A Rexx interface to the cURL library

Version 2.1.1

10 May 2024

Table of Contents

1. Introduction

This document describes the interface to the cURL package. cURL is a general purpose package that allows access to any URL-addressable resource. With Rexx/CURL you can access resources such as web pages, ftp sites, and telnet sessions under control of your Rexx program. Rexx/CURL is actually built on top of libcurl but for the purposes of this document it will be refered to simply as cURL. Rexx/CURL implements the libcurl *easy interface*.

2. Overview

Rexx/CURL consists of Rexx external functions that allows a Rexx program to access any URL. The basic concept of Rexx/CURL (and cURL), is that you specify a URL and all options appropriate to that URL, and then perform the URL access.
See Using Rexx/CURL for more details.

The Rexx/CURL external functions are:

- **CURLINIT** - initialise the cURL interface
- **CURLCLEANUP** - cleanup the cURL interface
- **CURLRESET** - resets all options on the cURL handle
- **CURLESCAPE** - converts all special characters in a string, usually a URL, to escaped strings based on RFC 2396
- **CURLUNESCAPE** - converts all special characters in a string, usually a URL, to their normal representation
- **CURLSETOPT** - defines all the options appropriate to the type of URL you wish to access
- **CURLFORMADD** - add a section to a multipart/formdata HTTP POST
- **CURLFORMFREE** - free up resources used by CURLFORMADD
- **CURLPERFORM** - carries out the actions defined by the various calls to CURLSETOPT
- **CURLGETINFO** - return information about the action carried out by CURLPERFORM
- **CURLVARIABLE** - set or retrieve default run-time values

- **CURLLOADFUNCS** - load all Rexx/CURL external functions
- **CURLDROPFUNCS** - unload all Rexx/CURL external functions

Rexx/CURL also provides a number of *constants* that provide useful information for the user. Each of these *constants* is generally prefixed by a stem name (default is !REXXCURL.!) to make it easier to access when inside a procedure. Remember you need to EXPOSE the stem to get access to these *constants* inside a procedure.
The *constants* are:

- **DIRSEP** - the operating system default seperator for directory names
- **PATHSEP** - the operating system default seperator for directories in PATH

eg. under Linux:

```
!REXXCURL.!DIRSEP -> '/'
!REXXCURL.!PATHSEP -> ':'
```

In addition to the above environmental *constants* a number of *constants* that provide information about the capabilities of the loaded cURL library are available. They are:

- **VERSION** - a string for the libcurl versionfor dir
- **VERSION_NUM** - a 24 bit number created like this: | | . Version 7.9.8 is therefore returned as 0x070908.
- **HOST** - a string showing what host information that this libcurl was built for. As discovered by a configure script or set by the build environment.
- **SSL_VERSION** - a string for the OpenSSL version used. If libcurl has no SSL support, this is blank.
- **LIBZ_VERSION** - a string for the version of libz used. If libcurl has no libz support, this is blank.
- **ARES** - a string for the version of ares used. If libcurl has no ares support, this is blank.
- **ARES_NUM** - a number representing the ares version used. If libcurl has no ares support, this is 0.
- **LIBIDN** - a string for the version of libidn used. If libcurl has no libidn support, this is blank.
- **LIBSSH_VERSION** - a string for the version of libssh used. If libcurl has no libssh support, this is blank.
- **ICONV_VER_NUM** - a number representing the iconv version used. If libcurl has no iconv support, this is 0.
- **SUPPORTS_IPV6** - supports IPv6
- **SUPPORTS_KERBEROS4** - supports kerberos4 (when using FTP)
- **SUPPORTS_SSL** - supports SSL (HTTPS/FTPS)
- **SUPPORTS_LIBZ** - supports HTTP deflate using libz
- **SUPPORTS_NTLM** - supports HTTP NTLM
- **SUPPORTS_GSSNEGOTIATE** - supports HTTP GSS-Negotiate
- **SUPPORTS_DEBUG** - libcurl was built with debug capabilities
- **SUPPORTS_CURLDEBUG** - libcurl was built with memory tracking debug capabilities. This is mainly of interest for libcurl hackers
- **SUPPORTS_ASYNCHDNS** - libcurl was built with support for asynchronous name lookups, which allows more exact timeouts (even on Windows)
- **SUPPORTS_SPEGNO** - libcurl was built with support for SPNEGO authentication (Simple and Protected GSS-API Negotiation Mechanism, defined in RFC 2478.)
- **SUPPORTS_LARGEFILE** - libcurl was built with support for large files.
- **SUPPORTS_IDN** - libcurl was built with support for IDNA, domain names with international letters
- **SUPPORTS_SSPI** - libcurl was built with support for SSPI. This is only available on Windows and makes libcurl use Windows-provided functions for NTLM authentication. It also allows libcurl to use the current user and the current user's password without the app having to pass them on
- **SUPPORTS_CONV** - libcurl was built with support for character conversions, as provided by the CURLOPT_CONV_* callbacks. N/A to Rexx/CURL
- **SUPPORTS_TLSAUTH_SRP** - libcurl was built with support for TLS-SRP
- **SUPPORTS_NTLM_WB** - libcurl was built with support for NTLM delegation to a winbind helper
- **PROTOCOLS** - an *array* (and a string) containing the names of protocols that libcurl supports (using lowercase letters). The protocol names are the same as would be used in URLs.

eg. under Linux:

```
!REXXCURL.!VERSION -> '7.19.7'
!REXXCURL.!VERSION_NUM -> '463623'
!REXXCURL.!HOST -> 'i486-pc-linux-gnu'
!REXXCURL.!SSL_VERSION -> 'OpenSSL/0.9.8k'
!REXXCURL.!LIBZ_VERSION -> '1.2.3.3'
!REXXCURL.!ARES -> ''
!REXXCURL.!ARES_NUM -> '0'
!REXXCURL.!LIBIDN -> '1.15'
!REXXCURL.!LIBSSH_VERSION -> ''
!REXXCURL.!ICONV_VER_NUM -> '0'
!REXXCURL.!SUPPORTS_IPV6 -> '1'
!REXXCURL.!SUPPORTS_KERBEROS4 -> '0'
```

```
        !REXXCURL.!SUPPORTS_SSL -> '1'
        !REXXCURL.!SUPPORTS_LIBZ -> '1'
        !REXXCURL.!SUPPORTS_NTLM -> '1'
        !REXXCURL.!SUPPORTS_GSSNEGOTIATE -> '1'
        !REXXCURL.!SUPPORTS_DEBUG -> '0'
        !REXXCURL.!SUPPORTS_CURLDEBUG -> '0'
        !REXXCURL.!SUPPORTS_ASYNCHDNS -> '0'
        !REXXCURL.!SUPPORTS_SPNEGO -> '0'
        !REXXCURL.!SUPPORTS_LARGEFILE -> '1'
        !REXXCURL.!SUPPORTS_IDN -> '1'
        !REXXCURL.!SUPPORTS_SSPI -> '0'
        !REXXCURL.!SUPPORTS_CONV -> '0'
        !REXXCURL.!SUPPORTS_TLSAUTH_SRP -> '0'
        !REXXCURL.!SUPPORTS_NTLM_WB -> '0'
        !REXXCURL.!PROTOCOLS.0-> '10'
        !REXXCURL.!PROTOCOLS.1 -> 'tftp'
        !REXXCURL.!PROTOCOLS.2 -> 'ftp'
        !REXXCURL.!PROTOCOLS.3 -> 'telnet'
        !REXXCURL.!PROTOCOLS.4 -> 'dict'
        !REXXCURL.!PROTOCOLS.5 -> 'ldap'
        !REXXCURL.!PROTOCOLS.6 -> 'ldaps'
        !REXXCURL.!PROTOCOLS.7 -> 'http'
        !REXXCURL.!PROTOCOLS.8 -> 'file'
        !REXXCURL.!PROTOCOLS.9 -> 'https'
        !REXXCURL.!PROTOCOLS.10 -> 'ftps'
        !REXXCURL.!PROTOCOLS -> 'tftp ftp telnet dict ldap ldaps http file https ftps'
```

The default constant prefix can be changed by calling CURLVARIABLE with the **CONSTANTPREFIX** variable. eg.

```
        Call CURLVariable 'CONSTANTPREFIX', '?MYCURL.'
```

3. Functions

This section provides the full syntax and usage of each function that comprises Rexx/CURL.

CURLINIT()

Initialises the cURL interface. The return value from this call is used as the first argument to most of the other Rexx/CURL functions.

Arguments:

none

Returns:

success:
any non-blank value
failure:
blank

CURLCLEANUP()

This function must be the last function called for a session. It is the opposite of the CURLINIT() function and must be called with the same handle as input that the CURLINIT() call returned. This will effectively close all connections this handle has used and possibly has kept open until now. Don't call this function if you intend to transfer more files. Any uses of the handle after this function has been called are illegal. This kills the handle and all memory associated with it!

Arguments:

handle
The value returned from CURLINIT.

Returns:

success:
any non-blank value
failure:
blank

CURLRESET()

Re-initializes all options previously set on a specified CURL handle to the default values. This puts back the handle to the same state as it was in when it was just created with CURLINIT(). It does not change the following information kept in the handle: live connections, the Session ID cache, the DNS cache, the cookies and shares.

Arguments:

handle
The value returned from CURLINIT.

Returns:

N/A

CURLESCAPE(*handle*, *URL*)

This function converts the given input string to an URL encoded string and returns it. All input characters that are not a-z, A-Z, 0-9, '-', '.', '_' or '~' are converted to their "URL escaped" version (%NN where NN is a two-digit hexadecimal number).

Arguments:

handle
The value returned from CURLINIT.
URL
This is the string (usually a URL) to be converted

Returns:

success:
The converted string
failure:
blank
On *failure* **CURLERROR.INTCODE** is set to a non-zero value.

CURLUNESCAPE(*handle*, *URL*)

This function converts the given URL encoded input string to a "plain string" and returns it. All input characters that are URL encoded (%XX where XX is a two-digit hexadecimal number) are converted to their binary versions.

Arguments:

handle
The value returned from CURLINIT.
URL
This is the string (usually a URL) to be converted

Returns:

success:
The converted string
failure:
blank
On *failure* **CURLERROR.INTCODE** is set to a non-zero value.

CURLSETOPT(*handle*, *option*,*option value*[,*more option values*,...])

This function is called to define all the parameters and data required to carry out the particular request. The *option* argument is case-insensitive. The table below defining the options available contains the equivalent cURL option value used in the C/C++ interface. Not all of the Rexx/CURL option strings are the same as the C/C++ equivalents; I've tried to be more consistent and clearer with the name of the option. The C/C++ equivalents are there if you wish to read alternate definitions of these options. See the curl_easy_setopt() function documentation.
All options set with this function stay in effect until CURLCLEANUP is called or the option reset with another value.

