

# The lualibs package

Élie Roux · [elie.roux@telecom-bretagne.eu](mailto:elie.roux@telecom-bretagne.eu)  
Philipp Gesang · [philipp.gesang@alumni.uni-heidelberg.de](mailto:philipp.gesang@alumni.uni-heidelberg.de)

2013/05/18 v2.00

## Abstract

Additional Lua functions taken from the `l-*` and `util-*` files of Con<sub>T</sub>E<sub>X</sub>t. For an introduction on this package (among others), please refer to the document `lualatex-doc.pdf`.

## Contents

I	Package Description	1
1	Overview	1
2	Usage	2
2.1	Loading Library Collections	2
2.2	Loading libraries Individually	2
3	Files	2
4	Packaging	3
II	<code>lualibs.lua</code>	4
III	<code>lualibs-basic.lua</code>	7
IV	<code>lualibs-extended.lua</code>	8

## I Package Description

### 1 OVERVIEW

Lua is a very minimal language, and it does only have a minimal standard library. The aim of this package is to provide an extended standard library, to be used by various Lua<sub>T</sub>E<sub>X</sub> packages. The code is specific to Lua<sub>T</sub>E<sub>X</sub> and depends on Lua<sub>T</sub>E<sub>X</sub> functions and modules not available in regular lua.

The code is derived from Con<sub>T</sub>E<sub>X</sub>t libraries.

## 2 USAGE

You can either load the `lualibs` module, which will in turn load one of two sets of libraries provided by this package: `require("lualibs")`, or explicitly load the modules you need, e.g.: `require("lualibs-table")`, please note that some modules depend on others.

### 2.1 *Loading Library Collections*

The libraries are split into a `basic` and an `extended` collection. Though neither initialization time nor memory consumption will be noticeably impacted,<sup>1</sup> the `lualibs` package can skip loading of the latter on demand. The `config` table needs to be present prior to loading the package for this to work (in the future this may be achieved by an option of `\usepackage`) for  $\text{\LaTeX}$  users). In the `lualibs` field, set `load_extended` to `false`:

```
\directlua{
  --- 1) create config table safely
  config          = config or { }
  config.lualibs   = config.lualibs or { }

  --- 2) unset the load_extended entry
  config.lualibs.load_extended = false

  --- 3) load the lualibs
  require "lualibs"
}
```

However, there is no guarantee that the `extended` set remains permanently excluded. Re-loading the package at a later point will cancel this option and possibly add the missing libraries.

### 2.2 *Loading libraries Individually*

In order to load the `table` module you would actually have to prepare it as follows:

```
require"lualibs-lua"
require"lualibs-lpeg"
require"lualibs-table"
```

If your code is run by the `texlua` interpreter, you will need to initialize `kpse` library so that `require()` can find files under `TEXMF` tree: `kpse.set_program_name("luatex")`.

## 3 FILES

The `lualibs` bundle contains files from two Con $\text{\TeX}$ t Lua library categories: The generic auxiliary functions (original file prefix: `1-`) together form something close to a standard

---

<sup>1</sup> Note that in terms of code this is only a small fraction of what Con $\text{\TeX}$ t loads with every run.

---

Table 1: The basic set.

lua <sub>libs</sub> name	ConT <sub>E</sub> Xt name	primary purpose
lua <sub>libs</sub> -lua.lua	l-lua.lua	compatibility
lua <sub>libs</sub> -package.lua	l-package.lua	Lua file loaders
lua <sub>libs</sub> -lpeg.lua	l-lpeg.lua	patterns
lua <sub>libs</sub> -function.lua	l-function.lua	defines a dummy function
lua <sub>libs</sub> -string.lua	l-string.lua	string manipulation
lua <sub>libs</sub> -table.lua	l-table.lua	serialization, conversion
lua <sub>libs</sub> -boolean.lua	l-boolean.lua	boolean converter
lua <sub>libs</sub> -number.lua	l-number.lua	bit operations
lua <sub>libs</sub> -math.lua	l-math.lua	math functions
lua <sub>libs</sub> -io.lua	l-io.lua	reading and writing files
lua <sub>libs</sub> -os.lua	l-os.lua	platform specific code
lua <sub>libs</sub> -file.lua	l-file.lua	filesystem operations
lua <sub>libs</sub> -md5.lua	l-md5.lua	checksum functions
lua <sub>libs</sub> -dir.lua	l-dir.lua	directory handling
lua <sub>libs</sub> -unicode.lua	l-unicode.lua	utf and unicode
lua <sub>libs</sub> -url.lua	l-url.lua	url handling
lua <sub>libs</sub> -set.lua	l-set.lua	sets

