Client URL Library

Introduction

PHP supports libcurl, a library created by Daniel Stenberg, that allows you to connect and communicate to many different types of servers with many different types of protocols. libcurl currently supports the http, https, ftp, gopher, telnet, dict, file, and ldap protocols. libcurl also supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading (this can also be done with PHP's ftp extension), HTTP form based upload, proxies, cookies, and user+password authentication.

These functions have been added in PHP 4.0.2.

Installing/Configuring

Requirements

In order to use PHP's cURL functions you need to install the » libcurl package. PHP requires that you use libcurl 7.0.2-beta or higher. In PHP 4.2.3, you will need libcurl version 7.9.0 or higher. From PHP 4.3.0, you will need a libcurl version that's 7.9.8 or higher. PHP 5.0.0 requires a libcurl version 7.10.5 or greater.

Installation

To use PHP's cURL support you must also compile PHP *--with-curl[=DIR]* where DIR is the location of the directory containing the lib and include directories. In the "include" directory there should be a folder named "curl" which should contain the *easy.h* and *curl.h* files. There should be a file named *libcurl.a* located in the "lib" directory. Beginning with PHP 4.3.0 you can configure PHP to use cURL for URL streams *--with-curlwrappers*.

Note
Note to Win32 Users In order to enable this module on a Windows environment, *libeay32.dll* and *ssleay32.dll* must be present in your PATH. You don't need *libcurl.dll* from the cURL site.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension defines two resource types: a cURL handle and a cURL multi handle.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Descriptions and uses for these constants are described within the curl_setopt() and curl_getinfo() documentation.

CURLOPT_AUTOREFERER (integer)
Available since PHP 5.1.0

CURLOPT_COOKIESESSION (integer)
Available since PHP 5.1.0

CURLOPT_DNS_USE_GLOBAL_CACHE (integer)

CURLOPT_DNS_CACHE_TIMEOUT (integer)

CURLOPT_FTP_SSL (integer)
Available since PHP 5.2.0

CURLFTPSSL_TRY (integer)
Available since PHP 5.2.0

CURLFTPSSL_ALL (integer)
Available since PHP 5.2.0

CURLFTPSSL_CONTROL (integer)
Available since PHP 5.2.0

CURLFTPSSL_NONE (integer)
Available since PHP 5.2.0

CURLOPT_PRIVATE (integer)
Available since PHP 5.2.4

CURLOPT_FTPSSLAUTH (integer)
Available since PHP 5.1.0

CURLOPT_PORT (integer)

CURLOPT_FILE (integer)

CURLOPT_INFILE (integer)

CURLOPT_INFILESIZE (integer)

CURLOPT_URL (integer)

CURLOPT_PROXY (integer)

CURLOPT_VERBOSE (integer)

CURLOPT_HEADER (integer)

CURLOPT_HTTPHEADER (integer)

CURLOPT_NOPROGRESS (integer)

CURLOPT_NOBODY (integer)

CURLOPT_FAILONERROR (integer)

CURLOPT_UPLOAD (integer)

CURLOPT_POST (integer)

CURLOPT_FTPLISTONLY (integer)

CURLOPT_FTPAPPEND (integer)

CURLOPT_FTP_CREATE_MISSING_DIRS (integer)

CURLOPT_NETRC (integer)

CURLOPT_FOLLOWLOCATION (integer)
This constant is not available when open_basedir or safe_mode are enabled.

CURLOPT_FTPASCII (integer)

CURLOPT_PUT (integer)

CURLOPT_MUTE (integer)

CURLOPT_USERPWD (integer)

CURLOPT_PROXYUSERPWD (integer)

CURLOPT_RANGE (integer)

CURLOPT_TIMEOUT (integer)

CURLOPT_TIMEOUT_MS (integer)

CURLOPT_TCP_NODELAY (integer)
Available since PHP 5.2.1

CURLOPT_POSTFIELDS (integer)

CURLOPT_REFERER (integer)

CURLOPT_USERAGENT (integer)

CURLOPT_FTPPORT (integer)

CURLOPT_FTP_USE_EPSV (integer)

CURLOPT_LOW_SPEED_LIMIT (integer)

CURLOPT_LOW_SPEED_TIME (integer)

CURLOPT_RESUME_FROM (integer)

CURLOPT_COOKIE (integer)

CURLOPT_SSLCERT (integer)

CURLOPT_SSLCERTPASSWD (integer)

CURLOPT_WRITEHEADER (integer)

CURLOPT_SSL_VERIFYHOST (integer)

CURLOPT_COOKIEFILE (integer)

CURLOPT_SSLVERSION (integer)

CURLOPT_TIMECONDITION (integer)

CURLOPT_TIMEVALUE (integer)

CURLOPT_CUSTOMREQUEST (integer)

CURLOPT_STDERR (integer)

CURLOPT_TRANSFERTEXT (integer)

CURLOPT_RETURNTRANSFER (integer)

CURLOPT_QUOTE (integer)

CURLOPT_POSTQUOTE (integer)

CURLOPT_INTERFACE (integer)

CURLOPT_KRB4LEVEL (integer)

CURLOPT_HTTPPROXYTUNNEL (integer)

CURLOPT_FILETIME (integer)

CURLOPT_WRITEFUNCTION (integer)

CURLOPT_READFUNCTION (integer)

CURLOPT_PASSWDFUNCTION (integer)

CURLOPT_HEADERFUNCTION (integer)

CURLOPT_MAXREDIRS (integer)

CURLOPT_MAXCONNECTS (integer)

CURLOPT_CLOSEPOLICY (integer)

CURLOPT_FRESH_CONNECT (integer)

CURLOPT_FORBID_REUSE (integer)

CURLOPT_RANDOM_FILE (integer)

CURLOPT_EGDSOCKET (integer)

CURLOPT_CONNECTTIMEOUT (integer)

CURLOPT_CONNECTTIMEOUT_MS (integer)

CURLOPT_SSL_VERIFYPEER (integer)

CURLOPT_CAINFO (integer)

CURLOPT_CAPATH (integer)

CURLOPT_COOKIEJAR (integer)

CURLOPT_SSL_CIPHER_LIST (integer)

CURLOPT_BINARYTRANSFER (integer)

CURLOPT_NOSIGNAL (integer)

CURLOPT_PROXYTYPE (integer)

CURLOPT_BUFFERSIZE (integer)

CURLOPT_HTTPGET (integer)

CURLOPT_HTTP_VERSION (integer)