Arguments:

handle
The value returned from CURLINIT.
option
This is the string identifying the option to set. Values with a grey background have been deprecated and should not be used.

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
ACCEPTENCODING	Sets the contents of the Accept-Encoding: header sent in an HTTP request, and enables decoding of a response when a Content-Encoding: header is received. ◊ **"IDENTITY"**: Does nothing ◊ **"DEFLATE"**: requests the server to compress its response using the zlib algorithm ◊ **"GZIP"**: requests the gzip algorithm If a zero-length string is set, then an Accept-Encoding: header containing all supported encodings is sent. This is a request, not an order; the server may or may not do it. This option must be set (to any non-empty value) or else any unsolicited encoding done by the server is ignored. See the special cURL file lib/README.encoding for details.	CURLOPT_ACCEPT_ENCODING
ACCEPTTIMEOUTMS	Not documented on cURL site.	CURLOPT_ACCEPT_TIMEOUTMS
ADDRESSSCOPE	A number sppecifying the scope_id value to use when connecting to IPv6 link-local or site-local addresses.	CURLOPT_ADDRESS_SCOPE
APPEND	Set this option to a **1** or **Y** to indicate that the remote file is to be appended to rather than replaced when an FTP upload is to be carried out.	CURLOPT_APPEND
BUFFERSIZE	Specify the size of the recieve buffer to use.	CURLOPT_BUFFERSIZE
CAINFO	The name of a file from which the SSL peer certificate data is read for the performed action. This file must exist.	CURLOPT_CAINFO
CAPATH	The CAPATH directory used to validate the peer certificate. This option is used only if SSL_VERIFYPEER is true.	CURLOPT_CAPATH
CERTINFO	Set this option to a **1** or **Y** to enable libcurl's certificate chain info gatherer. With this enabled, libcurl (if built with OpenSSL) will extract lots of information and data about the certificates in the certificate chain used in the SSL connection. This data is then possible to extract after a transfer using CURLGETINFO and its option **CERTINFO**	CURLOPT_CERTINFO
CLOSEPOLICY	Specify the type of disconnection policy to use if the connection cache is filled. This is only applicable if you potentially use more than 5 concurrent connections. Can be one of "**OLDEST**", or "**LEAST_RECENTLY_USED**".	CURLOPT_CLOSEPOLICY
CONNECTONLY	Set this option to a **1** or **Y** to tell the library to perform any required proxy authentication and connection setup, but no data transfer.	CURLOPT_CONNECT_ONLY
CONNECTTIMEOUT	To limit the time it takes to connect to the server, set this value to the number of seconds.	CURLOPT_CONNECTTIMEOUT
CONNECTTIMEOUTMS	Like CONNECTTIMEOUT but takes number of milliseconds instead.	CURLOPT_CONNECTTIMEOUT_MS
COOKIE	If you want to pass a cookie to the server, set	CURLOPT_COOKIE

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
	this option to the cookie. The format of the cookie is: *name=contents*, where *name* is the name of the cookie to be set.	
COOKIEFILE	The name of a file from which cookie data is read for the performed action. The cookie file can contain Netscape/Mozilla formated cookies, or regular HTTP header format. If a file name is supplied, this file must exist. The file name can be the empty string to start the cookie engine but not read any cookies.	CURLOPT_COOKIEFILE
COOKIEJAR	The name of a file to which all known cookies are written after the operation completes.	CURLOPT_COOKIEJAR
COOKIELIST	Pass a string containing a cookie string. Cookie can be either in Netscape / Mozilla format or just regular HTTP-style header (Set-Cookie: ...) format. If cURL cookie engine was not enabled it will enable its cookie engine. Additionally, there are commands available that perform actions if you pass in these exact strings: ◊ **"ALL"**: erases all cookies held in memory ◊ **"SESS"**: erases all session cookies held in memory ◊ **"FLUSH"**: writes all known cookies to the file specified by option **COOKIEJAR** ◊ **"RELOAD"**: loads all cookies from the files specified by option **COOKIEFILE**	CURLOPT_COOKIELIST
COOKIESESSION	Set this option to a **1** or **Y** to indicate that this operation is the start of a cookie session. Set this option to a **0** or **N** to indicate that this operation is the end of a cookie session.	CURLOPT_COOKIESESSION
CRLF	Set this option to a **1** or **Y** to indicate that LF charcaters should be converted to CRLF on transfers.	CURLOPT_CRLF
CRLFILE	Pass a string naming a file with the concatenation of CRL (in PEM format) to use in the certificate validation that occurs during the SSL exchange. When curl is built to use NSS or GnuTLS, there is no way to influence the use of CRL passed to help in the verification process. When libcurl is built with OpenSSL support, X509_V_FLAG_CRL_CHECK and X509_V_FLAG_CRL_CHECK_ALL are both set, requiring CRL check against all the elements of the certificate chain if a CRL file is passed. This option makes sense only when used in combination with the SSLVERIFYPEER option.	CURLOPT_CRLFILE
CUSTOMREQUEST	To carry out an HTTP request command other than GET or HEAD, pass the command in this option.	CURLOPT_CUSTOMREQUEST
DIRLISTONLY		CURLOPT_DIRLISTONLY

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
	Set this option to a **1** or **Y** to indicate that only filenames are to be returned when the **URL** option specifies an FTP directory. Normally, file sizes, dates etc. are returned. This works for FTP and SFTP URLs. This causes an FTP NLST command to be sent on an FTP server. Beware that some FTP servers list only files in their response to NLST; they might not include subdirectories and symbolic links. Setting this option to **1** or **Y** also implies a directory listing even if the URL doesn't end with a slash, which otherwise is necessary. Do NOT use this option if you also use the WILDCARDMATCH option as it will effectively break that feature.	
DNSCACHETIMEOUT	This sets the timeout in seconds. Name resolves will be kept in memory for this number of seconds. Set to zero to completely disable caching, or set to -1 to make the cached entries remain forever. By default, libcurl caches info for 60 seconds.	CURLOPT_DNS_CACHE_TIMEOUT
DNSSERVERS	Set the list of DNS servers to be used instead of the system default. The format of the dns servers option is: **host[:port][,host[:port]]...** For example: **192.168.1.100,192.168.1.101,3.4.5.6** This option requires that libcurl was built with a resolver backend that supports this operation. The c-ares backend is the only such one. This can be determined by testing for **!REXXCURL.!ARES** being non-blank or **!REXXCURL.!ARES_NUM** not being zero.	CURLOPT_DNS_SERVERS
DNSUSEGLOBALCACHE	Set this option to a **1** or **Y** to use a global DNS cahce that will last between operations.	CURLOPT_DNS_USE_GLOBAL_CACHE
EGDSOCKET	The name of the Entropy Gathering Socket which is used to seed the SSL random engine.	CURLOPT_EGDSOCKET
ENCODING Use:**ACCEPTENCODING**	Sets the contents of the Accept-Encoding: header sent in an HTTP request, and enables decoding of a response when a Content-Encoding: header is received. ◊ **"IDENTITY"**: Does nothing ◊ **"DEFLATE"**: requests the server to compress its response using the zlib algorithm ◊ **"GZIP"**: requests the gzip algorithm If a zero-length string is set, then an Accept-Encoding: header containing all supported encodings is sent. This is a request, not an order; the server may or may not do it. This option must be set (to any non-empty value) or else any unsolicited encoding done by the server is ignored. See the special cURL file lib/README.encoding for details.	CURLOPT_ENCODING
ERRFILE	The name of a file into which any error output from the performed action is written. By default, if this file exists, it will be	CURLOPT_STDERR

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
	over-written. You can pass **APPEND** as an extra argument to append data to an existing file.	
FAILONERROR	Set this option to a **1** or **Y** to get cURL to fail, rather than return the page, if the HTTP return code is greater than or equal to 300.	CURLOPT_FAILONERROR
FILETIME	Set this option to a **1** or **Y** to indicate that cURL should attempt to retrieve the modification date of the remote document. Not all servers will respond to this type of request. To retrieve the date, call CURLGETINFO with the **FILE_TIME** option.	CURLOPT_FILETIME
FOLLOWLOCATION	Set this option to a **1** or **Y** to get cURL to follow any *Location:* headers in the specified site.	CURLOPT_FOLLOWLOCATION
FORBIDREUSE	Set this option to a **1** or **Y** to get cURL to make the next transfer explicitly close the connection when done. Normally, libcurl keeps all connections alive when done with one transfer in case there comes a succeeding one that can re-use them. This option should be used with caution and only if you understand what it does. Set to 0 to have libcurl keep the connection open for possibly later re-use (default behavior).	CURLOPT_FORBID_REUSE
FRESHCONNECT	Set this option to a **1** or **Y** to indicate that the next transfer should use a new connection.	CURLOPT_FRESH_CONNECT
FTPALTERNATIVETOUSER	A string which will be used to authenticate if the usual FTP "USER user" and "PASS password" negotiation fails. This is currently only known to be required when connecting to Tumbleweed's Secure Transport FTPS server using client certificates for authentication.	CURLOPT_FTP_ALTERNATIVE_TO_USER
FTPAPPEND Use:**APPEND**	Set this option to a **1** or **Y** to indicate that the remote file is to be appended to rather than replaced when an FTP upload is to be carried out.	CURLOPT_FTPAPPEND
FTPCMDS	Specify a stem containing FTP commands to be issued before an FTP session is started.	CURLOPT_QUOTE
FTPCMDSAFTER	A list of FTP commands to be executed **after** the URL is accessed. The list of commands is specified as a stem name; ie the supplied string must end in a period, and represent a valid Rexx *array*.	CURLOPT_POSTQUOTE
FTPCMDSBEFORE	A list of FTP commands to be executed **before** the URL is accessed. The list of commands is specified as a stem name; ie the supplied string must end in a period, and represent a valid Rexx *array*.	CURLOPT_PREQUOTE
FTPCREATEMISSINGDIRS	Set this option to a **1** or **Y** to tell cURL to attempt to create any remote directory that it fails to CWD into. CWD is the command that changes working directory.	CURLOPT_FTP_CREATE_MISSING_DIRS
FTPCRLF	See the prefered CRLF option.	CURLOPT_CRLF
		CURLOPT_FTPLISTONLY

