

# The `hyphen.cfg` file for Lua<sub>T</sub><sub>E</sub>X

Khaled Hosny, Élie Roux, and Manuel Pégourié-Gonnard

khaledhosny@eglug.org

elie.roux@telecom-bretagne.eu

mpg@elzevir.fr

2010/04/28 v1.3beta

## Abstract

This is a modified version of the file `hyphen.cfg` distributed with the `babel` package, with a supporting Lua module, aimed at adapting `babel`'s hyphenation patterns loading mechanism to Lua<sub>T</sub><sub>E</sub>X's dynamic pattern loading capabilities. It makes use of a `language.dat.lua` file (whose format is described below) that should be present in the distribution, in addition to the regular `language.dat` file.

There is a version of `etex.src` modified for the same reasons using similar code, which also makes use of the `luatex-hyphen.lua` and `language.dat.lua` files described here.

## 1 Documentation

Hyphenation patterns should be loaded at runtime with Lua<sub>T</sub><sub>E</sub>X: if they appear in the format, they will be rehashed when the format is loaded anyway, which makes the format quite long to load (many seconds even on modern machines) and provides for bad user experience. Hence, it is desirable to load as few patterns as possible in the format, and load on-demand the needed patterns at runtime.

This package provides a modified version of `hyphen.cfg` adapted to Lua<sub>T</sub><sub>E</sub>X, as well as a supporting Lua module. Since a lot of things, especially the catcodes, are not as predictable at runtime than at format creation time, we don't `\input` the usual pattern files, but rather load the patterns using the Lua interface, using a special plain text version of the pattern files if available.

The existence and file name of such a version cannot be guessed, so we need a specific database: the file `language.dat.lua`. This file should be loadable by Lua and return a table whose keys are the canonical language names as found in `language.dat`, and the values are Lua tables consisting of:

1. A fixed part with two fields:

```
loader = <string> name of the TeX loader
synonyms = { <string> alternative name, ... }
```

Those field's values must be the same as in `language.dat`. The `loader` field is currently unused.

2. A variable part consisting of either:

- For most languages:

```
patterns = <string> filename for patterns
hyphenation = <string> filename for exceptions
lefthyphenmin = <number> value for \letfhyphenmin
righthyphenmin = <number> value for \letfhyphenmin
```

The files given by `patterns` (resp. `hyphenation`) must be plain text files encoded in utf8, with only patterns (resp. exceptions) and not even comments: their content will be used directly without being parsed by  $\text{\TeX}$ . If one of these keys is missing or is the empty string, it is ignored and no patterns (resp. exceptions) are loaded for this language. The values of `*hyphenmin` are currently unused.

- Special case are supported by a field `special`. Currently, the following kind of values are recognized:

`'null'` for languages with no hyphenation patterns nor exceptions. (Note that this is equivalent to both `hyphenation` and `patterns` being `nil` or `''`, but produces a more explicit message in the log.)

`'disabled:<reason>'` allows to disable specific languages: when the user tries to load this language, an error will be issued, with the `<reason>`.

`'language0'` only `english` should use this type of special, to indicate it is normally dumped in the format as `\language0` (see below).

Special languages may have `*hyphenmin` information when it makes sense (mostly `\language0`).

Languages that are mentioned in `language.dat` but not in `language.dat.lua` will be loaded in the format. So, if the `language.dat.lua` file is missing or incomplete, languages will just go back to the “old” behaviour, resulting in longer startup time, which seems less bad than complete breakage.

For backward compatibility, Knuth's original patterns for US English are always loaded in the format, as `\language0`.<sup>1</sup>

The modified version of `hyphen.cfg` provided here checks for the engine, and should continue to work with any engine without any modified behaviour. However, it is recommended to install it in such a way that the original `hyphen.cfg` from `babel` is found first by any engine other than `Lua $\text{\TeX}$` .

## 2 Implementation

### 2.1 luatex-hyphen.lua

---

<sup>1</sup>It is assumed to be the first entry in `language.dat`.

```

1  $\langle$ *lua $\rangle$ 

   Start a Lua module, importing only the necessary functions as locals.
2 local error, dofile, pairs, ipairs = error, dofile, pairs, ipairs
3 local io, texio, lang, kpse = io, texio, lang, kpse
4 module('luatexhyphen')

   Two functions for error and information reporting.
5 local function wlog(msg, ...)
6     texio.write_nl('log', 'luatex-hyphen: '..msg:format(...))
7 end
8 local function err(msg, ...)
9     error('luatex-hyphen: '..msg:format(...), 2)
10 end

   Load the language.dat.lua file with the Lua version of the language database.
11 local dbname = "language.dat.lua"
12 local language_dat
13 local dbfile = kpse.find_file(dbname)
14 if not dbfile then
15     err("file not found: "..dbname)
16 else
17     language_dat = dofile(dbfile)
18 end

   Look up a language in the database, and return the associated information, as
   well as the canonical name of the language.
19 function lookupname(name)
20     if language_dat[name] then
21         return language_dat[name], name
22     else
23         for canon, data in pairs(language_dat) do
24             for _,syn in ipairs(data.synonyms) do
25                 if syn == name then
26                     return data, canon
27                 end
28             end
29         end
30     end
31 end

   Set hyphenation patterns and exceptions for a language given by its name (in
   the database) and number (value of \language). Doesn't return anything, but
   will call error() if things go wrong.
32 function loadlanguage(lname, id)
33     local msg = "loading%s patterns and exceptions for: %s (\\language%d)"

   Lookup the language in the database.
34     local ldata, cname = lookupname(lname)
35     if not ldata then
36         err("no entry in %s for this language: %s", dbname, lname)
37     end

