MODULE_CURL

Older versions of NetCurl was not "chained" as default. All calls made had to be separately made, instead of X->CALL()chainedCall().

Content of this page

This page

Sub documents

All examples on this page are based on the unit tests delivered with the module.

The module name is as the name reveals based on the curl library. But it also supports other web-drivers. Besides of this, this module has a small driver for handling basic SOAP-calls. The class itself is called Tornevall_SimpleSoap, and is initialized automatically when using http-calls with "?wsdl" in the URL. If you think that the tiny parameter "?wsdl" is useless, enforcing the SoapClient during the web-call is possible by using **NETCURL_POST_DATATYPES::DATATYPE_JSON**, instead of DATATYPE_JSON or similar. More about this, will come.

Deprecation of class name Tornevall_cURL

As of v6.0.20, netcurl is splitted up in sections and classes instead of having all source code in one bundled file. The name Tornevall_cURL has been renamed and is from this version deprecated. The new name of all classes is based on MODULE_<name>, meaning Tornevall_cURL is now known as MODULE_CURL. Most of the components are backwards compatible, so using for example CURL_POST_AS instead of the new class NETCURL_POST_DATATYPES still works.

Module initialization

The easiest way to get started with the module is with composer:

```
# Install Composer
curl -sS https://getcomposer.org/installer | php
# Initialize components
php composer.phar require tornevall/tornelib-php-netcurl
```

Then add a code block to your php-script:

```
<?php
require_once('vendor/autoload.php');
use \TorneLIB\MODULE_CURL;
```

The last usage-row is added to simplify the instance creation of the library. Instead of running

```
new \TorneLIB\MODULE_CURL();
```

this makes it possible to just run a

```
new MODULE_CURL();
```

Note that if you initialize the module with no constructor arguments, you can set it up more detailed afterwards. The module was, before the chain implementation, built in a way that you in a default state never needed to configure anything, but configuration was available if requirements changed. For example, it was possible to initialize the module and then run more calls after each other, with different settings (i.e. different outgoing ip-interfaces, when there was a need to spread the calls between many hosts). Yes; this is basically a "going under the radar"-function, with spoofability.

Primary calls

To make a call to a website, NetCurl provides some basic methods that does the hard work for you. Depending on what you need to do (GET data, POST data, etc), the methods are called following:

Method	Type	Parameters	Description	Notes
doGet	GET	Goes with a simple URL		
doPost	POST	URL, postData, postAs	Makes a POST call with including data, if provided. Included data can be both strings and arrays. If the data is something else than a string, NetCurl tries to translate the data into a proper format. The format is a bit dependent on the postAs-Parameter. Default is CURL_POST_AS::POST_AS_NORMAL. This mode will try to post data in a to-websites-proper-formatting, for example &a=standard&post=format.	**Available POST-formats** &default=value&post=format JSON SOAP
doDelete	DELETE	URL, postData, postAs	Makes a DELETE call, with including data. See POST.	
doPut	PUT	URL, postData, postAs	Makes a PUT call, with including data. See POST.	

Quick start: Oneliners

Running with the latest version of NetCurl makes it possible to quick-fire a web-session, with only one row of code.

```
$getIpByJson = (new \TorneLIB\MODULE_CURL('http://identifier.tornevall.net/?json'))->getParsed();
```

The response of the request should look like something like this:

```
stdClass Object
(
    [ip] => 127.0.0.1
    [host] => localhost
    [HTTP_HOST] => identifier.tornevall.net
    [HTTP_USER_AGENT] => Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.0.3705; .NET CLR
1.1.4322; Media Center PC 4.0; +TorneLIB-NetCurl-6.0.17 +TorneLIB+cUrl-6.0.17)
    [HTTP_ACCEPT] => */*
    [HTTP_EXPECT] => 100-continue
)
```

Authentications

With NetCurl 6.0.18 and newer versions, initializing the library with authenication values is also possible. However, currently, the best supported authentication is CURLAUTH_BASIC (NETCURL_AUTH_TYPES::AUTHTYPE_BASIC), with failover on CURLAUTH_ANY. To use authentication at construction level, one can use something like this:

```
$myWebRequest = (new MODULE_CURL("https://test.com/get/this/info/with/my/personal/account", array(),
NETCURL_POST_METHODS::METHOD_GET, array('auth'=>array('username'=>'username', 'password'=>'password'))))-
>getParsed();
```

Without this oneliner, the call can also be done with the following code:

```
$regularRequest = new MODULE_CURL();
$regularRequest->setAuthentication("username", "password");
$chainRequest = $regularRequest->doGet("https://test.com/get/this/info/with/my/personal/account")->getParsed();
```