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
FTPLISTONLY Use:**DIRLISTONLY**	Set this option to a **1** or **Y** to indicate that only filenames are to be returned when the **URL** option specifies an FTP directory. Normally, file sizes, dates etc. are returned.	
FTPPORT	Set this option to a string to use as the parameter to the FTP *PORT* command. The parameter can be an IP address, a hostname, an interface name (undex Unix), or '-' to use the client machine's default IP address.	CURLOPT_FTPPORT
FTPRESPONSETIMEOUT	Causes uURL to set a timeout period (in seconds) on the amount of time that the server is allowed to take in order to generate a response message for a command before the session is considered hung. Note that while curl is waiting for a response, this value overrides "TIMEOUT". It is recommended that if used in conjunction with "TIMEOUT", you set "FTPRESPONSETIMEOUT" to a value smaller than "TIMEOUT".	CURLOPT_FTP_RESPONSE_TIMEOUT
FTPSKIPPASVIP	Set this option to a **1** or **Y** to instruct libcurl to not use the IP address the server suggests in its 227-response to libcurl's PASV command when libcurl connects the data connection. Instead libcurl will re-use the same IP address it already uses for the control connection. But it will use the port number from the 227-response. This option has no effect if PORT, EPRT or EPSV is used instead of PASV.	CURLOPT_FTP_SKIP_PASV_IP
FTPSSL Use:**USESSL**	Set this option to one of the following string values to make libcurl use your desired level of SSL for the ftp transfer. ◊ **"NONE"**: Don't attempt to use SSL. ◊ **"TRY"**: Try using SSL, proceed as normal otherwise. ◊ **"CONTROL"**: Require SSL for the control connection or fail with FTP_SSL_FAILED. ◊ **"ALL"**: Require SSL for all communication or fail with FTP_SSL_FAILED.	CURLOPT_FTP_SSL
FTPSSLAUTH	Set this option to one of the following string values to alter how libcurl issues AUTH TLS or AUTH SSL when FTP over SSL is activated. ◊ **"DEFAULT"**: Allow libcurl to decide. ◊ **"SSL"**: Try "AUTH SSL" first, and only if that fails try "AUTH TLS". ◊ **"TLS"**: Try "AUTH TLS" first, and only if that fails try "AUTH SSL".	CURLOPT_FTPSSLAUTH
FTPSSLCCC	If enabled, this option makes cURL use CCC (Clear Command Channel). It 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. Pass one of the values below. ◊ **"NONE"**: Don't attempt to use	CURLOPT_FTP_SSL_CCC

Rexx/CURL Option	Description	cURL Equivalent Option
	CCC. ◊ **"PASSIVE"**: Do not initiate the shutdown, but wait for the server to do it. Do not send a reply. ◊ **"ACTIVE"**: Initiate the shutdown and wait for a reply.	
FTPUSEEPSV	Set this option to a **1** or **Y** to tell cURL to use the EPSV command when doing passive FTP downloads (which it always does by default). Using EPSV means that it will first attempt to use EPSV before using PASV, but if you pass **0** or **N** to this option, it will not try using EPSV, only plain PASV.	CURLOPT_FTP_USE_EPSV
FTPUSEEPRET	Set this option to a **1** or **Y** to tell cURL to use the PRET (and LPRT) command when doing active FTP downloads (which is enabled by "FTPPORT"). Using EPRT means that it will first attempt to use EPRT and then LPRT before using PORT, but if you pass **0** to this option, it will not try using PRET or LPRT, only plain PORT.	CURLOPT_FTP_USE_PRET
GSSAPIDELEGATION	Set the extra parameter to one of the following: ◊ **"FLAG"**: Allow unconditional GSSAPI credential delegation. ◊ **"POLICY_FLAG"**: Delegate only if the OK-AS-DELEGATE flag is set in the service ticket in case this feature is supported by the GSSAPI implementation and the definition of GSS_C_DELEG_POLICY_FLAG was available at compile-time of libcurl. ◊ **"NONE"**: Disable delegation. This is the default setting.	CURLOPT_GSSAPI_DELEGATION
HEADER	Set this option to a **1** or **Y** to get cURL to return header information for those protocols, like HTTP, that have seperate headers and footers.	CURLOPT_HEADER
HEADERFILE	The name of a file into which headers from the performed action are written. By default, if this file exists, it will be over-written. You can pass **APPEND** as an extra argument to append data to an existing file.	CURLOPT_WRITEHEADER CURLOPT_HEADERDATA
HEADERSTEM	Specify the stem variable of an array into which any headers from the performed action are written. If the array exists, it will be over-written. Where possible, each entry in the array will contain a single line.	CURLOPT_WRITEHEADER
HTTP200ALIASES	Specify a stem containing a list of aliases to be treated as valid HTTP 200 responses. Some servers respond with a custom header response line. For example, IceCast servers respond with "ICY 200 OK". By including this string in your list of aliases, the response will be treated as a valid HTTP header line such as "HTTP/1.0 200 OK".	CURLOPT_HTTP200ALIASES
HTTPAUTH	Pass one or more of the following string values as seperate arguments. These options tell	CURLOPT_HTTPAUTH

Rexx/CURL Option	Description	cURL Equivalent Option
	cURL which HTTP authentication options to attempt from the list. ◊ **"BASIC"**: HTTP Basic authentication. This is the default choice, and the only method that is in wide-spread use and supported virtually everywhere. This is sending the user name and password over the network in plain text, easily captured by others. ◊ **"DIGEST"**: HTTP Digest authentication. Digest authentication is defined in RFC2617 and is a more secure way to do authentication over public networks than the regular old-fashioned Basic method. ◊ **"GSSNEGOTIATE"**: HTTP GSS-Negotiate authentication. The GSS-Negotiate (also known as plain "Negotiate") method was designed by Microsoft and is used in their web applications. It is primarily meant as a support for Kerberos5 authentication but may be also used along with another authentication methods. For more information see IETF draft draft-brezak-spnego-http-04.txt. ◊ **"NTLM"**: HTTP NTLM authentication. A proprietary protocol invented and used by Microsoft. It uses a challenge-response and hash concept similar to Digest, to prevent the password from being eavesdropped. ◊ **"ANY"**: This is a convenience macro that sets all bits and thus makes libcurl pick any it finds suitable. libcurl will automatically select the one it finds most secure. ◊ **"ANYSAFE"**: This is a convenience macro that sets all bits except Basic and thus makes libcurl pick any it finds suitable. libcurl will automatically select the one it finds most secure. ◊ All of the above values with **"AUTH_"** prefix are now deprecated.	
HTTPCONTENTDECODING	Set this option to tell cURL how to act on content decoding. If set to **0** or **N**, content decoding will be disabled. If set to **1** or **Y** it is enabled. Note however that cURL has no default content decoding but requires you to use ENCODING for that.	CURLOPT_HTTP_CONTENT_DECODING
HTTPGET	Set this option to a **1** or **Y** to get cURL to return to HTTP GET mode. Really only useful if a POST was set with the same connection handle.	CURLOPT_HTTPGET
HTTPHEADER	To pass a series of HTTP headers to the server, set this option to a valid Rexx stem. Any	CURLOPT_HTTPHEADER

Rexx/CURL Option	Description	cURL Equivalent Option
	headers included in this option, that would have normally been generated internally by cURL, will be replaced.	
HTTPPOST	Set this option to a **1** or **Y** to indicate that a *regular* (application/x-www-form-urlencoded) HTTP POST is to be carried out. Most HTTP forms are of this type. See **HTTPPOSTFIELDS** option for details on how to specify the content of the form fields.	CURLOPT_POST
HTTPPOSTDATA	Setting this option indicates that you wish to issue a multipart/formdata HTTP POST. You pass the data that is posted as a valid Rexx stem.	CURLOPT_HTTPPOST
HTTPPOSTFIELDS	Specify the content of the fields to be filled in with a **HTTPPOST**. The passed parameter is a Rexx array, with each item in the array a name/value pair. eg field.1 = 'email=mark@rexx.org' and field.0 is the number items in the array.	CURLOPT_POSTFIELDS CURLOPT_POSTFIELDSIZE
HTTPPOSTFORM	Setting this option indicates that the HTTP form data specified with CURLFORMADD is to be posted. No data is passed with this option.	CURLOPT_HTTPPOST
HTTPPROXYTUNNEL	Set this option to a **1** or **Y** to tunnel all non-http operations through the HTTP proxy.	CURLOPT_HTTPPROXYTUNNEL
HTTPPUT	Set this option to a **1** or **Y** to get indicate that a HTTP PUT command is issued for the URL. The file to be uploaded must be specified with **INFILE**.	CURLOPT_PUT
HTTPTRANSFERDECODING	Set this option to tell cURL how to act on transfer decoding. If set to **0** or **N**, transfer decoding will be disabled. If set to **1** or **Y** it is enabled (default). cURL does chunked transfer decoding by default unless this option is set to zero.	CURLOPT_HTTP_TRANSFER_DECODING
HTTPVERSION	Set this option to "**VERSION_NONE**", "**VERSION_1_0**" or "**VERSION_1_1**" to specify the version to be used in HTTP requests.	CURLOPT_HTTP_VERSION
IGNORECONTENTLENGTH	Ignore the Content-Length header. This is useful for Apache 1.x (and similar servers) which will report incorrect content length for files over 2 gigabytes. If this option is used, curl will not be able to accurately report progress, and will simply stop the download when the server ends the connection.	CURLOPT_IGNORE_CONTENT_LENGTH
INFILE	The name of a file from which data is read for the performed action. This file must exist. For ftp uploads, this is the file to upload.	CURLOPT_INFILE CURLOPT_READDATA
INSTEM	Specify the stem variable of an array from which any input for the performed action is read. You can pass an extra argument specifying a character or sequence of characters to be appended to the value of the variable. This is useful for supplying text files via a stem. In Regina you could specify	CURLOPT_INFILE CURLOPT_READDATA