---

library. Most of these are extensions of an existing namespace, like for instance `l-table.lua` which adds full-fledged serialization capabilities to the Lua table library. They were imported under the `lualibs-` prefix and are contained in the basic collection. (For a list see table 1.)

The extended category comprises a selection of files mostly from the utilities namespace (`util-`; cf. table 2). Their purpose is more specific and at times quite low-level. Additionally, the file `trac-inf.lua` has been included because it is essential to some of the code loaded subsequently.

#### 4 PACKAGING

By default, `lualibs` will not load the libraries individually. Instead, it includes two *merged packages* that have been compiled from the original files. This is achieved by means of `mtx-package`, a script for bundling Lua code shipped with ConT<sub>E</sub>Xt. This concatenates the code of several Lua files into a single file that is both easier to distribute and loading (marginally) faster. `mtx-package` ensures that the code from each file gets its own closure and strips newlines and comments, resulting in a smaller payload. Another package that relies on it heavily is the font loader as contained in `luaotfload` and `luatex-fonts`.

If ConT<sub>E</sub>Xt is installed on the system, the merge files can be created by running:

```
mtxrun --script package --merge lualibs-basic.lua
mtxrun --script package --merge lualibs-extended.lua
```

---

Table 2: The extended set.

lua <sub>libs</sub> name	ConT <sub>E</sub> Xt name	primary purpose
lua <sub>libs</sub> -util-str.lua	util-str.lua	extra string functions
lua <sub>libs</sub> -util-tab.lua	util-tab.lua	extra table functions
lua <sub>libs</sub> -util-sto.lua	util-sto.lua	table allocation
lua <sub>libs</sub> -util-prs.lua	util-sto.lua	miscellaneous parsers
lua <sub>libs</sub> -util-dim.lua	util-dim.lua	conversion between dimensions
lua <sub>libs</sub> -trac-inf.lua	trac-inf.lua	timing, statistics
lua <sub>libs</sub> -util-lua.lua	util-lua.lua	operations on bytecode
lua <sub>libs</sub> -util-deb.lua	util-deb.lua	extra debug functionality
lua <sub>libs</sub> -util-tpl.lua	util-tpl.lua	templating
lua <sub>libs</sub> -util-sta.lua	util-sta.lua	stacker (e. g. for PDF)
lua <sub>libs</sub> -util-env.lua	util-env.lua	argv handling
lua <sub>libs</sub> -util-jsn.lua	util-jsn.lua	conversion to and from json

---

Of course there is a make target for that:

```
make merge
```

will take care of assembling the packages from the files distributed with lua<sub>libs</sub>.

For this to work, the syntax of the Lua file needs to be well-formed: files that should be merged must be included via a function `loadmodule()`. It doesn't matter if the function actually does something; a dummy will suffice. Also, the argument to `loadmodule()` must be wrapped in parentheses. This rule is quite convenient, actually, since it allows excluding files from the merge while still using `loadmodule()` consistently.

```
...
loadmodule("my-lua-file.lua") -- <= will be merged
loadmodule('my-2nd-file.lua') -- <= will be merged
loadmodule "my-3rd-file.lua"  -- <= will be ignored
...
```

## II lua<sub>libs</sub>.lua

```
1 lualibs = lualibs or { }
2
3 lualibs.module_info = {
4   name      = "lualibs",
5   version   = 2.00,
6   date      = "2013/05/18",
7   description = "ConTEXt Lua standard libraries.",
```

```

8  author      = "Hans Hagen, PRAGMA-ADE, Hasselt NL & Elie Roux & Philipp Gesang",
9  copyright   = "PRAGMA ADE / ConTeXt Development Team",
10 license     = "See ConTeXt's mreadme.pdf for the license",
11 }
12

```

The behavior of the `lualibs` can be configured to some extent.