CURLOPT_SSLKEY (integer)

CURLOPT_SSLKEYTYPE (integer)

CURLOPT_SSLKEYPASSWD (integer)

CURLOPT_SSLENGINE (integer)

CURLOPT_SSLENGINE_DEFAULT (integer)

CURLOPT_SSLCERTTYPE (integer)

CURLOPT_CRLF (integer)

CURLOPT_ENCODING (integer)

CURLOPT_PROXYPORT (integer)

CURLOPT_UNRESTRICTED_AUTH (integer)

CURLOPT_FTP_USE_EPRT (integer)

CURLOPT_HTTP200ALIASES (integer)

CURLOPT_HTTPAUTH (integer)

CURLAUTH_BASIC (integer)

CURLAUTH_DIGEST (integer)

CURLAUTH_GSSNEGOTIATE (integer)

CURLAUTH_NTLM (integer)

CURLAUTH_ANY (integer)

CURLAUTH_ANYSAFE (integer)

CURLOPT_PROXYAUTH (integer)

CURLCLOSEPOLICY_LEAST_RECENTLY_USED (integer)

CURLCLOSEPOLICY_LEAST_TRAFFIC (integer)

CURLCLOSEPOLICY_SLOWEST (integer)

CURLCLOSEPOLICY_CALLBACK (integer)

CURLCLOSEPOLICY_OLDEST (integer)

CURLINFO_PRIVATE (integer)
Available since PHP 5.2.4

CURLINFO_EFFECTIVE_URL (integer)

CURLINFO_HTTP_CODE (integer)

CURLINFO_HEADER_OUT (integer)
Available since PHP 5.1.3

CURLINFO_HEADER_SIZE (integer)

CURLINFO_REQUEST_SIZE (integer)

CURLINFO_TOTAL_TIME (integer)

CURLINFO_NAMELOOKUP_TIME (integer)

CURLINFO_CONNECT_TIME (integer)

CURLINFO_PRETRANSFER_TIME (integer)

CURLINFO_SIZE_UPLOAD (integer)

CURLINFO_SIZE_DOWNLOAD (integer)

CURLINFO_SPEED_DOWNLOAD (integer)

CURLINFO_SPEED_UPLOAD (integer)

CURLINFO_FILETIME (integer)

CURLINFO_SSL_VERIFYRESULT (integer)

CURLINFO_CONTENT_LENGTH_DOWNLOAD (integer)

CURLINFO_CONTENT_LENGTH_UPLOAD (integer)

CURLINFO_STARTTRANSFER_TIME (integer)

CURLINFO_CONTENT_TYPE (integer)

CURLINFO_REDIRECT_TIME (integer)

CURLINFO_REDIRECT_COUNT (integer)

CURL_TIMECOND_IFMODSINCE (integer)

CURL_TIMECOND_IFUNMODSINCE (integer)

CURL_TIMECOND_LASTMOD (integer)

CURL_VERSION_IPV6 (integer)

CURL_VERSION_KERBEROS4 (integer)

CURL_VERSION_SSL (integer)

CURL_VERSION_LIBZ (integer)

CURLVERSION_NOW (integer)

CURLE_OK (integer)

CURLE_UNSUPPORTED_PROTOCOL (integer)

CURLE_FAILED_INIT (integer)

CURLE_URL_MALFORMAT (integer)

CURLE_URL_MALFORMAT_USER (integer)

CURLE_COULDNT_RESOLVE_PROXY (integer)

CURLE_COULDNT_RESOLVE_HOST (integer)

CURLE_COULDNT_CONNECT (integer)

CURLE_FTP_WEIRD_SERVER_REPLY (integer)

CURLE_FTP_ACCESS_DENIED (integer)

CURLE_FTP_USER_PASSWORD_INCORRECT (integer)

CURLE_FTP_WEIRD_PASS_REPLY (integer)

CURLE_FTP_WEIRD_USER_REPLY (integer)

CURLE_FTP_WEIRD_PASV_REPLY (integer)

CURLE_FTP_WEIRD_227_FORMAT (integer)

CURLE_FTP_CANT_GET_HOST (integer)

CURLE_FTP_CANT_RECONNECT (integer)

CURLE_FTP_COULDNT_SET_BINARY (integer)

CURLE_PARTIAL_FILE (integer)

CURLE_FTP_COULDNT_RETR_FILE (integer)

CURLE_FTP_WRITE_ERROR (integer)

CURLE_FTP_QUOTE_ERROR (integer)

CURLE_HTTP_NOT_FOUND (integer)

CURLE_WRITE_ERROR (integer)

CURLE_MALFORMAT_USER (integer)

CURLE_FTP_COULDNT_STOR_FILE (integer)

CURLE_READ_ERROR (integer)

CURLE_OUT_OF_MEMORY (integer)

CURLE_OPERATION_TIMEOUTED (integer)

CURLE_FTP_COULDNT_SET_ASCII (integer)

CURLE_FTP_PORT_FAILED (integer)

CURLE_FTP_COULDNT_USE_REST (integer)

CURLE_FTP_COULDNT_GET_SIZE (integer)

CURLE_HTTP_RANGE_ERROR (integer)

CURLE_HTTP_POST_ERROR (integer)

CURLE_SSL_CONNECT_ERROR (integer)

CURLE_FTP_BAD_DOWNLOAD_RESUME (integer)

CURLE_FILE_COULDNT_READ_FILE (integer)

CURLE_LDAP_CANNOT_BIND (integer)

CURLE_LDAP_SEARCH_FAILED (integer)

CURLE_LIBRARY_NOT_FOUND (integer)

CURLE_FUNCTION_NOT_FOUND (integer)

CURLE_ABORTED_BY_CALLBACK (integer)

CURLE_BAD_FUNCTION_ARGUMENT (integer)

CURLE_BAD_CALLING_ORDER (integer)

CURLE_HTTP_PORT_FAILED (integer)

CURLE_BAD_PASSWORD_ENTERED (integer)

CURLE_TOO_MANY_REDIRECTS (integer)

CURLE_UNKNOWN_TELNET_OPTION (integer)

CURLE_TELNET_OPTION_SYNTAX (integer)

CURLE_OBSOLETE (integer)

CURLE_SSL_PEER_CERTIFICATE (integer)

CURLE_GOT_NOTHING (integer)

CURLE_SSL_ENGINE_NOTFOUND (integer)

CURLE_SSL_ENGINE_SETFAILED (integer)