Rexx/CURL Option	Description	cURL Equivalent Option
	.endofline.	
INTERFACE	To specify an outgoing interface other than the default, pass the interface in this option. The interface can be specified as an IP address, a hostname, or an interface name (undex Unix).	CURLOPT_INTERFACE
IPRESOLVE	Allows an application to select what kind of IP addresses to use when resolving host names. This is only interesting when using host names that resolve addresses using more than one version of IP. The allowed values are: ◊ **"WHATEVER"**: Default, resolves addresses to all IP versions that your system allows. ◊ **"V4"**: Resolve to ipv4 addresses. ◊ **"V6"**: Resolve to ipv6 addresses. ◊ All of the above values with **"IPRESOLVE_"** prefix are now deprecated.	CURLOPT_IPRESOLVE
ISSUERCERT	Pass a string naming a file holding a CA certificate in PEM format. If the option is set, an additional check against the peer certificate is performed to verify the issuer is indeed the one associated with the certificate provided by the option. This additional check is useful in multi-level PKI where one needs to enforce that the peer certificate is from a specific branch of the tree. This option makes sense only when used in combination with the **SSLVERIFYPEER** option. Otherwise, the result of the check is not considered as failure.	CURLOPT_ISSUERCERT
KEYPASSWD	Pass a string to be used as the password to use the **SSLKEY** or **SSHPRIVATEKEYFILE** private key. You never needed a pass phrase to load a certificate but you need one to load your private key.	CURLOPT_KEYPASSWD
KRB4LEVEL Use:**KRBLEVEL**	Set the krb4 security level, this also enables krb4 awareness. This is a string, "**clear**", "**safe**", "**confidential**" or "**private**". If the string is set but doesn't match one of these, "**private**" will be used. Pass the empty string to disable kerberos4. The kerberos support only works for FTP.	CURLOPT_KRB4LEVEL
KRBLEVEL	Set the kerberos security level for FTP; this also enables kerberos awareness. This is a string, "**clear**", "**safe**", "**confidential**" or "**private**". If the string is set but doesn't match one of these, "**private**" will be used. Pass the empty string to disable kerberos.	CURLOPT_KRBLEVEL
LOCALPORT	Sets the local port number of the socket used for connection. This can be used in combination with the **INTERFACE** option and you are recommended to use **LOCALPORTRANGE** option as well when this is set. Note that port numbers are only valid 1 - 65535.	CURLOPT_LOCALPORT
LOCALPORTRANGE	This is the number of attempts libcurl should do to find a working local port number. It starts with the given **LOCALPORT** option	CURLOPT_LOCALPORTRANGE

Rexx/CURL Option	Description	cURL Equivalent Option
	and adds one to the number for each retry. Setting this value to 1 or below will make libcurl do only one try for exact port number. Note that port numbers by nature is a scarce resource that will be busy at times so setting this value to something too low might cause unnecessary connection setup failures.	
LOGINOPTIONS	Pass a string to specify login options for the transfer. For more information about the login options please see RFC 2384, RFC 5092 and IETF draft draft-earhart-url-smtp-00.txt. This option can be used to set protocol specific login options, such as the preferred authentication mechanism via "AUTH=NTLM" or "AUTH=*", and should be used in conjunction with the **USERNAME** option. Only IMAP, POP3 and SMTP protocols support login options.	CURLOPT_LOGIN_OPTIONS
LOWSPEEDLIMIT	This option should contain a number representing the bytes per second that cURL will use as the lowest transfer rate to run at before it aborts the session as being too slow.	CURLOPT_LOW_SPEED_LIMIT
LOWSPEEDTIME	This option should contain a number representing the number of seconds that cURL will use as the lowest transfer time to run for before it aborts the session as being too slow.	CURLOPT_LOW_SPEED_TIME
MAILAUTH	Pass a string to specify the authentication address (identity) of a submitted message that is being relayed to another server. This optional parameter allows co-operating agents in a trusted environment to communicate the authentication of individual messages and should only be used by the application program, using libcurl, if the application is itself a mail server acting in such an environment. If the application is operating as such and the AUTH address is not known or is invalid, then an empty string should be used for this parameter. Unlike **MAILFROM** and **MAILRCPT** options, the address should not be specified within a pair of angled brackets (<>). However, if an empty string is used then a pair of brackets will be sent by libcurl as required by RFC-2554.	CURLOPT_MAIL_AUTH
MAILFROM	Pass a string to specify the sender's email address when sending SMTP mail. An originator email address should be specified with angled brackets (<>) around it, which if not specified, will be added by libcurl from version 7.21.4 onwards. Failing to provide such brackets may cause the server to reject the email. If this parameter is not specified then an empty address will be sent to the mail server which may or may not cause the email to be rejected.	CURLOPT_MAIL_FROM
MAILRCPT	Specify a stem with a list of recipients to pass to the server in your SMTP mail request. Each recipient should be specified within a pair of angled brackets (<>), however, should you not	CURLOPT_MAIL_RCPT

Rexx/CURL Option	Description	cURL Equivalent Option
	use an angled bracket as the first character libcurl will assume you provided a single email address and enclose that address within brackets for you.	
MAXCONNECTS	You can specify how many persistent connections cURL is to maintain. The default is 5 and unless you know what you are doing leave it alone.	CURLOPT_MAXCONNECTS
MAXFILESIZE	This allows you to 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 FILESIZE_EXCEEDED will be returned. **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.	CURLOPT_MAXFILESIZE CURLOPT_MAXFILESIZE_LARGE
MAXRECVSPEEDLARGE	Pass an integer. If a download exceeds this speed (counted in bytes per second) on cumulative average during the transfer, the transfer will pause to keep the average rate less than or equal to the parameter value. Defaults to unlimited speed.	CURLOPT_MAX_RECV_SPEED_LARGE
MAXREDIRS	To limit the number of redirections followed, set this value with this option. This option only makes sense when **FOLLOWLOCATION** is also set.	CURLOPT_MAXREDIRS
MAXSENDSPEEDLARGE	Pass an integer. If an upload exceeds this speed (counted in bytes per second) on cumulative average during the transfer, the transfer will pause to keep the average rate less than or equal to the parameter value. Defaults to unlimited speed.	CURLOPT_MAX_SEND_SPEED_LARGE
NETRC	This parameter controls the preference of libcurl between using user names and passwords from your ~/.netrc file, relative to user names and passwords in the URL supplied with the URL option. **Note:** cURL uses a user name (and supplied or prompted password) supplied with the option USERPWD in preference to any of the options controlled by this parameter. ◊ **"OPTIONAL"**: The use of your ~/.netrc file is optional, and information in the URL is to be preferred. The file will be scanned with the host and user name (to find the password only) or with the host only, to find the first user name and password after that *machine*, which ever information is not specified in the URL. Undefined values of the option will have this effect. ◊ **"IGNORED"**: The library will ignore the file and use only the information in the URL. This is the default. ◊ **"REQUIRED"**: This value tells the	CURLOPT_NETRC

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
	library that use of the file is required, to ignore the information in the URL, and to search the file with the host only. Only machine name, user name and password are taken into account (init macros and similar things aren't supported). **Note:** cURL does not verify that the file has the correct properties set (as the standard Unix ftp client does). It should only be readable by user.	
NETRCFILE	The name of a file you want to be used as a .netrc file. If this option is omitted, and NETRC option is set, then Rexx/CURL attempts to look for .netrc in the current user's home directory.	CURLOPT_NETRC_FILE
NEWDIRECTORYPERMS	Pass a long as a parameter, containing the value of the permissions that will be assigned to newly created directories on the remote server. The default value is 0755, but any valid value can be used. The only protocols that can use this are sftp://, scp:// and file://.	CURLOPT_NEW_DIRECTORY_PERMS
NOBODY	Set this option to a **1** or **Y** to get cURL to **not** return the body information for those protocols, like HTTP, that have seperate headers and footers.	CURLOPT_NOBODY
NOPROGRESS	Set this option to a **1** or **Y** to turn off cURL's default progress meter.	CURLOPT_NOPROGRES
NOPROXY	Pass a string which should be a 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.	CURLOPT_NOPROXY
NOSIGNAL	N/A to Rexx/CURL	CURLOPT_NOSIGNAL
OUTFILE	The name of a file into which any output from the performed action is written. By default, if this file exists, it will be over-written. You can pass **APPEND** as an extra argument to append data to an existing file. This is necessary when using RESUMEFROM to resume a download.	CURLOPT_FILE CURLOPT_WRITEDATA
OUTSTEM	Specify the stem variable of an array into which any output from the performed action is written. If the array exists, it will be over-written. This option should only be used when the output is expected to be line oriented with a known line terminating sequence of characters. By default the output is split into lines using the line termination character; x0A. You can pass the line termination sequence as the 4th argument. Output from performing an HTML request or from reading email will be line terminated with the CRLF sequence; x0D0A, and this value should be used as the	CURLOPT_FILE CURLOPT_WRITEDATA

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
	4th argument.	
PASWORD	Pass a string which is used as the password for the transfer. This option should be used in conjunction with the **USERNAME** option.	CURLOPT_PASSWORD
POSTQUOTE	See the prefered FTPCMDSAFTER option.	CURLOPT_POSTQUOTE
POSTREDIR	Pass one or more of the following string values as seperate arguments. These options control how libcurl acts on redirects after POSTs that get a 301, 302 or 303 response back. ◊ **"GET_ALL"**: Tells the library to change a POST request to a GET request after a 301 or 302 response ◊ **"POST_301"**: Tells the library to respect RFC 2616/10.3.2 and not convert POST requests into GET requests when following a 301 redirection. ◊ **"POST_302"**: Makes libcurl maintain the request method after a 302 redirect. ◊ **"POST_ALL"**: A convenience option that sets both **POST_301** and **POST_302**	CURLOPT_POSTREDIR
POST301 Use:**POSTREDIR**	Set this option to a **1** or **Y** to tell the library to respect RFC 2616/10.3.2 and not convert POST requests into GET requests when following a 301 redirection. The non-RFC behaviour is ubiquitous in web browsers, so the library does the conversion by default to maintain consistency. However, a server may requires a POST to remain a POST after such a redirection. This option is meaningful only when setting the option **FOLLOWLOCATION**.	CURLOPT_POST301
PREQUOTE	See the prefered FTPCMDSBEFORE option.	CURLOPT_PREQUOTE
PRIVATE	Pass a string of private data that can be associated with the cURL handle to be retrieved later with a call to CURLGETINFO with the PRIVATE option.	CURLOPT_PRIVATE
PROGRESSFUNCTION	When uploading or downloading, cURL will call the specified internal Rexx procedure at regular intervals. Four arguments are passed; total bytes to download, bytes downloaded so far, total bytes to upload, bytes uploaded so far. This option is only valid if your Rexx interpreter has the RexxCallback() API (to date only Regina has this).	CURLOPT_PROGRESSFUNCTION
PROTOCOLS	Pass one or more string values as seperate arguments. These options limit which protocols libcurl may use in the transfer. This allows you to have a libcurl built to support a wide range of protocols but still limit specific transfers to only be allowed to use a subset of them. By default libcurl will accept all protocols it supports. See the values of the **!REXXCURL.!PROTOCOLS** stem for possible values. See also	CURLOPT_PROTOCOLS