- Based on the parameter `lualibs.prefer_merged`, the libraries can be loaded via the included merged packages or the individual files.
- Two classes of libraries are distinguished, mainly because of a similar distinction in ConTeXt, but also to make loading of the less fundamental functionality optional. While the “basic” collection is always loaded, the configuration setting `lualibs.load_extended` triggers inclusion of the extended collection.
- Verbosity can be increased via the `verbose` switch.

```

13
14 config      = config or { }
15 config.lualibs = config.lualibs or { }
16
17 lualibs.prefer_merged = config.lualibs.prefer_merged or true
18 lualibs.load_extended = config.lualibs.load_extended or true
19 config.lualibs.verbose = config.lualibs.verbose or false
20

```

The `lualibs` may be loaded in scripts. To account for the different environment, fallbacks for the `luatexbase` facilities are installed.

```

21
22 local dofile      = dofile
23 local kpsefind_file = kpse.find_file
24 local stringformat = string.format
25 local texiowrite_nl = texio.write_nl
26
27 local find_file, error, warn, info
28 do
29   local _error, _warn, _info
30   if luatexbase and luatexbase.provides_module then
31     _error, _warn, _info = luatexbase.provides_module(lualibs.module_info)
32   else
33     _error, _warn, _info = texiowrite_nl, texiowrite_nl, texiowrite_nl
34   end
35
36   if lualibs.verbose then
37     error, warn, info = _error, _warn, _info
38   else
39     local dummylogger = function ( ) end
40     error, warn, info = _error, dummylogger, dummylogger
41   end
42   lualibs.error, lualibs.warn, lualibs.info = error, warn, info
43 end

```

```

44
45 if luatexbase and luatexbase.find_file then
46   find_file = luatexbase.find_file
47 else
48   kpse.set_program_name"luatex"
49   find_file = kpsefind_file
50 end
51

```

The lualibs load a merged package by default. In order to create one of these, the meta file that includes the libraries must satisfy certain assumptions `mtx-package` makes about the coding style. Most important is that the functions that indicates which files to include must go by the name `loadmodule()`. For this reason we define a `loadmodule()` function as a wrapper around `dofile()`.

```

52
53 local loadmodule = loadmodule or function (name, t)
54   if not t then t = "library" end
55   local filepath = find_file(name, "lua")
56   if not filepath or filepath == "" then
57     warn(stringformat("Could not locate %s \"%s\".", t, name))
58     return false
59   end
60   dofile(filepath)
61   return true
62 end
63
64 lualibs.loadmodule = loadmodule
65

```

The separation of the “basic” from the “extended” sets coincides with the split into `luat-bas.mkiv` and `luat-lib.mkiv`.

```

66
67 if lualibs.basic_loaded      ~= true
68 or config.lualibs.force_reload == true
69 then
70   loadmodule"lualibs-basic.lua"
71   loadmodule"lualibs-compat.lua" --- restore stuff gone since v1.*
72 end
73
74 if lualibs.load_extended    == true
75 and lualibs.extended_loaded  ~= true
76 or config.lualibs.force_reload == true
77 then
78   loadmodule"lualibs-extended.lua"
79 end
80
81 --- This restores the default of loading everything should a package
82 --- have requested otherwise. Will be gone once there is a canonical
83 --- interface for parameterized loading of libraries.
84 config.lualibs.load_extended = true