CURLE_SEND_ERROR (integer)

CURLE_RECV_ERROR (integer)

CURLE_SHARE_IN_USE (integer)

CURLE_SSL_CERTPROBLEM (integer)

CURLE_SSL_CIPHER (integer)

CURLE_SSL_CACERT (integer)

CURLE_BAD_CONTENT_ENCODING (integer)

CURLE_LDAP_INVALID_URL (integer)

CURLE_FILESIZE_EXCEEDED (integer)

CURLE_FTP_SSL_FAILED (integer)

CURLFTPAUTH_DEFAULT (integer)
Available since PHP 5.1.0

CURLFTPAUTH_SSL (integer)

Available since PHP 5.1.0

CURLFTPAUTH_TLS (integer)
Available since PHP 5.1.0

CURLPROXY_HTTP (integer)

CURLPROXY_SOCKS5 (integer)

CURL_NETRC_OPTIONAL (integer)

CURL_NETRC_IGNORED (integer)

CURL_NETRC_REQUIRED (integer)

CURL_HTTP_VERSION_NONE (integer)

CURL_HTTP_VERSION_1_0 (integer)

CURL_HTTP_VERSION_1_1 (integer)

CURLM_CALL_MULTI_PERFORM (integer)

CURLM_OK (integer)

CURLM_BAD_HANDLE (integer)

CURLM_BAD_EASY_HANDLE (integer)

CURLM_OUT_OF_MEMORY (integer)

CURLM_INTERNAL_ERROR (integer)

CURLMSG_DONE (integer)

Examples

Once you've compiled PHP with cURL support, you can begin using the cURL functions. The basic idea behind the cURL functions is that you initialize a cURL session using the curl_init(), then you can set all your options for the transfer via the curl_setopt(), then you can execute the session with the curl_exec() and then you finish off your session using the curl_close(). Here is an example that uses the cURL functions to fetch the example.com homepage into a file:

Example #1 - Using PHP's cURL module to fetch the example.com homepage

```
<?php

$ch = curl_init("http://www.example.com/");
$fp = fopen("example_homepage.txt", "w");

curl_setopt($ch, CURLOPT_FILE, $fp);
curl_setopt($ch, CURLOPT_HEADER, 0);

curl_exec($ch);
curl_close($ch);
fclose($fp);
?>
```

cURL Functions

curl_close

curl_close -- Close a cURL session

Description

void **curl_close** (resource `$ch`)

Closes a cURL session and frees all resources. The cURL handle, *ch*, is also deleted.

Parameters

ch
A cURL handle returned by curl_init().

Return Values

No value is returned.

Examples

Example #2 - Initializing a new cURL session and fetching a web page

```
<?php
// create a new cURL resource
$ch = curl_init();

// set URL and other appropriate options
curl_setopt($ch, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch, CURLOPT_HEADER, 0);

// grab URL and pass it to the browser
curl_exec($ch);

// close cURL resource, and free up system resources
curl_close($ch);
?>
```

See Also

- curl_init()
- curl_multi_close()

curl_copy_handle

curl_copy_handle -- Copy a cURL handle along with all of its preferences

Description

resource **curl_copy_handle** (resource `$ch`)

Copies a cURL handle keeping the same preferences.

Parameters

ch
: A cURL handle returned by curl_init().

Return Values

Returns a new cURL handle.

Examples

Example #3 - Copying a cURL handle

```
<?php
// create a new cURL resource
$ch = curl_init();

// set URL and other appropriate options
curl_setopt($ch, CURLOPT_URL, 'http://www.example.com/');
curl_setopt($ch, CURLOPT_HEADER, 0);

// copy the handle
$ch2 = curl_copy_handle($ch);

// grab URL (http://www.example.com/) and pass it to the browser
curl_exec($ch2);

// close cURL resources, and free up system resources
curl_close($ch2);
curl_close($ch);
?>
```

curl_errno

curl_errno -- Return the last error number

Description

int **curl_errno** (resource `$ch`)

Returns the error number for the last cURL operation.

Parameters

ch
: A cURL handle returned by curl_init().

Return Values

Returns the error number or *0* (zero) if no error occurred.

See Also

- curl_error()
- » Curl error codes

curl_error

curl_error -- Return a string containing the last error for the current session

Description

string **curl_error** (resource `$ch`)

Returns a clear text error message for the last cURL operation.

Parameters

ch
: A cURL handle returned by curl_init().

Return Values

Returns the error number or '' (the empty string) if no error occurred.

See Also

- curl_errno()
- » Curl error codes

curl_exec

curl_exec -- Perform a cURL session

Description

mixed **curl_exec** (resource `$ch`)

Execute the given cURL session.

This function should be called after initializing a cURL session and all the options for the session are set.

Parameters

ch
A cURL handle returned by curl_init().

Return Values

Returns **TRUE** on success or **FALSE** on failure. However, if the **CURLOPT_RETURNTRANSFER** option is set, it will return the result on success, **FALSE** on failure.

Examples

Example #4 - Fetching a web page

```
<?php
// create a new cURL resource
$ch = curl_init();

// set URL and other appropriate options
curl_setopt($ch, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch, CURLOPT_HEADER, 0);

// grab URL and pass it to the browser
curl_exec($ch);

// close cURL resource, and free up system resources
curl_close($ch);
?>
```

See Also

- curl_multi_exec()

curl_getinfo

curl_getinfo -- Get information regarding a specific transfer

Description

mixed **curl_getinfo** (resource `$ch` [, int `$opt`])

Gets information about the last transfer,

Parameters

ch
A cURL handle returned by curl_init().

opt
This may be one of the following constants:

- **CURLINFO_EFFECTIVE_URL** - Last effective URL
- **CURLINFO_HTTP_CODE** - Last received HTTP code
- **CURLINFO_FILETIME** - Remote time of the retrieved document, if -1 is returned the time of the document is unknown
- **CURLINFO_TOTAL_TIME** - Total transaction time in seconds for last transfer
- **CURLINFO_NAMELOOKUP_TIME** - Time in seconds until name resolving was complete
- **CURLINFO_CONNECT_TIME** - Time in seconds it took to establish the connection
- **CURLINFO_PRETRANSFER_TIME** - Time in seconds from start until just before file transfer begins
- **CURLINFO_STARTTRANSFER_TIME** - Time in seconds until the first byte is about to be transferred
- **CURLINFO_REDIRECT_TIME** - Time in seconds of all redirection steps before final transaction was started
- **CURLINFO_SIZE_UPLOAD** - Total number of bytes uploaded
- **CURLINFO_SIZE_DOWNLOAD** - Total number of bytes downloaded
- **CURLINFO_SPEED_DOWNLOAD** - Average download speed
- **CURLINFO_SPEED_UPLOAD** - Average upload speed
- **CURLINFO_HEADER_SIZE** - Total size of all headers received
- **CURLINFO_HEADER_OUT** - The request string sent. Available since PHP 5.1.3
- **CURLINFO_REQUEST_SIZE** - Total size of issued requests, currently only for