Rexx/CURL Option	Description	cURL Equivalent Option
	REDIR_PROTOCOLS.	
PROXY	If you need to use a HTTP proxy to access the outside world, specify it with this option. The format of the proxy string is *host[:port]*. The default port number is 1080.	CURLOPT_PROXY
PROXYAUTH	Pass one or more string values as seperate arguments. These options tell cURL which PROXY authentication options to attempt from the list. See HTTPAUTH for the list of valid string values.	CURLOPT_PROXYAUTH
PROXYPASSWORD	Pass a string containing the password to use for the transfer while connecting to a Proxy. This option should be used in conjunction with the **PROXYUSERNAME** option.	CURLOPT_PROXYPASSWORD
PROXYPORT	Specify the port to use for the proxy server.	CURLOPT_PROXYPORT
PROXYTRANSFERMODE	Set this option to a **1** or **Y** to tell libcurl to set the transfer mode (binary or ASCII) for FTP transfers done via a HTTP proxy, by appending ;type=a or ;type=i to the URL. Without this setting, or it being set to **0** or **N** (the default), **TRANSFERTEXT** has no effect when doing FTP via a proxy. Beware that not all proxies support this feature.	CURLOPT_PROXY_TRANSFER_MODE
PROXYTYPE	Set this option to "**HTTP**", "**SOCKS4**" or "**SOCKS5**".	CURLOPT_PROXYTYPE
PROXYUSERNAME	Pass a string containing the user name to use for the transfer while connecting to a Proxy. This option should be used in same way as the **PROXYUSERPWD** is used. In comparison to **PROXYUSERPWD** this option allows the username to contain a colon, like in the following example: "sip:user@example.com". This option is an alternative way to set the user name while connecting to Proxy. There is no meaning to use it together with the **PROXYUSERPWD** option. In order to specify the password to be used in conjunction with the user name use the **PROXYPASSWORD** option.	CURLOPT_PROXYUSERNAME
PROXYUSERPWD	Specify the username/password to use for the the HTTP proxy connection. The format is *username[:password]*. If the password is omitted, you will be prompted for it.	CURLOPT_PROXYUSERPWD
QUOTE	See the prefered **FTPCMDS** option.	CURLOPT_QUOTE
RANDOMFILE	The name of a file from which is used to seed the SSL random engine. This file must exist.	CURLOPT_RANDOM_FILE
RANGE	Specify the required range you want in the format *X-Y*. I have no idea what this is!	CURLOPT_RANGE
REDIRPROTOCOLS	Pass one or more string values as seperate arguments. These options limit which protocols libcurl may use after a redirection if **FOLLOWLOCATION** is set to 1. By default libcurl will allow all protocols except for FILE and SCP. See the values of the **!REXXCURL.!PROTOCOLS** stem for	CURLOPT_REDIR_PROTOCOLS

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
	possible values. See also **PROTOCOLS**.	
REFERER	Set this option to a string to set the header field: *referer:* in the HTTP request.	CURLOPT_REFERER
RESOLVE	Specify a stem containing strings with host name resolve information to use for requests with this handle. Each single name resolve string should be written using the format **HOST:PORT:ADDRESS** where **HOST** is the name libcurl will try to resolve, **PORT** is the port number of the service where libcurl wants to connect to the **HOST** and **ADDRESS** is the numerical IP address. If libcurl is built to support IPv6, **ADDRESS** can of course be either IPv4 or IPv6 style addressing. This option effectively pre-populates the DNS cache with entries for the host+port pair so redirects and everything that operations against the **HOST+PORT** will instead use your provided **ADDRESS**. You can remove names from the DNS cache again, to stop providing these fake resolves, by including a string in the stem that uses the format **"-HOST:PORT"**. The host name must be prefixed with a dash, and the host name and port number must exactly match what was already added previously.	CURLOPT_RESOLVE
RESUMEFROM	Set this option to the byte at which you want a request to start from. This is particularly useful for restarting a download that was interrupted.	CURLOPT_RESUME_FROM CURLOPT_RESUME_FROM_LARGE
RTSPCLIENTCSEQ	Manually set the the CSEQ number to issue for the next RTSP request. Useful if the application is resuming a previously broken connection. The CSEQ will increment from this new number henceforth.	CURLOPT_RTSP_CLIENT_CSEQ
RTSPHEADER	This option is simply an alias for **HTTPHEADER**. Use this to replace the standard headers that RTSP and HTTP share. It is also valid to use the shortcuts such as **USERAGENT** option.	CURLOPT_RTSP_HEADER
RTSPREQUEST	Tell libcurl what kind of RTSP request to make. Pass one of the following strings. Unless noted otherwise, commands require the Session ID to be initialized (via **RTSPSESSIONID**. ◊ "**OPTIONS**" - Used to retrieve the available methods of the server. The application is responsbile for parsing and obeying the response. (The session ID is not needed for this method.) ◊ "**DESCRIBE**" - Used to get the low level description of a stream. The application should note what formats it understands in the "Accept:" header. Unless set manually, libcurl will automatically fill in "Accept: application/sdp". Time-condition headers will be added to Describe requests if the **TIMECONDITION**	CURLOPT_RTSP_REQUEST

Rexx/CURL Option	Description	cURL Equivalent Option
	option is active. (The session ID is not needed for this method) ◊ "**ANNOUNCE**" - When sent by a client, this method changes the description of the session. For example, if a client is using the server to record a meeting, the client can use Announce to inform the server of all the meta-information about the session. **ANNOUNCE** acts like a HTTP PUT or POST just like **SET_PARAMETER** sub-option ◊ "**SETUP**" - Setup is used to initialize the transport layer for the session. The application must set the desired Transport options for a session by using the **TRANSPORT** sub-option prior to calling setup. If no session ID is currently set with **SESSIONID** option, libcurl will extract and use the session ID in the response to this request. (The session ID is not needed for this method). ◊ "**PLAY**" - Send a Play command to the server. Use the **RANGE** option to modify the playback time (e.g. 'npt=10-15'). ◊ "**PAUSE**" - Send a Pause command to the server. Use the **RANGE** option with a single value to indicate when the stream should be halted. (e.g. npt='25') ◊ "**TEARDOWN**" - This command terminates an RTSP session. Simply closing a connection does not terminate the RTSP session since it is valid to control an RTSP session over different connections. ◊ "**GET_PARAMETER**" - Retrieve a parameter from the server. By default, libcurl will automatically include a "Content-Type: text/parameters" header on all non-empty requests unless a custom one is set. **GET_PARAMETER** acts just like a HTTP PUT or POST (see SET_PARAMETER sub-option). Applications wishing to send a heartbeat message (e.g. in the presence of a server-specified timeout) should send an empty **GET_PARAMETER** request. ◊ "**SET_PARAMETER**" - Set a parameter on the server. By default, libcurl will automatically include a "Content-Type: text/parameters" header unless a custom one is set. The interaction with **SET_PARAMTER** is much like a HTTP PUT or POST. An application may either use **UPLOAD** option with **READDATA** option like a HTTP PUT, or it may use	

Rexx/CURL Option	Description	cURL Equivalent Option
	POSTFIELDS like a HTTP POST. Also, there is no use of multi-part POSTs within RTSP. ◊ "**RECORD**" - Used to tell the server to record a session. Use the **RANGE** option to modify the record time. ◊ "**RECEIVE**" - This is a special request because it does not send any data to the server. The application may call this function in order to receive interleaved RTP data. It will return after processing one read buffer of data in order to give the application a chance to run.	
RTSPSERVERCSEQ	Manually set the CSEQ number to expect for the next RTSP Server->Client request. At the moment, this feature (listening for Server requests) is unimplemented.	CURLOPT_RTSP_SERVER_CSEQ
RTSPSESSIONID	Pass a string to set the value of the current RTSP Session ID for the handle. Useful for resuming an in-progress session. Once this value is set to any non-empty value, libcurl will return "**RTSP_SESSION_ERROR**" if ID received from the server does not match. If unset (or set to the empty string), libcurl will automatically set the ID the first time the server sets it in a response.	CURLOPT_RTSP_SESSION_ID
RTSPSTREAMURI	Set the stream URI to operate on by passing a string . For example, a single session may be controlling *rtsp://foo/twister/audio* and *rtsp://foo/twister/video* and the application can switch to the appropriate stream using this option. If unset, libcurl will default to operating on generic server options by passing "*" in the place of the RTSP Stream URI. This option is distinct from **URL**. When working with RTSP, this option indicates what URL to send to the server in the request header while the **URL** indicates where to make the connection to. (e.g. the **URL** for the above examples might be set to *rtsp://foo/twister*	CURLOPT_RTSP_STREAM_URI
RTSPTRANSPORT	Pass a string to tell libcurl what to pass for the Transport: header for this RTSP session. This is mainly a convenience method to avoid needing to set a custom Transport: header for every **SETUP** request. The application must set a Transport: header before issuing a **SETUP** request.	CURLOPT_RTSP_TRANSPORT
SOCKS5GSSAPINEC	Set this option to a **1** or **Y** to enable or **0** or **N** to disable. As part of the gssapi negotiation a protection mode is negotiated. The rfc1961 says in section 4.3/4.4 it should be protected, but the NEC reference implementation does not. If enabled, this option allows the unprotected exchange of the protection mode negotiation.	CURLOPT_SOCKS5_GSSAPI_NEC
SOCKS5GSSAPISERVICE	Pass a string holding the name of the service. The default service name for a SOCKS5 server is rcmd/server-fqdn. This option allows you to	CURLOPT_SOCKS5_GSSAPI_SERVICE

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
	change it.	
SOURCEPOSTQUOTE Removed in Rexx/CURL 2.0	Exactly like, **POSTQUOTE**, but for source host.	CURLOPT_SOURCE_POSTQUOTE
SOURCEPREQUOTE Removed in Rexx/CURL 2.0	Exactly like, **PREQUOTE**, but for source host.	CURLOPT_SOURCE_PREQUOTE
SOURCEQUOTE Removed in Rexx/CURL 2.0	Exactly like, **QUOTE**, but for source host.	CURLOPT_SOURCE_QUOTE
SOURCEURL Removed in Rexx/CURL 2.0	When set, it enables a FTP third party transfer, using this value as source, while the option; **URL** is the target.	CURLOPT_SOURCE_URL
SOURCEUSERPWD Removed in Rexx/CURL 2.0	Set this to a string in the format; "username:password" to use for the source connection when doing FTP third party transfers.	CURLOPT_SOURCE_USERPWD
SSHAUTHTYPES	Pass one or more of the following string values as seperate arguments. These are the authentication types cURL will use for SSH. ◊ **"PUBLICKEY"** ◊ **"PASSWORD"** ◊ **"HOST"** ◊ **"KEYBOARD"** ◊ **"ANY"**: Let cURL pick one. ◊ All of the above values with **"AUTH_"** prefix are now deprecated.	CURLOPT_SSH_AUTH_TYPES
SSHHOSTPUBLICKEYMD5	Pass a string containing 32 hexadecimal digits. The string should be the 128 bit MD5 cheksum of the remote host's public key, and Rexx/CURL will reject the connection to the host unless the md5sums match. This option is only for SCP and SFTP transfers.	CURLOPT_SSH_HOST_PUBLIC_KEY_MD5
SSHKNOWNHOSTS	Pass a string holding the file name of the known_host file to use. The known_hosts file should use the OpenSSH file format as supported by libssh2. If this file is specified, libcurl will only accept connections with hosts that are known and present in that file, with a matching public key. Use **SSHKEYFUNCTION** to alter the default behavior on host and key (mis)matching.	CURLOPT_SSH_KNOWN_HOSTS
SSHPRIVATEKEYFILE	Pass a string pointing to a file name for your public key. If not used, libcurl defaults to using ~/.ssh/id_dsa.pub.	CURLOPT_SSH_PRIVATE_KEY_FILE
SSHPUBLICKEYFILE	Pass a string pointing to a file name for your private key. If not used, libcurl defaults to using ~/.ssh/id_dsa. If the file is password-protected, set the password with **KEYPASSWD**.	CURLOPT_SSH_PUBLIC_KEY_FILE
SSLCERT	To set a SSL certificate, set this option to the filename containing the certificate. The certificate should be in PEM format.	CURLOPT_SSLCERT
SSLCERTPASSWD Use:**KEYPASSWD**	The password associated with the SSL certificate set by **SSLCERT** can be set with this option. If you don't supply the password with this option, you will be prompted for it.	CURLOPT_SSLCERTPASSWD

