

# The luaotfload package

Elie Roux

`elie.roux@telecom-bretagne.eu`

2009/11/15 v1.05

## Abstract

ConT<sub>E</sub>Xt font loading system, providing the possibility to load OTF fonts with a lot of features, and the XeT<sub>E</sub>X font loading syntax.

## Contents

<b>1</b>	<b>Documentation</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	ConT <sub>E</sub> Xt files needed . . . . .	2
1.3	Troubleshooting . . . . .	3
<b>2</b>	<b>luaotfload.lua</b>	<b>3</b>
<b>3</b>	<b>luaotfload.sty</b>	<b>7</b>

## 1 Documentation

### 1.1 Introduction

Font management and installation has always been painful with T<sub>E</sub>X (and even more with L<sup>A</sup>T<sub>E</sub>X). A lot of files are needed for one font (tfm, pfb, map, fd, vf), and they are limited to 256 characters. But the font world has evolved since, and new standard types of fonts have appeared, like **truetype** or **opentype** fonts. These fonts can contain a lot of characters, and have some functionalities (ligatures, old-style numbers, small capitals, etc.). They are everywhere, as the system fonts and most modern text softwares fonts are of this type. Until now the (almost) only way to use them with T<sub>E</sub>X was to use them with XeT<sub>E</sub>X.

Unlike XeT<sub>E</sub>X, LuaT<sub>E</sub>X does not provide facilities for these fonts by default, but it provides a way to hook lua code in some points of the T<sub>E</sub>X algorithm, for instance we can improve the font loading system; this is what we do in this package.

This package is quite low-level, and should be loaded directly in the macro package, like it is in ConT<sub>E</sub>Xt. Sadly, Plain and L<sup>A</sup>T<sub>E</sub>X are frozen and it's even impossible to adapt them to the new engines.

## 1.2 ConTeXt files needed

This package is a wrapper for several files taken from the ConTeXt macro package. The philosophy is to let ConTeXt do all the implementation and update these files from time to time. To do so we did not modify the files taken from ConTeXt, we only changed their names to prevent name clashes. You can thus update the font system of this package simply by updating the files taken from ConTeXt, without (theoretically) changing the `.sty` file nor the main `.lua` file.

The ConTeXt files are renamed by adding the prefix `otfl-` to them (`otfl` as `OTF Load`). The files are:

- `luat-dum.lua`
- `data-con.lua`
- `node-ini.lua`
- `node-inj.lua`
- `node-fnt.lua`
- `node-dum.lua`
- `font-ini.lua`
- `font-tfm.lua`
- `font-cid.lua`
- `font-ott.lua`
- `font-otf.lua`
- `font-otd.lua`
- `font-oti.lua`
- `font-otb.lua`
- `font-otn.lua`
- `font-ota.lua`
- `font-otc.lua`
- `font-def.lua`
- `font-ctx.lua`
- `font-map.lua`
- `font-dum.lua`

### 1.3 Troubleshooting

If you encounter problems with some fonts, please first update to the latest version of this package before reporting a bug, as this package is under active development.

A very common problem is the lack of features for some otf fonts even when specified. It can be related to the fact that some fonts do not provide features for the `df1t` script, which is the default one in this package, so you may have to specify the script in the command line, for example:

```
\font\myfont = MyFont.otf:script=latn;+liga;
```

## 2 luaotfload.lua

First some usual initializations.

```
1 luaotfload      = { }
2
3 luaotfload.module = {
4   name          = "luaotfload",
5   version       = 1.04,
6   date          = "2009/11/15",
7   description    = "ConTeXt font loading system.",
8   author        = "Elie Roux & Hans Hagen",
9   copyright     = "Elie Roux",
10  license       = "CC0"
11 }
12
13 luatextra.provides_module(luaotfload.module)
14
```

We load the ConTeXt files with this function. It automatically adds the `otfl-` prefix to it, so that we call it with the actual ConTeXt name.

```
15
16
17 function luaotfload.loadmodule(name)
18   local foundname = kpse.find_file('otfl-'..name,"tex")
19   if not foundname then
20     luatextra.module_error('luaotfload', string.format('file otfl-%s not found.', name))
21     return
22   end
23   dofile(foundname)
24 end
25
```