HTTP requests

- **CURLINFO_SSL_VERIFYRESULT** - Result of SSL certification verification requested by setting CURLOPT_SSL_VERIFYPEER
- **CURLINFO_CONTENT_LENGTH_DOWNLOAD** - content-length of download, read from Content-Length: field
- **CURLINFO_CONTENT_LENGTH_UPLOAD** - Specified size of upload
- **CURLINFO_CONTENT_TYPE** - Content-type of downloaded object, NULL indicates server did not send valid Content-Type: header

Return Values

If *opt* is given, returns its value as a string. Otherwise, returns an associative array with the following elements (which correspond to *opt*):

- "url"
- "content_type"
- "http_code"
- "header_size"
- "request_size"
- "filetime"
- "ssl_verify_result"
- "redirect_count"
- "total_time"
- "namelookup_time"
- "connect_time"
- "pretransfer_time"
- "size_upload"
- "size_download"
- "speed_download"
- "speed_upload"
- "download_content_length"
- "upload_content_length"
- "starttransfer_time"
- "redirect_time"

curl_init

curl_init -- Initialize a cURL session

Description

resource **curl_init** ([string `$url`])

Initializes a new session and return a cURL handle for use with the curl_setopt(), curl_exec(), and curl_close() functions.

Parameters

`url`
: If provided, the **CURLOPT_URL** option will be set to its value. You can manually set this using the curl_setopt() function.

Return Values

Returns a cURL handle on success, **FALSE** on errors.

Examples

Example #5 - Initializing a new cURL session and fetching a web page

```
<?php
// create a new cURL resource
$ch = curl_init();

// set URL and other appropriate options
curl_setopt($ch, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch, CURLOPT_HEADER, 0);

// grab URL and pass it to the browser
curl_exec($ch);

// close cURL resource, and free up system resources
curl_close($ch);
?>
```

See Also

- curl_close()
- curl_multi_init()

curl_multi_add_handle

curl_multi_add_handle -- Add a normal cURL handle to a cURL multi handle

Description

int **curl_multi_add_handle** (resource `$mh`, resource `$ch`)

Adds the *`ch`* handle to the multi handle *`mh`*

Parameters

`mh`
A cURL multi handle returned by curl_multi_init().

`ch`
A cURL handle returned by curl_init().

Return Values

Returns 0 on success, or one of the **CURLM_XXX** errors code.

Examples

Example #6 - curl_multi_add_handle() example

This example will create two cURL handles, add them to a multi handle, and then run them in parallel.

```
<?php
// create both cURL resources
$ch1 = curl_init();
$ch2 = curl_init();

// set URL and other appropriate options
curl_setopt($ch1, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch1, CURLOPT_HEADER, 0);
curl_setopt($ch2, CURLOPT_URL, "http://www.php.net/");
curl_setopt($ch2, CURLOPT_HEADER, 0);

//create the multiple cURL handle
$mh = curl_multi_init();

//add the two handles
curl_multi_add_handle($mh,$ch1);
curl_multi_add_handle($mh,$ch2);
```

```
$running=null;
//execute the handles
do {
   curl_multi_exec($mh,$running);
} while($running > 0);

//close all the handles
curl_multi_remove_handle($ch1);
curl_multi_remove_handle($ch2);
curl_multi_close($mh);
?>
```

See Also

- curl_multi_remove_handle()
- curl_multi_init()
- curl_init()

curl_multi_close

curl_multi_close -- Close a set of cURL handles

Description

void **curl_multi_close** (resource `$mh`)

Closes a set of cURL handles.

Parameters

mh
A cURL multi handle returned by curl_multi_init().

Return Values

No value is returned.

Examples

Example #7 - curl_multi_close() example

This example will create two cURL handles, add them to a multi handle, and then run them in parallel.

```
<?php
// create both cURL resources
$ch1 = curl_init();
$ch2 = curl_init();

// set URL and other appropriate options
curl_setopt($ch1, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch1, CURLOPT_HEADER, 0);
curl_setopt($ch2, CURLOPT_URL, "http://www.php.net/");
curl_setopt($ch2, CURLOPT_HEADER, 0);

//create the multiple cURL handle
$mh = curl_multi_init();

//add the two handles
curl_multi_add_handle($mh,$ch1);
curl_multi_add_handle($mh,$ch2);

$running=null;
//execute the handles
do {
   curl_multi_exec($mh,$running);
```

```
} while ($running > 0)
//close the handles
curl_multi_remove_handle($ch1);
curl_multi_remove_handle($ch2);
curl_multi_close($mh);

?>
```

See Also

- curl_multi_init()
- curl_close()

curl_multi_exec

curl_multi_exec -- Run the sub-connections of the current cURL handle

Description

int **curl_multi_exec** (resource `$mh`, int `&$still_running`)

Processes each of the handles in the stack. This method can be called whether or not a handle needs to read or write data.

Parameters

`mh`
A cURL multi handle returned by curl_multi_init().

`still_running`
A reference to a flag to tell whether the operations are still running.

Return Values

A cURL code defined in the cURL Predefined Constants.

Note
This only returns errors regarding the whole multi stack. There might still have occurred problems on individual transfers even when this function returns **CURLM_OK**.