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
SSLCERTTYPE	The type of SSL certificate. One of "**DER**" or "**PEM**".	CURLOPT_SSLCERTTYPE
SSLCIPHERLIST	The list of ciphers to use for the SSL connection. It consists of one or more cipher strings separated by colons. Commas or spaces are also acceptable separators but colons are normally used, "-" and "+" can be used as operators. Valid examples of cipher lists include "RC4-SHA", "SHA1+DES", "TLSv1" and "DEFAULT". The default list is normally set when you compile OpenSSL. You'll find more details about cipher lists here	CURLOPT_SSL_CIPHER_LIST
SSLENGINE	Identifies the name of the crypto engine used for your private key.	CURLOPT_SSL_ENGINE
SSLENGINEDEFAULT	Identifies the name of the default crypto engine used for asymetric crypto operations.	CURLOPT_SSL_ENGINEDEFAULT
SSLKEY	The name of the file containing your private key.	CURLOPT_SSLKEY
SSLKEYPASSWD Use:**KEYPASSWD**	This will be used as the password required to use the SSLKEY private key.	CURLOPT_SSLKEYPASSWD
SSLKEYTYPE	The format of your private key. One of "**DER**", "**PEM**" or "**ENG**".	CURLOPT_SSLKEYTYPE
SSLOPTIONS	Pass one or more of the following string values as seperate arguments to tell libcurl about specific SSL behaviors: ◊ ALLOWBEAST - tell libcurl to not attempt to use any work-arounds for a security flaw in the SSL3 and TLS1.0 protocols. If this sub-option isn't used, the SSL layer libcurl uses may use a work-around for this flaw although it might cause interoperability problems with some (older) SSL implementations. WARNING: avoiding this work-around loosens the security, and by setting this sub-option you ask for exactly that. ◊ NO_REVOKE - tells libcurl to disable certificate revocation checks for those SSL backends where such behavior is present.	CURLOPT_SSL_OPTIONS
SSLPEERCERT	The name of a file from which the SSL peer certificate data is read for the performed action. This file must exist.	CURLOPT_CAINFO
SSLSESSIONIDCACHE	Set this option to **0** or **N** to disable cURL's use of SSL session-ID caching. Set this to a **1** or **Y** to enable it. By default all transfers are done using the cache. Note that while nothing ever should 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.	CURLOPT_SSL_SESSIONID_CACHE
SSLVERIFYHOST	Set this if we should verify the Common name from the peer certificate in the SSL hand-shake, set to "**1**" to check existence, "**2**" to ensure that it matches the provided	CURLOPT_SSL_VERIFYHOST

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
	hostname.	
SSLVERIFYPEER	Set this option to a **1** or **Y** to indicate that the SSL peer's certificate should be verified. The SSL peer certificate must be specified with the **SSLPEERCERT** option.	CURLOPT_SSL_VERIFYPEER
SSLVERIFYSTATUS	Set this option to a **1** or **Y** to determines whether libcurl verifies the status of the server cert using the "Certificate Status Request" TLS extension (aka. OCSP stapling). Note that if this option is enabled but the server does not support the TLS extension, the verification will fail.	CURLOPT_SSL_VERIFYSTATUS
SSLVERSION	To over-ride the default SSL version used, pass this option in as a number.	CURLOPT_SSLVERSION
TCPKEEPALIVE	Set this option to a **1** or **Y** to indicate that TCP keepalive probes will be sent. The delay and frequency of these probes can be controlled by the **TCPKEEPIDLE** and **TCPKEEPINTVL** options, provided the operating system supports them. The default behaiour is to disable keepalive probes.	CURLOPT_TCP_KEEPALIVE
TCPKEEPIDLE	Specify the delay, in seconds, that the operating system will wait while the connection is idle before sending keepalive probes. Not all operating systems support this option.	CURLOPT_TCP_KEEPIDLE
TCPKEEPINTVL	Specify the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option.	CURLOPT_TCP_KEEPINTVL
TCPNODELAY	Set this option to a **1** or **Y** to get cURL to disable TCP's Nagle algorithm. The purpose of this algorithm is to try to minimize the number of small packets on the network (where "small packets" means TCP segments less than the Maximum Segment Size (MSS) for the network). Maximizing the amount of data sent per TCP segment is good because it amortizes the overhead of the send. However, in some cases (most notably telnet or rlogin) small segments may need to be sent without delay. This is less efficient than sending larger amounts of data at a time, and can contribute to congestion on the network if overdone. The option is off by default. This will have no effect after the connection has been established.	CURLOPT_TCP_NODELAY
TFTPBLKSIZE	Specify the block size to use for TFTP data transmission. Valid range as per RFC 2348 is 8-65464 bytes. The default of 512 bytes will be used if this option is not specified. The specified block size will only be used pending support by the remote server. If the server does not return an option acknowledgement or returns an option acknowledgement with no blksize, the default of 512 bytes will be used.	CURLOPT_TFTP_BLKSIZE

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
TIMECONDITION	This defines how the TIMEVALUE time value is treated. You can set this parameter to "**IFMODSINCE**", "**TIMECOND_IFUNMODSINCE**" or "**LASTMOD**". This option is valid for HTTP or FTP.	CURLOPT_TIMECONDITION
TIMEOUT	Specify the number of seconds in which the whole transaction is to complete. Note that this timeout period also includes the time taken to do name lookups, so don't specify too short a period.	CURLOPT_TIMEOUT
TIMEOUTMS	Like TIMEOUT but takes number of milliseconds instead.	CURLOPT_TIMEOUT_MS
TIMEVALUE	This should be the time in seconds since 1 Jan 1970 (Unix time_t format), and the time will be used in a condition as specified with TIMECONDITION.	CURLOPT_TIMEVALUE
TLSAUTHPASSWORD	Pass the password to use for the TLS authentication method specified with the **TLSAUTHTYPE** option. Requires that the **TLSUSERNAME** option also be set.	CURLOPT_TLSAUTH_PASSWORD
TLSAUTHTYPE	Pass one or more of the following string values as seperate arguments to tell libcurl which authentication method(s) you want it to use for TLS authentication. ◊ SRP - Secure Remote Password authentication for TLS is defined in RFC 5054 and provides mutual authentication if both sides have a shared secret. To use TLS-SRP, you must also set the **TLSAUTHUSERNAME** and **TLSAUTHPASSWORD** options. The libcurl being used needs to have been built with GnuTLS or OpenSSL with TLS-SRP support for this to work; ie !REXXCURL.!SUPPORTS_TLSAUTH_SRP has a value of 1.	CURLOPT_TLSAUTH_TYPE
TLSAUTHUSERNAME	Pass the username to use for the TLS authentication method specified with the **TLSAUTHTYPE** option. Requires that the **TLSPASSWORD** option also be set.	CURLOPT_TLSAUTH_USERNAME
TRANSFERENCODING	Set this option to a **1** or **Y** to request compressed Transfer Encoding in the outgoing HTTP request. If the server supports this and so desires, it can respond with the HTTP resonse sent using a compressed Transfer-Encoding that will be automatically uncompressed by libcurl on receival. Transfer-Encoding differs slightly from the Content-Encoding you ask for with **ACCEPTENCODING** option in that a Transfer-Encoding is strictly meant to be for the transfer and thus MUST be decoded before the data arrives in the client. Traditionally, Transfer-Encoding has been much less used and supported by both HTTP clients and HTTP servers.	CURLOPT_TRANSFER_ENCODING

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
TRANSFERTEXT	Set this option to a **1** or **Y** to get indicate that FTP transfers are to be done in ASCII mode. For LDAP requests, the details are returned in plain text rather than HTML.	CURLOPT_TRANSFERTEXT
UNRESTRICTEDAUTH	Set this option to a **1** or **Y** to indicate that Rexx/CURL can continue to send authentication (user+password) when following locations, even when hostname changed. Note that this is meaningful only when setting FOLLOWLOCATION.	CURLOPT_UNRESTRICTED_AUTH
UPLOAD	Set this option to a **1** or **Y** to indicate that an upload is to be done, rather than a download.	CURLOPT_UPLOAD
URL	The URL against which the access is to be made.	CURLOPT_URL
USERAGENT	Set this option to a string to set the header field: *user-agent:* in the HTTP request. This can be useful to imitate different browser clients.	CURLOPT_USERAGENT
USERPWD	Specify the username/password to use for this connection. The format is *username[:password]*. If the password is omitted, you will be prompted for it.	CURLOPT_USERPWD
USESSL	Set this option to one of the following string values to make libcurl use your desired level of SSL for the transfer. ◊ **"NONE"**: Don't attempt to use SSL. ◊ **"TRY"**: Try using SSL, proceed as normal otherwise. ◊ **"CONTROL"**: Require SSL for the control connection or fail with USE_SSL_FAILED. ◊ **"ALL"**: Require SSL for all communication or fail with USE_SSL_FAILED.	CURLOPT_USE_SSL
VERBOSE	Set this option to a **1** or **Y** to get cURL to display lots of details about what it is doing.	CURLOPT_VERBOSE
XOATH2BEARER	Specifies the OAuth 2.0 Bearer Access Token for use with HTTP, IMAP, POP3 and SMTP servers that support the OAuth 2.0 Authorization Framework. For IMAP, POP3 and SMTP, the user name used to generate the Bearer Token should be supplied via the **USERNAME** option.	CURLOPT_XOATH2_BEARER

option value

The value of the option to set.

Returns:

success:

blank

failure:

blank

On *failure* **CURLERROR.INTCODE** is set to a non-zero value. If this value is 1 (one) then **CURLERROR.CURLCODE** is also set to a non-zero value.

