

CURL STklos extension

Erick Galesio

Table of Contents

1. Introduction.....	1
2. The (stklos curl) library.....	3
2.1. Low level primitives.....	3

Chapter 1. Introduction

This extension gives access to the functions of the *libcurl* library, the multiprotocol file transfer library which comes with the famous [cURL package](#).

When the STklos cURL extension library is installed, it will be available as a normal **STklos** library. It can be imported with the following stance:

```
(import (stklos curl))
```

Hereafter is a simple function which uses the **curl** library:

```
(define (print-content url)
  (let ((handle (curl-init)))           ; ①
    (curl-set-option handle #:url url) ; ②
    (curl-set-option handle #:verbose #t) ; ③
    (curl-perform handle)               ; ④
    (curl-cleanup handle)))             ; ⑤
```

Here, we have:

- ① create a new CURL handle. This function must be called before any connection.
- ② add option to the current handle. The `#:url` option used here tells curl the URL of the text we want to grab. The libcurl uses a long list of options that can be passed here. This is the only option that must be set, as otherwise there can be no transfer.
- ③ Add another option (be verbose) for this transfer. Any number of option can be added before starting data transfer.
- ④ Perform the data transfer as described by the previous options.
- ⑤ `curl-cleanup` must be called to end the CURL session and free the resources used by `libcurl` for the transer



For more information on the available options, see [curl_easy_setopt manual page](#). To use an option, such as `CURLOPT_XYZ`, you can use the keyword `#:xyz` or `#:XYZ` (case doesn't matter here in fact).

Note that not all the options are supported for now. Only the options which use a string or an integer as parameter are handled by the `curl-set-option` function.

Calling `(print-content "https://example.com")` permits to display the HTML content of the site

<https://example.com>.

To grab the content of a given site, you can use the primitive `get-content-as-string` which is described below.

Chapter 2. The (stklos curl) library

2.1. Low level primitives

This section describes the exported symbols of the (stklos curl) library which are (more or less) a direct port of the primitives of the [Easy Interface](#) of *libcurl*. The main primitives which fall in this category are the ones used in [previous example](#).

STklos procedure

(curl-version)

Returns the version of libcurl as a string

```
(curl-version)
=> "libcurl/7.81.0 OpenSSL/1.1.1m zlib/1.2.11 brotli/1.0.9 zstd/1.5.2 libidn2/2.3.2
libpsl/0.21.1 (+libidn2/2.3.0) libssh2/1.10.0 nghttp2/1.46.0"
```

STklos procedure

(curl-init)

This function must be the first function to call, and it returns a CURL handle that you must use as input to other functions in the *libcurl* interface. This call **MUST** have a corresponding call to `curl-clean` when the operation is complete.

STklos procedure

(curl-set-option handle key value)

This function permits to tell libcurl how it must do the transfer. It takes an `handle` obtained by a previous call to `curl-init`, a keyword `key` specifying the option that must be set and the value `val` that must be given to this option. The type of `val` depends of the options to be set (see the [libcurl](#)

[documentation](#) for complete list settable options).



For now, the current version of the library can only set *libcurl* options which accepts a string or an integer (a long for *libcurl*). Boolean values can also be used for integer parameters (they are converted to 0 and 1). As a special case; Scheme ports can also be passed with the options on ports explained below.

Passing ports to `curl-set-option`:

By default, `libcurl` uses the system standard ports for its IO. They can be changed with special options which cannot easily be used in Scheme. As a consequence, `curl-set-option` has been extended to accept the following keywords:

- `#:iport` can be used to set the Scheme port where the input must be read.
- `#:oport` can be used to set the Scheme port where the output must be written.
- `#:eport` can be used to set the Scheme port where the error messages must be written.

For instance, the following piece of code permits to grab the content of <https://example.com> as a string:

```
(import (stklos curl))

(let ((out (open-output-string))
      (handle (curl-init)))
  (curl-set-option handle #:url "https://example.com")
  (curl-set-option handle #:oport out)      ;; place the output in the 'out' string port
  (curl-perform handle)
  (curl-cleanup handle)
  (get-output-string out))
```

STklos procedure

```
(curl-perform handle)
```

Performs the transfer as described in the options given before with `curl-set-option` on the given CURL `handle`. The entire request is done and returns when done, or earlier if it fails.

STklos procedure

```
(curl-cleanup handle)
```

Closes the current session started with th `handle` obtained with a previous `curl-init`. It closes all the

connections kept open by `handle`.



The *libcurl* documentation says: *Do not call this function if you intend to transfer more files, re-using handles is a key to good performance with libcurl.*