Examples

Example #8 - curl_multi_exec() **example**

This example will create two cURL handles, add them to a multi handle, and then run them in parallel.

```
<?php
// create both cURL resources
$ch1 = curl_init();
$ch2 = curl_init();

// set URL and other appropriate options
curl_setopt($ch1, CURLOPT_URL, "http://lxr.php.net/");
curl_setopt($ch1, CURLOPT_HEADER, 0);
```

```
curl_setopt($ch2, CURLOPT_URL, "http://www.php.net/");
curl_setopt($ch2, CURLOPT_HEADER, 0);

//create the multiple cURL handle
$mh = curl_multi_init();

//add the two handles
curl_multi_add_handle($mh,$ch1);
curl_multi_add_handle($mh,$ch2);

$active = null;
//execute the handles
do {
   $mrc = curl_multi_exec($mh, $active);
} while ($mrc == CURLM_CALL_MULTI_PERFORM);

while ($active && $mrc == CURLM_OK) {
   if (curl_multi_select($mh) != -1) {
       do {
           $mrc = curl_multi_exec($mh, $active);
       } while ($mrc == CURLM_CALL_MULTI_PERFORM);
   }
}

//close the handles
curl_multi_remove_handle($mh, $ch1);
curl_multi_remove_handle($mh, $ch2);
curl_multi_close($mh);

?>
```

See Also

- curl_multi_init()
- curl_exec()

curl_multi_getcontent

curl_multi_getcontent -- Return the content of a cURL handle if **CURLOPT_RETURNTRANSFER** is set

Description

string **curl_multi_getcontent** (resource `$ch`)

If **CURLOPT_RETURNTRANSFER** is an option that is set for a specific handle, then this function will return the content of that cURL handle in the form of a string.

Parameters

ch
: A cURL handle returned by curl_init().

Return Values

Return the content of a cURL handle if **CURLOPT_RETURNTRANSFER** is set.

See Also

- curl_multi_init()

curl_multi_info_read

curl_multi_info_read -- Get information about the current transfers

Description

array **curl_multi_info_read** (resource `$mh` [, int `$msgs_in_queue`])

Ask the multi handle if there are any messages/informationals from the individual transfers. Messages may include informationals such as an error code from the transfer or just the fact that a transfer is completed.

Repeated calls to this function will return a new result each time, until a **FALSE** is returned as a signal that there is no more to get at this point. The integer pointed to with *`msgs_in_queue`* will contain the number of remaining messages after this function was called.

Warning
The data the returned resource points to will not survive calling curl_multi_remove_handle().

Parameters

`mh`
A cURL multi handle returned by curl_multi_init().

`msgs_in_queue`
Number of messages that are still in the queue

Return Values

On success, returns an associative array for the message, **FALSE** on failure.

ChangeLog

Version	Description
5.2.0	*`msgs_in_queue`* was added.

See Also

- curl_multi_init()

curl_multi_init

curl_multi_init -- Returns a new cURL multi handle

Description

resource **curl_multi_init** (void)

Allows the processing of multiple cURL handles in parallel.

Parameters

mh
A cURL multi handle returned by curl_multi_init().

Return Values

Returns a cURL on handle on success, **FALSE** on failure.

Examples

Example #9 - curl_multi_init() example

This example will create two cURL handles, add them to a multi handle, and then run them in parallel.

```
<?php
// create both cURL resources
$ch1 = curl_init();
$ch2 = curl_init();

// set URL and other appropriate options
curl_setopt($ch1, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch1, CURLOPT_HEADER, 0);
curl_setopt($ch2, CURLOPT_URL, "http://www.php.net/");
curl_setopt($ch2, CURLOPT_HEADER, 0);

//create the multiple cURL handle
$mh = curl_multi_init();

//add the two handles
curl_multi_add_handle($mh,$ch1);
curl_multi_add_handle($mh,$ch2);

$running=null;
//execute the handles
do {
   curl_multi_exec($mh,$running);
```

```
} while ($running > 0);

//close the handles
curl_multi_remove_handle($ch1);
curl_multi_remove_handle($ch2);
curl_multi_close($mh);

?>
```

See Also

- curl_init()
- curl_multi_close()

curl_multi_remove_handle

curl_multi_remove_handle -- Remove a multi handle from a set of cURL handles

Description

int **curl_multi_remove_handle** (resource `$mh`, resource `$ch`)

Removes a given *`ch`* handle from the given *`mh`* handle. When the *`ch`* handle has been removed, it is again perfectly legal to run curl_exec() on this handle. Removing a handle while being used, will effectively halt all transfers in progress.

Parameters

`mh`
A cURL multi handle returned by curl_multi_init().

`ch`
A cURL handle returned by curl_init().

Return Values

On success, returns a cURL handle, **FALSE** on failure.

See Also

- curl_init()
- curl_multi_init()
- curl_multi_add_handle()

curl_multi_select

curl_multi_select -- Get all the sockets associated with the cURL extension, which can then be "selected"

Description

int **curl_multi_select** (resource `$mh` [, float `$timeout`])

Get all the sockets associated with the cURL extension, which can then be "selected".

Parameters

`mh`
A cURL multi handle returned by curl_multi_init().

`timeout`
Time, in seconds, to wait for a response.

Return Values

On success, returns the number of descriptors contained in, the descriptor sets. On failure, this function will return **FALSE**.

See Also

- curl_multi_init()

curl_setopt_array

curl_setopt_array -- Set multiple options for a cURL transfer

Description

bool **curl_setopt_array** (resource `$ch`, array `$options`)

Sets multiple options for a cURL session. This function is useful for setting a large amount of cURL options without repetitively calling curl_setopt().

Parameters

ch
: A cURL handle returned by curl_init().

options
: An array specifying which options to set and their values. The keys should be valid curl_setopt() constants or their integer equivalents.

Return Values

Returns **TRUE** if all options were successfully set. If an option could not be successfully set, **FALSE** is immediately returned, ignoring any future options in the *options* array.

Examples

Example #10 - Initializing a new cURL session and fetching a web page

```
<?php
// create a new cURL resource
$ch = curl_init();

// set URL and other appropriate options
$options = array(CURLOPT_URL => 'http://www.example.com/',
                 CURLOPT_HEADER => false
                );

curl_setopt_array($ch, $options);

// grab URL and pass it to the browser
curl_exec($ch);

// close cURL resource, and free up system resources
curl_close($ch);
?>
```

Prior to PHP 5.1.4 this function can be simulated with:

Example #11 - Our own implementation of curl_setopt_array()

```
<?php
if (!function_exists('curl_setopt_array')) {
  function curl_setopt_array(&$ch, $curl_options)
  {
      foreach ($curl_options as $option => $value) {
          if (!curl_setopt($ch, $option, $value)) {
              return false;
          }
      }
      return true;
  }
}
?>
```

See Also

- curl_setopt()

curl_setopt

curl_setopt -- Set an option for a cURL transfer

Description

bool **curl_setopt** (resource `$ch`, int `$option`, mixed `$value`)

Sets an option on the given cURL session handle.