```

```

85
86 -- vim:tw=71:sw=2:ts=2:expandtab
87

```

### III **lua<sub>libs</sub>-basic.lua**

```

1 lualibs                = lualibs or { }
2 local info              = lualibs.info
3 local loadmodule        = lualibs.loadmodule
4
5 local lualibs_basic_module = {
6   name          = "lualibs-basic",
7   version       = 2.00,
8   date          = "2013/05/18",
9   description    = "ConTeXt Lua libraries -- basic collection.",
10  author         = "Hans Hagen, PRAGMA-ADE, Hasselt NL & Elie Roux & Philipp Gesang",
11  copyright      = "PRAGMA ADE / ConTeXt Development Team",
12  license        = "See ConTeXt's mreadme.pdf for the license",
13 }
14
15 local loaded = false --- track success of package loading
16
17 if lualibs.prefer_merged then
18   info"Loading merged package for collection "basic"."
19   loaded = loadmodule('lualibs-basic-merged.lua')
20 else
21   info"Ignoring merged packages."
22   info"Falling back to individual libraries from collection "basic"."
23 end
24

```

mtx-package expects the files to be included by loadmodule. If run on this file, it will create lua<sub>libs</sub>-basic-merged.lua from all the files mentioned in the next block.

```

25
26 if loaded == false then
27   loadmodule("lualibs-lua.lua")
28   loadmodule("lualibs-package.lua")
29   loadmodule("lualibs-lpeg.lua")
30   loadmodule("lualibs-function.lua")
31   loadmodule("lualibs-string.lua")
32   loadmodule("lualibs-table.lua")
33   loadmodule("lualibs-boolean.lua")
34   loadmodule("lualibs-number.lua")
35   loadmodule("lualibs-math.lua")
36   loadmodule("lualibs-io.lua")
37   loadmodule("lualibs-os.lua")
38   loadmodule("lualibs-file.lua")
39   loadmodule("lualibs-md5.lua")
40   loadmodule("lualibs-dir.lua")

```

```

41 loadmodule("lualibs-unicode.lua")
42 loadmodule("lualibs-url.lua")
43 loadmodule("lualibs-set.lua")
44 end
45
46 lualibs.basic_loaded = true
47 -- vim:tw=71:sw=2:ts=2:expandtab
48

```

#### IV lualibs-extended.lua

```

1 lualibs = lualibs or { }
2

```

Loading the *extended* set requires a tad more effort, but it's well invested.

Since we only want the functionality, we have to simulate parts of a running ConT<sub>E</sub>Xt environment, above all logging, that some of the more involved libraries cannot be loaded without. Also, one utility file cannot be packaged because it returns a table which would preclude loading of later code. Thus, we remove it from the natural loading chain (it is not critical) and append it at the end.

```

3
4 local lualibs_extended_module = {
5   name      = "lualibs-extended",
6   version   = 2.00,
7   date      = "2013/05/18",
8   description = "ConTEXt Lua libraries -- extended collection.",
9   author    = "Hans Hagen, PRAGMA-ADE, Hasselt NL & Elie Roux & Philipp Gesang",
10  copyright  = "PRAGMA ADE / ConTEXt Development Team",
11  license    = "See ConTEXt's mreadme.pdf for the license",
12 }
13
14
15 local stringformat = string.format
16 local loadmodule   = lualibs.loadmodule
17 local texiowrite   = texio.write
18 local texiowrite_nl = texio.write_nl
19

```

Here we define some functions that fake the elaborate logging/tracking mechanism Context provides.

```

20
21 local error, logger, mklog
22 if luatexbase and luatexbase.provides_module then
23   --- TODO test how those work out when running tex
24   local __error, __, __logger =
25     luatexbase.provides_module(lualibs_extended_module)
26   error = __error
27   logger = __logger
28   mklog = function ( ) return logger end

```



```

29 else
30   mklog = function (t)
31     local prefix = stringformat("[%s] ", t)
32     return function (...)
33       texiowrite_nl(prefix)
34       texiowrite (stringformat(...))
35     end
36   end
37   error = mklog"ERROR"
38   logger = mklog"INFO"
39 end
40
41 local info = luaLlibs.info
42

```

We temporarily put our own global table in place and restore whatever we overloaded afterwards.