CURLFORMADD(*handle,option=COPYCONTENTS,section name,content type,content*)
CURLFORMADD(*handle,option=FILE,section name,content type,filename*,[...])

CURLFORMADD(*handle*,*option=FILE*,*section name*,*content type array*,*filename array*)

This function is used to append sections when building multipart/formdata HTTP POST transfers. Any number of sections can be added by calling this multiple times before executing the transfer with the CURLPERFORM function.
Once CURLPERFORM has been called, call CURLFORMFREE to free resources used in this function.

Arguments:

handle
The value returned from CURLINIT.

option
One of **COPYCONTENTS** or **FILE** which specifies the valid arguments to follow.

section name
The name to associate with this section of posted data.

content type
The type of data present in *content*.**Mandatory for COPYCONTENTS option, but for FILE option, if not specified, cURL tries to guess.**

content
The data comprising this part.

filename
The name of the file containing the data for this part.

filename array
A Rexx *array* containing names of the files containing the data for this part.

content type array
A Rexx *array* of the types of data present in the *filename array*.**If an item in the array is not specified, cURL tries to guess.**

Returns:

success:
blank
failure:
blank
On *failure* **CURLERROR.INTCODE** is set to a non-zero value. If this value is 1 (one) then **CURLERROR.CURLCODE** is also set to a non-zero value.

CURLFORMFREE(*handle*)

This function is required to be called after calling CURLPERFORM if HTTP POST data is specified by calling CURLFORMADD.

Arguments:

handle
The value returned from CURLINIT.

Returns:

success:
blank
failure:
blank
On *failure* **CURLERROR.INTCODE** is set to a non-zero value. If this value is 1 (one) then **CURLERROR.CURLCODE** is also set to a non-zero value.

CURLPERFORM(*handle*)

Once all the options have been set up, call this function to carry out the transfer.

Arguments:

handle

The value returned from CURLINIT.

Returns:

success:
blank
failure:
blank
On *failure* **CURLERROR.INTCODE** is set to a non-zero value. If this value is 1 (one) then **CURLERROR.CURLCODE** is also set to a non-zero value.

CURLGETINFO(*handle*, *option* [,*stem*]

Retrieves information about the most recently executed command.
Arguments:

handle
The value returned from CURLINIT.
option
This is the string identifying the information to retrieve.

Rexx/CURL Option	Description	cURL Equivalent Option
APPCONNECT_TIME	Returns the time, in seconds, it took from the start until the SSL/SSH connect/handshake to the remote host was completed. This time is most often very near to the **PRETRANSFER_TIME** time, except for cases such as HTTP pipelining where the pretransfer time can be delayed due to waits in line for the pipeline and more.	CURLINFO_APPCONNECT_TIME
CERTINFO	Returns information about all certificate chains in the named stem (argument 3) assuming you had **CERTINFO** of CURLSETOPT set when the previous request was done. NOTE: this option is only available in libcurl built with OpenSSL support.	CURLINFO_CERTINFO
CONDITION_UNMET	Returns 1 if the condition provided in the previous request didn't match (see **TIMECONDITION** option). Alas, if this returns a 1 you know that the reason you didn't get data in return is because it didn't fulfill the condition. Zero will be returned if the condition instead was met.	CURLINFO_CONDITION_UNMET
CONNECT_TIME	Returns the number of seconds it took to connect to the remote server.	CURLINFO_CONNECT_TIME
CONTENT_LENGTH_DOWNLOAD	Returns the length of the contents returned. This is the value returned by the header field; *Content-Length:* Since cURL 7.19.4 this will return -1 if the size isn't known.	CURLINFO_CONTENT_LENGTH_DOWNLOAD
CONTENT_LENGTH_UPLOAD	Returns the length of the specified upload size. Since cURL 7.19.4 this will return -1 if the size isn't known.	CURLINFO_CONTENT_LENGTH_UPLOAD
CONTENT_TYPE	Returns the content-type of the downloaded object. This is the value read from the "Content-Type:" field. If you get an empty string, it means	CURLINFO_CONTENT_TYPE

Rexx/CURL Option	Description	cURL Equivalent Option
	that the server didn't send a valid Content-Type header or that the protocol used doesn't support this.	
COOKIELIST	Returns the list of all cookies cURL knows (expired ones, too). The results are returned in the named stem (argument 3).	CURLINFO_COOKIELIST
EFFECTIVE_URL	Returns the last used effective URL.	CURLINFO_EFFECTIVE_URL
FILE_TIME	The file date and time of the remote document received in Unix time_t format. This is only returned if setopt **FILETIME** is called.	CURLINFO_FILETIME
FTP_ENTRY_PATH	Returns a string containing the initial path Rexx/CURL ended up in when logging into the FTP server. The empty string is returned if there was an error.	CURLINFO_FTP_ENTRY_PATH
HEADER_SIZE	Returns the length of all headers returned.	CURLINFO_HEADER_SIZE
HTTPAUTH_AVAIL	Returns a string containing the authentication method(s) available. The authentication method(s) are explained in the **HTTPAUTH** option for	CURLINFO_HTTPAUTH_AVAIL
HTTP_CODE Use:**RESPONSE_CODE**	Returns the last received HTTP code. Only useful if the last URL requested was HTTP.	CURLINFO_HTTP_CODE
HTTP_CONNECTCODE	Returns the last received proxy response code to a CONNECT	CURLINFO_HTTP_CONNECTCODE
LASTSOCKET	Returns the last socket used by this curl session. If the socket is no longer valid, -1 is returned. When you finish working with the socket, you must call CURLCLEANUP as usual and let libcurl close the socket and cleanup other resources associated with the handle. This is typically used in combination with **CONNECT_ONLY** option of CURLSETOPT.	CURLINFO_LASTSOCKET
LOCAL_IP	Returns a string holding the local (source) IP address of the most recent connection done with this curl handle. This string may be IPv6 if that's enabled. The same restrictions apply as to **PRIMARY_IP**.	CURLINFO_LOCAL_IP
LOCAL_PORT	Retruns the local (source) port of the most recent connection done with this curl handle.	CURLINFO_LOCAL_PORT
NAMELOOKUP_TIME	Returns the time in seconds for the time taken to resolve the remote server name.	CURLINFO_NAMELOOKUP_TIME
NUM_CONNECTS	Returns the number of new connections libcurl had to create to	CURLINFO_NUM_CONNECTS

Rexx/CURL Option	**Description**	**cURL Equivalent Option**
	achieve the previous transfer (only the successful connects are counted). Combined with CURLINFO_REDIRECT_COUNT you are able to know how many times libcurl successfully reused existing connection(s) or not. See the Connection Options of curl_easy_setopt(3) to see how libcurl tries to make persistent connections to save time.	
OS_ERRNO	Returns the operating system errno from a connect failure.	CURLINFO_OS_ERRONO
PRETRANSFER_TIME	Returns the total time taken from the start up until the transfer is about to begin. This time includes all pre-transfer commands and negotiations.	CURLINFO_PRETRANSFER_TIME
PRIMARY_IP	Retruns a string holding the IP address of the most recent connection done with this curl handle. This string may be IPv6 if that's enabled.	CURLINFO_PRIMARY_IP
PRIMARY_PORT	Retruns the destination port of the most recent connection done with this curl handle.	CURLINFO_PRIMARY_PORT
PRIVATE	Returns the private string associated with the curl handle (set with the PRIVATE option to CURLSETOPT).	CURLINFO_PRIVATE
PROXYAUTH_AVAIL	Returns a string containing the proxy authentication method(s) available. The meaning of the bits is explained in the PROXYAUTH option for CURLSETOPT	CURLINFO_PROXYAUTH_AVAIL
REDIRECT_COUNT	Returns the total number of redirections that were actually followed.	CURLINFO_REDIRECT_COUNT
REDIRECT_TIME	Returns the total time, in seconds, it took for all redirection steps include name lookup, connect, pretransfer and transfer before final transaction was started. REDIRECT_TIME contains the complete execution time for multiple redirections.	CURLINFO_REDIRECT_TIME
REDIRECT_URL	Returns the URL a redirect would take you to if you would enable **FOLLOWLOCATION** option. This can come very handy if you think using the built-in libcurl redirect logic isn't good enough for you but you would still prefer to avoid implementing all the magic of figuring out the new URL.	CURLINFO_REDIRECT_URL
REQUEST_SIZE	Returns the total size of all requests. Note that this may be for more than one request if	CURLINFO_REQUEST_SIZE

Rexx/CURL Option	Description	cURL Equivalent Option
	FOLLOWLOCATION was set.	
RESPONSE_CODE	Returns the last received HTTP, FTP or SMTP response code. This option was previously known as **HTTP_CODE**. The value will be zero if no server response code has been received. Note that a proxy's CONNECT response should be read with **HTTP_CONNECTCODE** and not this. Support for SMTP responses added in libcurl 7.25.0.	CURLINFO_RESPONSE_CODE
RTSP_CLIENT_CSEQ	Returns the next CSEQ that will be used by the application.	CURLINFO_RTSP_CLIENT_CSEQ
RTSP_CSEQ_RECV	Returns the most recently received CSeq from the server. If your application encounters a **CURLE_RTSP_CSEQ_ERROR** then you may wish to troubleshoot and/or fix the CSeq mismatch by peeking at this value.	CURLINFO_RTSP_CSEQ_RECV
RTSP_SERVER_CSEQ	Returns the next CSEQ that will be expected by the application. Applications wishing to resume an RTSP session on another connection should retreive this info before closing the active connection.	CURLINFO_RTSP_SERVER_CSEQ
RTSP_SESSION_ID	Returns a string holding the most recent RTSP Session ID. Applications wishing to resume an RTSP session on another connection should retreive this info before closing the active connection.	CURLINFO_RTSP_SESSION_ID
SIZE_DOWNLAOD	Returns the total number of bytes downloaded.	CURLINFO_SIZE_DOWNLOAD
SIZE_UPLOAD	Returns the total number of bytes uploaded.	CURLINFO_SIZE_UPLOAD
SPEED_DOWNLOAD	Returns the average download speed in bytes per second.	CURLINFO_SPEED_DOWNLOAD
SPEED_UPLOAD	Returns the average upload speed in bytes per second.	CURLINFO_SPEED_UPLOAD
SSL_ENGINES	Returns the list of supported SSL engines available. The results are returned in the named stem (argument 3). The return value will be the number of entries in the stem.	CURLINFO_SSL_ENGINES
SSL_VERIFY_RESULT	Returns the result of the SSL certificate verification requested by **SSLVERIFYPEER**.	CURLINFO_SSL_VERIFYRESULT
STARTTRANSFER_TIME	Returns the time in seconds it took from the start until the first byte is received by libcurl. This includes **PRETRANSFER_TIME** and also the time the server needs to calculate the result.	CURLINFO_STARTTRANSFER_TIME
TOTAL_TIME		CURLINFO_TOTAL_TIME

Rexx/CURL Option	Description	cURL Equivalent Option
	Returns the time in seconds for the complete transfer.	