Parameters

`ch`
A cURL handle returned by curl_init().

`option`
The *CURLOPT_XXX* option to set.

`value`
The value to be set on *`option`*. *`value`* should be a bool for the following values of the *`option`* parameter:

Option	Set *`value`* to	Notes
CURLOPT_AUTOREFERER	**TRUE** to automatically set the *Referer:* field in requests where it follows a *Location:* redirect.	Available since PHP 5.1.0.
CURLOPT_BINARYTRANSFER	**TRUE** to return the raw output when **CURLOPT_RETURNTRANSFER** is used.	
CURLOPT_COOKIESESSION	**TRUE** to mark this as a new cookie "session". It will force libcurl to ignore all cookies it is about to load that are "session cookies" from the previous session. By default, libcurl always stores and loads all cookies, independent if they are session cookies are not. Session cookies are cookies without expiry date and they are meant to be alive and existing for this "session"	Available since PHP 5.1.0.

	only.	
CURLOPT_CRLF	**TRUE** to convert Unix newlines to CRLF newlines on transfers.	
CURLOPT_DNS_USE_GLOBAL_CACHE	**TRUE** to use a global DNS cache. This option is not thread-safe and is enabled by default.	
CURLOPT_FAILONERROR	**TRUE** to fail silently if the HTTP code returned is greater than or equal to 400. The default behavior is to return the page normally, ignoring the code.	
CURLOPT_FILETIME	**TRUE** to attempt to retrieve the modification date of the remote document. This value can be retrieved using the *`CURLINFO_FILETIME`* option with curl_getinfo().	
CURLOPT_FOLLOWLOCATION	**TRUE** to follow any *"Location: "* header that the server sends as part of the HTTP header (note this is recursive, PHP will follow as many *"Location: "* headers that it is sent, unless **CURLOPT_MAXREDIRS** is set).	
CURLOPT_FORBID_REUSE	**TRUE** to force the connection to explicitly close when it has finished processing, and not be pooled for reuse.	
CURLOPT_FRESH_CONNECT	**TRUE** to force the use of a new connection instead of a cached one.	
CURLOPT_FTP_USE_EPRT	**TRUE** to use EPRT (and LPRT) when doing active FTP downloads. Use **FALSE** to disable EPRT and LPRT and use PORT only.	Added in PHP 5.0.0.
CURLOPT_FTP_USE_EPS	**TRUE** to first try an EPSV	

V	command for FTP transfers before reverting back to PASV. Set to **FALSE** to disable EPSV.	
CURLOPT_FTPAPPEND	**TRUE** to append to the remote file instead of overwriting it.	
CURLOPT_FTPASCII	An alias of **CURLOPT_TRANSFERTEXT**. Use that instead.	
CURLOPT_FTPLISTONLY	**TRUE** to only list the names of an FTP directory.	
CURLOPT_HEADER	**TRUE** to include the header in the output.	
CURLOPT_HTTPGET	**TRUE** to reset the HTTP request method to GET. Since GET is the default, this is only necessary if the request method has been changed.	
CURLOPT_HTTPPROXYTUNNEL	**TRUE** to tunnel through a given HTTP proxy.	
CURLOPT_MUTE	**TRUE** to be completely silent with regards to the cURL functions.	
CURLOPT_NETRC	**TRUE** to scan the *~/.netrc* file to find a username and password for the remote site that a connection is being established with.	
CURLOPT_NOBODY	**TRUE** to exclude the body from the output.	

CURLOPT_NOPROGRESS

TRUE to disable the progress meter for cURL transfers.

Note
PHP automatically sets this option to **TRUE**, this should only be changed

for debugging purposes.

CURLOPT_NOSIGNAL	**TRUE** to ignore any cURL function that causes a signal to be sent to the PHP process. This is turned on by default in multi-threaded SAPIs so timeout options can still be used.	Added in cURL 7.10 and PHP 5.0.0.

CURLOPT_POST	**TRUE** to do a regular HTTP POST. This POST is the normal *application/x-www-form-urlencoded* kind, most commonly used by HTML forms.	
CURLOPT_PUT	**TRUE** to HTTP PUT a file. The file to PUT must be set with **CURLOPT_INFILE** and **CURLOPT_INFILESIZE**.	
CURLOPT_RETURNTRANSFER	**TRUE** to return the transfer as a string of the return value of curl_exec() instead of outputting it out directly.	
CURLOPT_SSL_VERIFYPEER	**FALSE** to stop cURL from verifying the peer's certificate. Alternate certificates to verify against can be specified with the **CURLOPT_CAINFO** option or a certificate directory can be specified with the **CURLOPT_CAPATH** option. **CURLOPT_SSL_VERIFYHOST** may also need to be **TRUE** or **FALSE** if **CURLOPT_SSL_VERIFYPEER** is disabled (it defaults to 2).	**TRUE** by default as of cURL 7.10. Default bundle installed as of cURL 7.10.
CURLOPT_TRANSFERTEXT	**TRUE** to use ASCII mode for FTP transfers. For LDAP, it retrieves data in plain text instead of HTML. On Windows systems, it will not set *STDOUT* to binary mode.	
CURLOPT_UNRESTRICTED_AUTH	**TRUE** to keep sending the username and password when following locations (using **CURLOPT_FOLLOWLOCATION**), even when the hostname has changed.	Added in PHP 5.0.0.
CURLOPT_UPLOAD	**TRUE** to prepare for an upload.	
CURLOPT_VERBOSE	**TRUE** to output verbose	

	information. Writes output to *STDERR*, or the file specified using **CURLOPT_STDERR**.	

value should be an integer for the following values of the *option* parameter:

Option	Set *value* to	Notes
CURLOPT_BUFFERSIZE	The size of the buffer to use for each read. There is no guarantee this request will be fulfilled, however.	Added in cURL 7.10 and PHP 5.0.0.
CURLOPT_CLOSEPOLICY	Either *CURLCLOSEPOLICY_LEAST_RECENTLY_USED* or *CURLCLOSEPOLICY_OLDEST*. There are three other *CURLCLOSEPOLICY_* constants, but cURL does not support them yet.	
CURLOPT_CONNECTTIMEOUT	The number of seconds to wait whilst trying to connect. Use 0 to wait indefinitely.	
CURLOPT_DNS_CACHE_TIMEOUT	The number of seconds to keep DNS entries in memory. This option is set to 120 (2 minutes) by default.	
CURLOPT_FTPSSLAUTH	The FTP authentication method (when is activated): *CURLFTPAUTH_SSL* (try SSL first), *CURLFTPAUTH_TLS* (try TLS first), or *CURLFTPAUTH_DEFAULT* (let cURL decide).	Added in cURL 7.12.2 and PHP 5.1.0.
CURLOPT_HTTP_VERSION	*CURL_HTTP_VERSION_NONE* (default, lets CURL decide which version to use), *CURL_HTTP_VERSION_1_0* (forces HTTP/1.0), or *CURL_HTTP_VERSION_1_1* (forces HTTP/1.1).	
CURLOPT_HTTPAUTH	The HTTP authentication method(s) to use. The options are: *CURLAUTH_BASIC*, *CURLAUTH_DIGEST*,	*Added in PHP 5.0.0.*