The following functions are made to map ConTeXt functions to luaextra functions.

```
26
27 string.strip = string.stripspaces
28
29 file = fpath
```

```
30 file.extname = fpath.suffix
```

```
31
```

These are small functions that are not already in luatextra.

```
32
```

```
33 local splitters_s, splitters_m = { }, { }
```

```
34
```

```
35 function lpeg.splitat(separator, single)
```

```
36     local splitter = (single and splitters_s[separator]) or splitters_m[separator]
```

```
37     if not splitter then
```

```
38         separator = lpeg.P(separator)
```

```
39         if single then
```

```
40             local other, any = lpeg.C((1 - separator)^0), lpeg.P(1)
```

```
41             splitter = other * (separator * lpeg.C(any^0) + "")
```

```
42             splitters_s[separator] = splitter
```

```
43         else
```

```
44             local other = lpeg.C((1 - separator)^0)
```

```
45             splitter = other * (separator * other)^0
```

```
46             splitters_m[separator] = splitter
```

```
47         end
```

```
48     end
```

```
49     return splitter
```

```
50 end
```

```
51
```

```
52 function table.compact(t)
```

```
53     if t then
```

```
54         for k,v in next, t do
```

```
55             if not next(v) then
```

```
56                 t[k] = nil
```

```
57             end
```

```
58         end
```

```
59     end
```

```
60 end
```

```
61
```

```
62 function table.sortedhashkeys(tab) -- fast one
```

```
63     local srt = { }
```

```
64     for key,_ in next, tab do
```

```
65         srt[#srt+1] = key
```

```
66     end
```

```
67     table.sort(srt)
```

```
68     return srt
```

```
69 end
```

```
70
```

```
71 function table.reverse_hash(h)
```

```
72     local r = { }
```

```
73     for k,v in next, h do
```

```
74         r[v] = string.lower(string.gsub(k, " ", ""))
```

```
75     end
```

```
76     return r
```

```
77 end
```

```

78
79 function table.reverse(t)
80     local tt = { }
81     if #t > 0 then
82         for i=#t,1,-1 do
83             tt[#tt+1] = t[i]
84         end
85     end
86     return tt
87 end
88

```

We start loading some lua files. These two are some code not used by ConTeXt at all that allow other modules to be used, it provides some low-level ConTeXt functions.

```

89
90 luaotfload.loadmodule('luat-dum.lua') -- not used in context at all
91 luaotfload.loadmodule('data-con.lua') -- maybe some day we don't need this one
92

```

This one is for node support.

```

93
94 luaotfload.loadmodule('node-ini.lua')
95

```

By default ConTeXt takes some private attributes for internal use. With Plain and L<sup>A</sup>T<sub>E</sub>X we can't do so, we use `\newluaattribute`. This functions overrides a function defined in the previous module that returns the number of a private attribute. We allocate new attributes in the `.sty` file, and this function returns their number. Like this we don't need any private attribute, and this package is compatible with the others. We use the `otfl@` prefix for attributes.

```

96
97 function attributes.private(name)
98     local number = tex.attributenum[otfl@'..name]
99     if not number then
100         luatextra.module_error('luaotfload', string.format('asking for attribute %s, but not', name))
101     end
102     return number
103 end
104

```

Some more modules. We don't load neither `font-enc.lua` nor `font-afm.lua` as it will never be used here.

```

105
106 luaotfload.loadmodule('node-res.lua')
107 luaotfload.loadmodule('node-inj.lua')
108 luaotfload.loadmodule('node-fnt.lua')
109 luaotfload.loadmodule('node-dum.lua')
110

