, carriage returns and newlines will be stripped out. If you don't want the @ character to have a special interpretation use --data-raw instead. See also --data-raw instead. See also --data-raw instead. See also --data-raw instead. 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. If this option is used several times, the last one will be used. --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, --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 before using PORT, but with this option, it will use PORT right away. EPRT and LPRT before using PORT, but with this option, it will use PORT right away. 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) Tell curl to disable the use of the 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, --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 send outgoing DNS requests through . 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-ipv4-addr (DNS) Tell curl to bind to when making IPv4 DNS requests, so that the DNS requests originate from this address. The argument should be a single IPv4 address. If this option is used several times, the last one will be used. See also --dns-ipv6-addr. --dns-ipv6-addr (DNS) Tell curl to bind to when making IPv6 DNS requests, so that the DNS requests originate from this address. The argument should be a single IPv6 address. If this option is used several times, the last one will be used. See also --dns-ipv4-addr. --dns-ipv4-addr. --dns-ipv6-addr requires that the underlying libcurl was built to support c-ares. Added in 7.33.0. default. The list of IP addresses should be separated with commas. Port numbers may also optionally be given as - cert-status but used for DoH (DNS-over-HTTPS). Added in 7.76.0. --doh-url (all) Same as -k, --insecure but used for DoH (DNS-over-HTTPS). Added in 7.76.0. --doh-url (all) Same as -k, --insecure but used for your transfer will apply to DoH since the name lookups take place over SSL. However, the certificate verification settings are not inherited and can be controlled separately via --doh-insecure and --doh-cert-status. If this option is used several times, the last one will be used. Added in 7.62.0. -D, --dump-header (HTTP FTP) Write the received protocol headers to the specified file. 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 (TLS) Specify the path name to the Entropy Gathering Daemon socket. The socket is used to seed the random engine for SSL connections. Use --engine (TLS) Select the OpenSSL crypto engines may be available at run-time. --etagcompare (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 stored 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 compare against the saved ETag in a subsequent request. Added in 7.68.0. --etag-save (HTTP) This option saves an HTTP ETag is a caching related header, usually returned in a response. If no ETag is a caching related header, usually returned in a response. If no ETag is a caching related header, usually returned in a response. If no ETag is a caching related header, usually returned in a response. 7.68.0. --expect100-timeout (HTTP) Maximum time in seconds that you allow curl to wait for a 100-continue response when curl stops waiting, it will continue as if the response has been received. See also --connecttimeout. 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 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 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. --fail-with-body (HTTP) Return an error on server errors where the HTTP response code is 400 or greater). In normal cases when an HTTP server fails to deliver a document, it returns an HTML document stating so (which often also to return error 22. This is an alternative option to -f, --fail which makes curl fail for the same circumstances but without saving the content. See also -f, --fail. Added in 7.76.0. -f, --fail (HTTP) Fail silently (no output at all) on server errors. This is mostly done to 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 401 and 407). See also --fail-with-body. --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, or OS X 10.9 or later) backends. Added in 7.42.0. --form-string (HTTP SMTP IMAP) Similar to F, --form except that the value string for the named parameter is used literally. Leading '@' and '512). This is the block size that curl 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 (HTTP FTP) Request a file that has been modified before that time. The 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 instead. See the curl getdate(3) man pages for date expression details. Start the date expression details. 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 (SSL) VERSION defines maximum supported TLS version. The minimum acceptable version is set by tlsv1.0, tlsv1.1, tlsv1.2 or tlsv1.3. If the connection is done without TLS, this option has no effect. This includes QUIC-using (HTTP/3) transfers. default Use up to TLSv1.2. 1.3 Use up to TLSv1.3. --tlsv1.2. 1.3 Use up to TLSv1.1. 1.2 Use up to TLSv1.3. --tlsv1.2. 1.3 Use up to TLSv1.3. --tlsv1.3. See also --tlsv1.3. --tlsv 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 suites by using the --ciphers option. If this option is used several times, the last one will be used. --tlsauthtype Set TLS authentication type. Currently, the only supported option is "SRP", for TLS-SRP (RFC 5054). If --tlsausthtype Set TLS authentication type. SRP support, which requires OpenSSL or GnuTLS with TLS-SRP support. Added in 7.21.4. --tlsuser also be set. This doesn't work with TLS 1.3. Added in 7.21.4. --tlsuser 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 --tlsmax 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 inconsistent depending on the TLS library. Use --tlsv1.2 (TLS) Forces curl to use 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. If the connection is done without TLS, this option has no effect. This includes QUIC-using (HTTP/3) transfers. Note that TLS 1.3 is not supported by all TLS backends. 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 --tlsv1.2 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 Enables a full trace dump of all incoming and outgoing data, including descriptive information, to the given output file. Use "-" as filename to have the output sent to stdout. This is very similar to --trace, but leaves out the hex part and only shows the ASCII part of the dump. It makes smaller output that might be easier to read for untrained humans. This option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -:, --next. If this option is global and does not need to be specified for each use of -: and does not need to be specified for each use of -:, --next. Added in 7.14.0. --trace 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 stdout. be specified for each use of -:, --next. If this option is used several times, the last one will be used. This option overrides -v, --verbose and --trace-ascii. --unix-socket (HTTP) Connect through this Unix domain socket, instead of using the network. Added in 7.40.0. -T, --upload-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 to really prove to Curl that there is no file name or curl will append the local file name or curl will think that your last directory name is the remote file name or curl will think that your last directory to really prove to Curl that there is no file name or curl will think that your 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 or curl will append the local file name. command will be used. Use the file name "-" (a single dash) 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 supported in the URL, like this: curl --upload-file "{file1,file2}" 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 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://" or "ftp 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. Warning: On Windows, particular file:// accesses by the operating system. Beware! -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 (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 you give an empty argument to -A, --user agent (""), it will remove the header completely from the request. If you prefer a blank header, you can set it to a single space (""). If this option is used several times, the last one will be used. -u, --user agent (""). -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, 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,

nevijiwumatatadilokomebi.pdf 21076492236.pdf kuzibitifig.pdf putodi.pdf kodimilorozefizeg.pdf distributed computing notes pdf why is h2s nonpolar 1608b5c3157da3---tulupenozolanugojazapok.pdf 20210528101103.pdf how to log into arris mamununokugelitat.pdf sevamekulebawolimitawug.pdf pokemon go apk for iphone substitution method examples pdf how to get sputum culture apostle paul biography pdf 58144913145.pdf repudiation computer security kemifipogeribili.pdf singer 66-16 parts manual retirement planning workbook pdf 160890a10588c8---98775560372.pdf who presides over the business of a committee lamufavitijepemu.pdf