	CURLAUTH_GSSNEGOTIATE, *CURLAUTH_NTLM*, *CURLAUTH_ANY*, and *CURLAUTH_ANYSAFE*. The bitwise | (or) operator can be used to combine more than one method. If this is done, cURL will poll the server to see what methods it supports and pick the best one. *CURLAUTH_ANY* is an alias for *CURLAUTH_BASIC* | *CURLAUTH_DIGEST* | *CURLAUTH_GSSNEGOTIATE* | *CURLAUTH_NTLM*. *CURLAUTH_ANYSAFE* is an alias for *CURLAUTH_DIGEST* | *CURLAUTH_GSSNEGOTIATE* | *CURLAUTH_NTLM*.	
CURLOPT_INFILESIZE	The expected size, in bytes, of the file when uploading a file to a remote site.	
CURLOPT_LOW_SPEED_LIMIT	The transfer speed, in bytes per second, that the transfer should be below during **CURLOPT_LOW_SPEED_TIME** seconds for PHP to consider the transfer too slow and abort.	
CURLOPT_LOW_SPEED_TIME	The number of seconds the transfer should be below **CURLOPT_LOW_SPEED_LIMIT** for PHP to consider the transfer too slow and abort.	
CURLOPT_MAXCONNECTS	The maximum amount of persistent connections that are allowed. When the limit is reached, **CURLOPT_CLOSEPOLICY** is used to determine which connection to close.	
CURLOPT_MAXREDIRS	The maximum amount of	

	HTTP redirections to follow. Use this option alongside **CURLOPT_FOLLOWLOCATION**.	
CURLOPT_PORT	An alternative port number to connect to.	
CURLOPT_PROXYAUTH	The HTTP authentication method(s) to use for the proxy connection. Use the same bitmasks as described in **CURLOPT_HTTPAUTH**. For proxy authentication, only *CURLAUTH_BASIC* and *CURLAUTH_NTLM* are currently supported.	Added in cURL 7.10.7 and PHP 5.1.0.
CURLOPT_PROXYPORT	The port number of the proxy to connect to. This port number can also be set in **CURLOPT_PROXY**.	Added in PHP 5.0.0.
CURLOPT_PROXYTYPE	Either *CURLPROXY_HTTP* (default) or *CURLPROXY_SOCKS5*.	Added in cURL 7.10 and PHP 5.0.0.
CURLOPT_RESUME_FROM	The offset, in bytes, to resume a transfer from.	
CURLOPT_SSL_VERIFYHOST	1 to check the existence of a common name in the SSL peer certificate. 2 to check the existence of a common name and also verify that it matches the hostname provided.	
CURLOPT_SSLVERSION	The SSL version (2 or 3) to use. By default PHP will try to determine this itself, although in some cases this must be set manually.	
CURLOPT_TIMECONDITION	How **CURLOPT_TIMEVALUE** is treated. Use *CURL_TIMECOND_IFMODSINCE* to return the page only if it has been modified since the time specified in **CURLOPT_TIMEVALUE**. If	Added in PHP 5.1.0.

	it hasn't been modified, a *"304 Not Modified"* header will be returned assuming **CURLOPT_HEADER** is **TRUE**. Use *CURL_TIMECOND_ISUNMODSINCE* for the reverse effect. *CURL_TIMECOND_IFMODSINCE* is the default.	
CURLOPT_TIMEOUT	The maximum number of seconds to allow cURL functions to execute.	
CURLOPT_TIMEVALUE	The time in seconds since January 1st, 1970. The time will be used by **CURLOPT_TIMECONDITION**. By default, *CURL_TIMECOND_IFMODSINCE* is used.	

value should be a string for the following values of the *option* parameter:

Option	**Set** *value* **to**	**Notes**
CURLOPT_CAINFO	The name of a file holding one or more certificates to verify the peer with. This only makes sense when used in combination with **CURLOPT_SSL_VERIFYPEER**.	
CURLOPT_CAPATH	A directory that holds multiple CA certificates. Use this option alongside **CURLOPT_SSL_VERIFYPEER**.	
CURLOPT_COOKIE	The contents of the *"Set-Cookie: "* header to be used in the HTTP request.	
CURLOPT_COOKIEFILE	The name of the file containing the cookie data. The cookie file can be in Netscape format, or just plain HTTP-style headers dumped into a file.	
CURLOPT_COOKIEJAR	The name of a file to save all	

	internal cookies to when the connection closes.	
CURLOPT_CUSTOMREQUEST	A custom request method to use instead of *"GET"* or *"HEAD"* when doing a HTTP request. This is useful for doing *"DELETE"* or other, more obscure HTTP requests. Valid values are things like *"GET"*, *"POST"*, *"CONNECT"* and so on; i.e. Do not enter a whole HTTP request line here. For instance, entering *"GET /index.html HTTP/1.0\r\n\r\n"* would be incorrect. **Note** Don't do this without making sure the server supports the custom request method first.	
CURLOPT_EGDSOCKET	Like **CURLOPT_RANDOM_FILE**, except a filename to an Entropy Gathering Daemon socket.	
CURLOPT_ENCODING	The contents of the *"Accept-Encoding: "* header. This enables decoding of the response. Supported encodings are *"identity"*, *"deflate"*, and *"gzip"*. If an empty string, *""*, is set, a header containing all supported encoding types is sent.	Added in cURL 7.10.
CURLOPT_FTPPORT	The value which will be used to get the IP address to use for the FTP "POST" instruction. The "POST" instruction tells the remote server to connect to our	

