Getting started: Vital modules

NetCURL 6.1 is built as individual modules, so it is possible to communicate directly with each of them separately. If you are sure that curl for example is always available, regardless of what platform you put your code, the CurlWrapper can be used instead of NetWrapper.

The unittests that was built initially, was built in a way so each module could be tested separately. It was in the second step the MODULE_CURL was written, but with the new name NetWrapper - which is the module that is self selecting "best available driver".

Sections

Configurables and requirements

In netcurl 6.0 all of the data communication was basically running through MODULE_CURL and then, that data was passed over to a proper communications driver. Basically, curl was the only reliable driver, but with a failsafe selection to components that was installable via composer.

- When curl was not found and drivers from Wordpress or Guzzle became available, the module could fail over to those drivers. At that next lever, both Wordpress and Guzzle handled the php-internal streamdriver.
- If the webrequest was based on SOAP, netcurl automatically chose to run through SoapClient, if the conditions allowed it (SoapClient and some XML-drivers had to be installed).

In netcurl 6.1, NetWrapper (MODULE_CURL), almost do the same as above but it is configured to handle more by itself. There is also a register-function (undocumented) that allows developers to do their own module. However, from 6.1, netcurl tries to cover the most important parts itself.

- Curl is the primary selected driver.
- When curl is not available it fails over to SimpleStreamWrapper. This wrapper requires that allow_url_fopen is true, in php.ini
- When a Soapcall is requested, we still use an internal SoapWrapper. However, it is as of 6.1.0 called SoapClientWrapper instead of SimpleSoap
- When a network request is being made and the content type indicates RSS, netcurl is trying to utilize the RssWrapper. If you set composer to install Laminas RSS-parser, it tries to utilize Laminas before SimpleXML-drivers.

Own drivers

To configure anything, this meant you have to run all setups through that module. As 6.1 is converted to fit PSR4-requirements each module is practically independent. There are external components however, that can (and is) injectable in the modules, depending on how they are written. It will be possible to write own drivers, that takes advantage of the same modules.

Configuration

The drivers should initially not require any preconfiguration like for other modules where you have to write your communication drivers from scratch. This is the biggest thing with netcurl, as it tries to pick any available driver from the system, parse requested data and give it back as formatted as possible with defaults from WrapperConfig - see below.

Preconfigured modules

There are a few modules that you'd want to be aware of. Here's a list of them and how to use them.

Wrappers and drivers

Lonewolf Driver

This is actually not a driver. As the documentation is built a bit at the time as development goes forward (taking a look at Developer notes and secrets may help a lot when you're lost as this is the primary notes-to-self section), you could very much getting starting without the documentation if you prefer to follow the phpunits by looking here or here. That's the lonewolf method in short.

Generic Requests

All wrappers is built around a request method. In version 6.0, each request method has its own method: doGet for GET-requests, doPost for POST-requests, doDelete for DELETE, and so on. In 6.1 we stay with the request method. NetWrapper and MODULE_CURL however supports the old do-methods, even if it is highly recommended to not use those methods.

Basic Drivers

TorneLIB\Module\Config\WrapperConfig

Link: **git-docs**

The major configuration module. **Almost** every feature and defaults that is used within (at least) the curl extension are configured here. WrapperConfig is very much the heart of the module as much configurations are processed through this class. Both streamoptions, curloptions and user-agent setups.

TorneLIB\Module\Network\NetWrapper

Link: **git-docs**

Replacement for MODULE_CURL. If you don't want to make the calls yourself, you'd use this. The request methods (doGet, doPost, doPut, doDelete, etc) have been replaced with request().

TorneLIB\Module\Network\Wrappers\CurlWrapper

Link: **git-docs**

The main curlwrapper. If nothing else is defined and curl is installed, this is the driver that will be used for "almost everything".

TorneLIB\Module\Network\Wrappers\SimpleStreamWrapper

Link: **git-docs**

The failover wrapper, uses the binary safe file_get_contents (and not the fopen/fread/etc). If curl is unavailable, this is where your requests will land.

TorneLIB\Module\Network\Wrappers\SoapClientWrapper

Link: **git-docs**

TorneLIB\Helpers\GenericParser

Link: **git-docs**

Prior location was TorneLIB\Module\Config\GenericParser. This is a generic parser used for parsing data in the wrappers that is commonly used by more than one wrapper.

For example, header extractions is being made here (when you for example need to get information about the http-head response). It also takes care of content-type based parsing from the body, or whatever needs to be properly parsed. In netcurl 6.0, most of the data transmitted in the interfaces was running guessing games which may have caused problems in the way that the IO-parser genereated its own error handler to catch XML errors. This is completely removed.

The others

TorneLIB\Module\Config\WrapperConstants

Link: **git-docs**

Internal constants requested by netcurl, if necessary. In 6.1.0, there are no preset constants.

TorneLIB\Module\Config\WrapperDriver

Link: git-docs

This driver can be instantiated but should normally not do this. This part of netcurls keeps track of available communication drivers in the system, and is also used for third parties in case they really need to register their own wrapper. Registrations always goes this way and is basically what NetWrapper (the real replacement for MODULE_CURL) uses to select what to use when calling for it.

TorneLIB\Module\Config\WrapperCurlOpt

Link: git-docs

Protective layer for CURLOPT constants, in case they are not installed via curl. Instead of netcurl screaming out warnings of missing constants, we use those.

TorneLIB\Module\Config\WrapperSSL

Link: git-docs

SSL is a quite big part of the communication drivers, so the SSL configurator has been separated from the standard WrapperConfig. This is however not very used by the curlwrapper. It rather aims at the stream based communications like the built in fetchers from PHP, and SoapClient.

TorneLIB\Module\Config\WrapperDriver

Link: git-docs

Basically the wrapper container used globally in netcurl, to figure out which communication drivers that is available. This one is also used to configure your own drivers if you miss any.