```

```

111 luaotfload.loadmodule('font-ini.lua')
112 luaotfload.loadmodule('font-tfm.lua')
113 luaotfload.loadmodule('font-cid.lua')
114 luaotfload.loadmodule('font-ott.lua')
115 luaotfload.loadmodule('font-otf.lua')
116 luaotfload.loadmodule('font-otd.lua')
117 luaotfload.loadmodule('font-oti.lua')
118 luaotfload.loadmodule('font-otb.lua')
119 luaotfload.loadmodule('font-otn.lua')
120 luaotfload.loadmodule('font-ota.lua')
121 luaotfload.loadmodule('font-otc.lua')
122

```

`font-def.lua` calls the function `callback.register` to register its callbacks. We override it with a dumb function so that it does not register any callback. We will register the callbacks later.

```

123
124 do
125   local temp = callback.register
126   callback.register = function (...)
127     return
128   end
129   luaotfload.loadmodule('font-def.lua')
130   callback.register = temp
131 end
132
133 luaotfload.loadmodule('font-ctx.lua')
134 luaotfload.loadmodule('font-map.lua')
135 luaotfload.loadmodule('font-dum.lua')
136

```

This is a small patch that prevents errors in some  $\text{\LaTeX}$  files.

```

137
138 fonts.enc.known = {}
139

```

We have to register a function in the `find_vf_file` callback in order to make everything work.

```

140
141 function luaotfload.find_vf_file(name)
142   name = file.removesuffix(file.basename(name))
143   local result = kpse.find_file(name, "vf") or ""
144   if result == "" then
145     result = kpse.find_file(name, "ovf") or ""
146   end
147   return result
148 end
149

```

Finally two functions

```

150
151 function luaotfload.register_callbacks()
152     callback.add('pre_linebreak_filter', nodes.simple_font_handler, 'luaotfload.pre_linebreak_filter')
153     callback.add('hpack_filter', nodes.simple_font_handler, 'luaotfload.hpack_filter')
154     callback.reset('define_font')
155     callback.add('define_font', fonts.define.read, 'luaotfload.define_font', 1)
156     callback.add('find_vf_file', luaotfload.find_vf_file, 'luaotfload.find_vf_file')
157 end
158
159 function luaotfload.unregister_callbacks()
160     callback.remove('pre_linebreak_filter', 'luaotfload.pre_linebreak_filter')
161     callback.remove('hpack_filter', 'luaotfload.hpack_filter')
162     callback.remove('define_font', 'luaotfload.define_font')
163     callback.remove('find_vf_file', 'luaotfload.find_vf_file')
164 end

```

### 3 luaotfload.sty

Classical Plain+L<sup>A</sup>T<sub>E</sub>X package initialization.

```

165 \csname ifluaotfloadloaded\endcsname
166 \let\ifluaotfloadloaded\endinput
167
168 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
169     \input luatextra.sty
170 \else
171     \NeedsTeXFormat{LaTeX2e}
172     \ProvidesPackage{luaotfload}%
173         [2009/11/15 v1.05 ConTeXt font loading system]
174     \RequirePackage{luatextra}
175 \fi
176
177 \expandafter\edef\csname otfl@AtEnd\endcsname{%
178     \catcode64 \the\catcode64\relax
179 }
180
181 \catcode64 11
182

```

The attributes are allocated here. The `otfl@` prefix is added to prevent name collision.

```

183
184 \newluatexattribute\otfl@state
185 \newluatexattribute\otfl@markbase
186 \newluatexattribute\otfl@markdone
187 \newluatexattribute\otfl@markmark
188 \newluatexattribute\otfl@cursbase
189 \newluatexattribute\otfl@curscurs
190 \newluatexattribute\otfl@cursdone

```

```

191 \newluatexattribute\otfl@kernpair
192 \newluatexattribute\otfl@color
193

```

Two small macros to register or unregister the callbacks. Without the callbacks this package is totally turned off.

```

194
195 \def\otfl@off{
196 \luadirect{luaotfload.unregister_callbacks()}
197 }
198
199 \def\otfl@on{
200 \luadirect{luaotfload.register_callbacks()}
201 }
202

```

We load the lua file, and we turn the package on.

```

203
204 \luatexUseModule{luaotfload}
205
206 \otfl@on
207
208 \otfl@AtEnd

```