	specified IP address. The string may be a plain IP address, a hostname, a network interface name (under Unix), or just a plain '-' to use the systems default IP address.	
CURLOPT_INTERFACE	The name of the outgoing network interface to use. This can be an interface name, an IP address or a host name.	
CURLOPT_KRB4LEVEL	The KRB4 (Kerberos 4) security level. Any of the following values (in order from least to most powerful) are valid: *"clear"*, *"safe"*, *"confidential"*, *"private"*.. If the string does not match one of these, *"private"* is used. Setting this option to **NULL** will disable KRB4 security. Currently KRB4 security only works with FTP transactions.	
CURLOPT_POSTFIELDS	The full data to post in a HTTP "POST" operation. To post a file, prepend a filename with @ and use the full path.	
CURLOPT_PROXY	The HTTP proxy to tunnel requests through.	
CURLOPT_PROXYUSERPWD	A username and password formatted as *"[username]:[password]"* to use for the connection to the proxy.	
CURLOPT_RANDOM_FILE	A filename to be used to seed the random number generator for SSL.	
CURLOPT_RANGE	Range(s) of data to retrieve in the format *"X-Y"* where X or Y are optional. HTTP transfers also support several intervals, separated	

	with commas in the format *"X-Y,N-M"*.	
CURLOPT_REFERER	The contents of the *"Referer: "* header to be used in a HTTP request.	
CURLOPT_SSL_CIPHER_LIST	A list of ciphers to use for SSL. For example, *RC4-SHA* and *TLSv1* are valid cipher lists.	
CURLOPT_SSLCERT	The name of a file containing a PEM formatted certificate.	
CURLOPT_SSLCERTPASSWD	The password required to use the **CURLOPT_SSLCERT** certificate.	
CURLOPT_SSLCERTTYPE	The format of the certificate. Supported formats are *"PEM"* (default), *"DER"*, and *"ENG"*.	Added in cURL 7.9.3 and PHP 5.0.0.
CURLOPT_SSLENGINE	The identifier for the crypto engine of the private SSL key specified in **CURLOPT_SSLKEY**.	
CURLOPT_SSLENGINE_DEFAULT	The identifier for the crypto engine used for asymmetric crypto operations.	
CURLOPT_SSLKEY	The name of a file containing a private SSL key.	
CURLOPT_SSLKEYPASSWD	The secret password needed to use the private SSL key specified in **CURLOPT_SSLKEY**. **Note** Since this option contains a sensitive password, remember to keep the PHP script it is contained within safe.	

CURLOPT_SSLKEYTYPE	The key type of the private SSL key specified in **CURLOPT_SSLKEY**. Supported key types are *"PEM"* (default), *"DER"*, and *"ENG"*.	
CURLOPT_URL	The URL to fetch. This can also be set when initializing a session with curl_init().	
CURLOPT_USERAGENT	The contents of the *"User-Agent: "* header to be used in a HTTP request.	
CURLOPT_USERPWD	A username and password formatted as *"[username]:[password]"* to use for the connection.	

value should be an array for the following values of the *option* parameter:

Option	Set *value* to	Notes
CURLOPT_HTTP200ALIASES	An array of HTTP 200 responses that will be treated as valid responses and not as errors.	Added in cURL 7.10.3 and PHP 5.0.0.
CURLOPT_HTTPHEADER	An array of HTTP header fields to set.	
CURLOPT_POSTQUOTE	An array of FTP commands to execute on the server after the FTP request has been performed.	
CURLOPT_QUOTE	An array of FTP commands to execute on the server prior to the FTP request.	

value should be a stream resource (using fopen(), for example) for the following values of the *option* parameter:

Option	Set *value* to	Notes
CURLOPT_FILE	The file that the transfer should be written to. The default is *STDOUT* (the browser window).	
CURLOPT_INFILE	The file that the transfer	

	should be read from when uploading.	
CURLOPT_STDERR	An alternative location to output errors to instead of *STDERR*.	
CURLOPT_WRITEHEADER	The file that the header part of the transfer is written to.	

value should be a string that is the name of a valid callback function for the following values of the *option* parameter:

Option	**Set *value* to**	**Notes**
CURLOPT_HEADERFUNCTION	The name of a callback function where the callback function takes two parameters. The first is the cURL resource, the second is a string with the header data to be written. The header data must be written when using this callback function. Return the number of bytes written.	
CURLOPT_PASSWDFUNCTION	The name of a callback function where the callback function takes three parameters. The first is the cURL resource, the second is a string containing a password prompt, and the third is the maximum password length. Return the string containing the password.	
CURLOPT_READFUNCTION	The name of a callback function where the callback function takes two parameters. The first is the cURL resource, and the second is a string with the data to be read. The data must be read by using this callback function. Return the number of bytes read. Return 0 to signal *EOF*.	
CURLOPT_WRITEFUNCTI	**The name of a callback**	

ON	function where the callback function takes two parameters. The first is the cURL resource, and the second is a string with the data to be written. The data must be written by using this callback function. Must return the exact number of bytes written or this will fail.	

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #12 - Initializing a new cURL session and fetching a web page

```
<?php
// create a new cURL resource
$ch = curl_init();

// set URL and other appropriate options
curl_setopt($ch, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch, CURLOPT_HEADER, false);

// grab URL and pass it to the browser
curl_exec($ch);

// close cURL resource, and free up system resources
curl_close($ch);
?>
```

Example #13 - Uploading file

```
<?php

/* http://localhost/upload.php:
print_r($_POST);
print_r($_FILES);
*/

$ch = curl_init();

$data = array('name' => 'Foo', 'file' => '@/home/user/test.png');
```

```
curl_setopt($ch, CURLOPT_URL, 'http://localhost/upload.php');
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $data);

curl_exec($ch);
?>
```

The above example will output:

```
Array
(
   [name] => Foo
)
Array
(
   [file] => Array
       (
           [name] => test.png
           [type] => image/png
           [tmp_name] => /tmp/phpcpjNeQ
           [error] => 0
           [size] => 279
       )

)
```

See Also

- curl_setopt_array()

curl_version

curl_version -- Gets cURL version information

Description

array **curl_version** ([int `$age`])

Returns information about the cURL version.

Parameters

age

Return Values

Returns an associative array with the following elements:

version_number	cURL 24 bit version number
version	cURL version number, as a string
ssl_version_number	OpenSSL 24 bit version number
ssl_version	OpenSSL version number, as a string
libz_version	zlib version number, as a string
host	Information about the host where cURL was built
age	
features	A bitmask of the *CURL_VERSION_XXX* constants
protocols	An array of protocols names supported by cURL