ConT<sub>E</sub>Xt modules each have a custom logging mechanism that can be enabled for debugging. In order to fake the presence of this facility we need to define at least the function `logs.reporter`. For now it's sufficient to make it a reference to `mklog` as defined above.

```

43
44 local dummy_function = function ( ) end
45 local newline        = function ( ) texiowrite_nl"" end
46
47 local fake_logs = function (name)
48   return {
49     name      = name,
50     enable    = dummy_function,
51     disable   = dummy_function,
52     reporter  = mklog,
53     newline   = newline
54   }
55 end
56
57 local fake_trackers = function (name)
58   return {
59     name      = name,
60     enable    = dummy_function,
61     disable   = dummy_function,
62     register  = mklog,
63     newline   = newline,
64   }
65 end
66

```

Among the libraries loaded is `util-env.lua`, which adds ConT<sub>E</sub>Xt's own, superior command line argument handler. Packages that rely on their own handling of arguments might not be aware of this, or the library might have been loaded by another package altogether. For these cases we provide a copy of the original `arg` list and restore it after

we are done loading.

```
67
68 local backup_store = { }
69
70 local fake_context = function ( )
71   if logs      then backup_store.logs      = logs      end
72   if trackers then backup_store.trackers = trackers end
73   logs        = fake_logs"logs"
74   trackers     = fake_trackers"trackers"
75
76   backup_store.argv = table.fastcopy(arg)
77 end
78
79
```

Restore a backed up logger if appropriate.

```
80 local unfake_context = function ( )
81   if backup_store then
82     local bl, bt = backup_store.logs, backup_store.trackers
83     local argv   = backup_store.argv
84     if bl then logs      = bl      end
85     if bt then trackers = bt      end
86     if argv then arg     = argv    end
87   end
88 end
89
90 fake_context()
91
92 local loaded = false
93 if lualibs.prefer_merged then
94   info"Loading merged package for collection "extended"."
95   loaded = loadmodule('lualibs-extended-merged.lua')
96 else
97   info"Ignoring merged packages."
98   info"Falling back to individual libraries from collection "extended"."
99 end
100
101 if loaded == false then
102   loadmodule("lualibs-util-str.lua")--- string formatters (fast)
103   loadmodule("lualibs-util-tab.lua")--- extended table operations
104   loadmodule("lualibs-util-sto.lua")--- storage (hash allocation)
105   -----("lualibs-util-pck.lua")---!packers; necessary?
106   -----("lualibs-util-seq.lua")---!sequencers (function chaining)
107   -----("lualibs-util-mrg.lua")---!only relevant in mtx-package
108   loadmodule("lualibs-util-prs.lua")--- miscellaneous parsers; cool. cool cool cool
109   -----("lualibs-util-fmt.lua")---!column formatter (rarely used)
110   loadmodule("lualibs-util-dim.lua")--- conversions between dimensions
111   loadmodule("lualibs-util-jsn.lua")--- JSON parser
112
113   -----("lualibs-trac-set.lua")---!generalization of trackers
```

```

114 -----("lua-libs-trac-log.lua")---!logging
115 loadmodule("lua-libs-trac-inf.lua")--- timing/statistics
116 loadmodule("lua-libs-util-lua.lua")--- operations on lua bytecode
117 loadmodule("lua-libs-util-deb.lua")--- extra debugging
118 loadmodule("lua-libs-util-tpl.lua")--- templating
119 loadmodule("lua-libs-util-sta.lua")--- stacker (for writing pdf)
120 -----!data-* -- Context specific
121 -----("lua-libs-util-lib.lua")---!swiglib; there is a luatex-swiglib
122 loadmodule("lua-libs-util-env.lua")--- environment arguments
123 -----("lua-libs-mult-ini.lua")---
124 -----("lua-libs-core-con.lua")---
125 end
126
127 unfake_context() --- TODO check if this works at runtime
128
129 lua-libs.extended_loaded = true
130 -- vim:tw=71:sw=2:ts=2:expandtab
131

```