Returns:

success:
the value specified above
failure:
blank
On *failure* **CURLERROR.INTCODE** is set to a non-zero value. If this value is 1 (one) then **CURLERROR.CURLCODE** is also set to a non-zero value.

CURLVARIABLE(*variable name*[,*variable value*])

Set or get the value for the specified variable.

The following variables are available in all implementations:

- **VERSION** *(readonly)* the version of Rexx/CURL, consisting of:
  - ♦ *package name* - **rexxcurl**
  - ♦ *Rexx/CURL version* - numerical version; eg. 1.0
  - ♦ *Rexx/CURL date* - Rexx standard date format; eg. 4 Jul 2001
  - ♦ *OS platform* - current operating system
  - ♦ *cURL version* - version of cURL library: **libcurl** *version*

  eg. REXXCURL 1.0 4 Jul 2001 UNIX libcurl 7.8

- **DEBUG** *(setable)* level of debugging requested.
  - ♦ *0* - no debugging information displayed (default)
  - ♦ *1* - Rexx variables displayed as set
    Equivalent to *-v* command line flag.
  - ♦ *2* - function entry/exit information displayed
    Equivalent to *-d* command line flag.
  - ♦ *4* - internal tracing information displayed
    Equivalent to *-D* command line flags.

  Any of the above values may be added together to combine their effects. eg. 6 is equivalent to -dD on the command line.
- **DEBUGFILE** *(setable)* file where debug information is sent.
  Any valid filename. If the file exists it will be overwritten If not specified, all dubugging information gets written to *stderr*.
- **ERROR** *(setable)* prefix for error code variables.
  Any valid Rexx variable name; usually a stem name. The default is **CURLERROR.**
- **CONSTANTPREFIX** *(setable)* prefix for Rexx/CURL variables.
  Any valid Rexx variable name; usually a stem name. The default is **"!REXXCURL.!"**.
- **LISTSETOPT** *(readonly)* list of options supported by CURLSETOPT returned as a string if no optional stem name is supplied or as a Rexx *array*. For deprecated options, the option is listed followed by a colon and the option name that replaces the deprecated option.
- **LISTGETINTO** *(readonly)* list of options supported by CURLGETINFO returned as a string if no optional stem name is supplied or as a Rexx *array*. For deprecated options, the option is listed followed by a colon and the option name that replaces the deprecated option.

Arguments:

variable name
The name of the variable who's value is to be set or retrieved.
variable value
The value that the variable to be set to.

Returns:

with *variable value* specified:
success
blank if a valid *variable name* specified and it is able to be set;
failure
blank and **CURLERROR.INTCODE** set to a non-zero value.

with *variable value* NOT specified:

success
the current value of the variable
failure
blank and **CURLERROR.INTCODE** is set to a non-zero value.

CURLLOADFUNCS()

This function is used to load all the Rexx/CURL external functions. This function is called after the function has been loaded with the Rexx builtin function rxfuncadd().

Although this function is useful only for dynamic library implementations of Rexx/CURL, it can be called by the executable version of Rexx/CURL. In this case it does nothing.

Arguments:

none

Returns:

success:
zero
failure:
non-zero

CURLDROPFUNCS()

This function is used to terminate Rexx/CURL and free up all resources that have been used.

It should be called at the end of every Rexx/CURL program. In particular, this function should be called after a syntax error has been caught with SIGNAL ON SYNTAX.

Arguments:

none

Returns:

success:
zero
failure:
non-zero

4. Errors

The success or failure of each function call is determined by the Rexx compound variable; **CURLERROR.INTCODE**. If the function call succeeds, this will be set to zero. If the function call fails, this will be set to a non-zero value. If the value set is 1 (one), a cURL error occured, and **CURLERROR.CURLCODE** is set to the appropriate error code. Associated with both error code variables, is an equivalent textual error message. These are **CURLERROR.INTERRM** and **CURLERROR.CURLERRM** respectively.
The stem name initially set for the error variables is **CURLERROR.** *(including trailing period)*. You can change this to any value you prefer, with a call to CURLVARIABLE with the **ERROR** argument.

Internal Errors:

```
 1  - Error from cURL
 2  - Invalid Number
 3  - Invalid Option
 4  - Out of memory
 5  - Invalid cURL handle
 6  - Invalid filename
 7  - Invalid boolean
 8  - Expecting a stem as parameter
 9  - Invalid variable name specified
10  - Attempt to set a readonly variable name
11  - Too few arguments supplied
12  - Field must be specified
13  - Error writing to temporary file
```

5. Using Rexx/CURL

A typical Rexx/CURL program looks like the following program. This example will simply display the contents of my home page:

```
        Call RXFuncAdd 'CurlLoadFuncs','rexxcurl','CurlLoadFuncs'
        Call CurlLoadFuncs
        curl = CURLInit()
        If curl \= '' Then
           Do
              Call CURLSetOpt curl, 'URL', "http://www.rexx.org/"
              If curlerror.intcode \= 0 Then Call Abort 'Error setting URL option'
              Call CURLPerform curl
              If curlerror.intcode \= 0 Then Call Abort 'Error performing action'
              Call CURLCleanup curl
           End
        Call CURLDropFuncs
        Return 0
        Abort:
        Parse Arg msg
        Say msg
        If curlerror.intcode = 1 Then Say 'Internal error:' curlerror.intcode curlerror.interrm
        Else Say 'CURL error:' curlerror.curlcode curlerror.curlerrm
        Call CURLCleanup curl
        Exit 1
```

Note that the third parameter to RxFuncAdd is case sensitive, so should always be specified as "CurlLoadFuncs".

Examples
Several example programs are provided with all Rexx/CURL distributions. **getright.rexx URL output_directory**
httppost.rexx your_email_address your_list_password
scp.rexx remote_file_spec local_file_spec
upload.rexx filenames
sendmail-smtp.rexx
getmail-pop3.rexx

- **rexxcurl.rexx**
  Displays the available functions. Used to check package works.
- **testcurl.rexx URL**
  Displays the HTML source of the specified URL.
- **getright.rexx URL output_directory**
  Simplistic implementation of the "GetRight" download utility. Downloads the specified URL to the specified directory. If the file already exists, it will use the RESUME capability of the server (if implemented).
- **httppost.rexx your_email_address your_list_password**
  Suscribes to the rexxcurl mailing list at SourceForge using HTTP POST to fill in an HTML form.
- **scp.rexx remote_file_spec local_file_spec**
  scp.rexx local_file_spec remote_file_spec
  Mimics the SSH program: scp, but only copies to/from an ftp server.
- **upload.rexx filenames**
  Uploads the files specified to the SourceForge "uploads" directory.
- **sendmail-smtp.rexx**
  Sends a simple text email using GMail via SNMP. Change GMail credentials as appropriate.
- **getmail-pop3.rexx**
  Connects to a POP3 server and for each email message displays the From: Subject: and Date: headers. Change email server/account details as appropriate.

History of Rexx/CURL

This section provides details of changes and additions made to the Rexx/CURL interface as it evolves.

Version 2.1.1: 12 May 2024

- Enable building with dynamic loading of API functions rather than static linking
- Change to CURLLoadFuncs() to allow the API dll/so to be specified when dynamic loading
- Windows builds now statically linked with libcurl 8.7.0

Version 2.1.0: 9 Feb 2019

- Addressed bugs: #8, #10, #11, #12, #13
- Implement Support Request: #2
- More options
- Fix HTTPPOSTFIELDS; don't append & for each field unless optional arg is supplied
- Windows builds now with libcurl 7.64.0
- Full support for concurrent installations of Rexx/CURL on Windows for different interpreters and/or 32bit or 64bit. If you have Rexx/CURL already installed, you should manually uninstall it prior to installing this version.

Version 2.0.2: 12 Apr 2012

- Fixes compilation erros with versions of cURL between 7.38.0 and 7.41.0
- Add CurlReset function

Version 2.0: 25 April 2012

- This release aligns with 7.25.0 of cURL
- Added CURLESCAPE and CURLUNESCAPE function for converting URLs
- Add better error message when invalid options are specified
- Display warnings when deprecated (curlsetopt) options are used
- Display warning on startup when using a runtime version of libcurl that is less than the version Rexx/CURL was built with
- Add new constants providing information about libcurl support
- Additional options (curlsetopt) from cURL 7.15.5 to 7.25.0
- Additional options (curlgetinfo) from cURL 7.15.5 to 7.25.0
- Removed SOURCE_* options (curlsetopt) as at cURL 7.16.0
- Changed return format of HTTPAUTH_AVAIL (curlgetinfo)
- OS/2 and eCS now supported
- Rexx/CURL now thread-safe.
- Licensing changed to CPLv1.0

Version 1.6: 10 July 2006 (not released)

- Addition of FTP_ENTRY_PATH (curlgetinfo)
- Bundle 7.15.4 library with Windows installer

Version 1.5: 10 March 2006 (not released)

- Additional options (curlsetopt) from cURL 7.15.2: CONNECTONLY, LOCALPORT, LOCALPORTRANGE
- Additional return information (curlgetinfo) from cURL 7.15.2: LASTSOCKET

Version 1.4: 16 October 2005

- Additional return information (curlgetinfo) from cURL 7.14.0: NUM_CONNECTS, OS_ERRNO, SSL_ENGINES
- Additional return information (curlgetinfo) from cURL 7.14.1: COOKIELIST
- Added CURLFORMADD and CURLFORMFREE to provide more flexibility with HTTP POST options
- Additional options (curlsetopt) from cURL 7.15.0: FTPSKIPPASVIP

Version 1.3: 4 October 2005

- Additional options (curlsetopt) from cURL 7.14.0: FTPSSL, FTPSSLAUTH, SOURCEPOSTQUOTE, SOURCEPREQUOTE, SOURCEQUOTE, SOURCEURL, SOURCEUSERPWD, TCPNODELAY

Version 1.2: 2 December 2004

- Fixed bug with CURLGETINFO and returned string values
- Bring up-to-date with cURL 7.12.1 features
- First callback (PROGRESSFUNCTION) added (only works with Regina)
- Fixed SF bugs 516517 618127

Version 1.1: 27 Jul 2003 (not released)

- Bring up-to-date with cURL 7.10.2 features

Version 1.0: 4 Jul 2001

- Initial release for Unix platforms.

Last updated 10 February 2019