```

```

Handle special languages.
38   if ldata.special then
39       if ldata.special == 'null' then
40           wlog(msg, ' (null)', cname, id)
41           return
42       elseif ldata.special:find('^disabled:') then
43           err("language disabled by %s: %s (%s)", dbname, cname,
44               ldata.special:gsub('^disabled:', ''))
45       elseif ldata.special == 'language0' then
46           err("\\language0 should be dumped in the format")
47       else
48           err("bad entry in %s for language %s")
49       end
50   end

The generic case: load hyphenation patterns and exceptions from files given
by the language code.
51   wlog(msg, '', cname, id)
52   for _, item in ipairs{'hyphenation', 'patterns'} do
53       local file = ldata[item]
54       if file ~= nil and file ~= '' then
55           local file = kpse.find_file(file) or err("file not found: %s", file)
56           local fh = io.open(file, 'r')
57           local data = fh:read('*a') or err("file not readable: %s", f)
58           fh:close()
59           lang[item](lang.new(id), data)
60       end
61   end
62 end
63 </lua>

```

## 2.2 hyphen.cfg

```

64 <*hyphen>

Start with unmodified code from babel.
65 \ifx\ProvidesFile\@undefined
66 \def\ProvidesFile#1[#2 #3 #4]{%
67 \wlog{File: #1 #4 #3 <#2>}%

Use a modified banner for LuaTeX.
68 \ifx\directlua\@undefined
69 \toks8{Babel <#3> and hyphenation patterns for }%
70 \else
71 \toks8{LuaTeX adaptation of babel <#3>
72 and hyphenation patterns for }%
73 \fi

74 \let\ProvidesFile\@undefined
75 }
76 \def\ProvidesLanguage#1[#2 #3 #4]{%
77 \wlog{Language: #1 #4 #3 <#2>}%

```

```

78     }
79 \else
80   \let\bbl@tempa\ProvidesFile
81   \def\ProvidesFile#1[#2 #3 #4]{%
      Same here.
82     \ifx\directlua\undefined
83       \toks8{Babel <#3> and hyphenation patterns for }%
84     \else
85       \toks8{LuaTeX adaptation of babel <#3>
86         and hyphenation patterns for }%
87     \fi
88     \bbl@tempa#1[#2 #3 #4]%
89     \let\ProvidesFile\bbl@tempa}
90 \def\ProvidesLanguage#1{%
91   \begingroup
92     \catcode'\ 10 %
93     \@makeother\/%
94     \@ifnextchar[%]
95       {\@provideslanguage{#1}}{\@provideslanguage{#1}[]}}
96 \def\@provideslanguage#1[#2]{%
97   \wlog{Language: #1 #2}%
98   \expandafter\xdef\csname ver@#1.ldf\endcsname{#2}%
99   \endgroup}
100 \fi
101

```

File identification is modified again.

```

102 \ProvidesFile{hyphen.cfg}
103       [2010/04/26 v3.81-luatex-1.3beta %
104       Language switching mechanism for LuaTeX, adapted from babel v3.81]
105 \ifx\AtBeginDocument\undefined
106   \input plain.def\relax
107 \fi
108 \ifx\language\undefined
109   \csname newcount\endcsname\language
110 \fi
111 \ifx\newlanguage\undefined
112   \csname newcount\endcsname\last@language
113 \else
114   \countdef\last@language=19
115 \fi
116 \ifx\newlanguage\undefined
117   \def\addlanguage#1{%
118     \global\advance\last@language \@ne
119     \ifnum\last@language<\ccclvi
120     \else
121       \errmessage{No room for a new \string\language!}%
122     \fi
123     \global\chardef#1\last@language

```

```

124     \wlog{\string#1 = \string\language\the\last@language}}
125 \else
126   \def\addlanguage{\alloc@9\language\chardef\@cclvi}
127 \fi
128 \def\adddialect#1#2{%
129   \global\chardef#1#2\relax
130   \wlog{\string#1 = a dialect from \string\language#2}}
131 \def\iflanguage#1{%
132   \expandafter\ifx\csname l@#1\endcsname\relax
133   \@nolanerr{#1}%
134   \else
135     \bbl@afterfi{\ifnum\csname l@#1\endcsname=\language
136       \expandafter\@firstoftwo
137       \else
138         \expandafter\@secondoftwo
139       \fi}%
140   \fi}
141 \edef\selectlanguage{%
142   \noexpand\protect
143   \expandafter\noexpand\csname selectlanguage \endcsname
144   }
145 \ifx@undefined\protect\let\protect\relax\fi
146 \ifx\documentclass\@undefined
147   \def\xstring{\string\string\string}
148 \else
149   \let\xstring\string
150 \fi
151 \xdef\bbl@language@stack{}
152 \def\bbl@push@language{%
153   \xdef\bbl@language@stack{\language\name+\bbl@language@stack}%
154   }
155 \def\bbl@pop@lang#1+#2-#3{%
156   \def\language\name{#1}\xdef#3{#2}%
157   }
158 \def\bbl@pop@language{%
159   \expandafter\bbl@pop@lang\bbl@language@stack-\bbl@language@stack
160   \expandafter\bbl@set@language\expandafter{\language\name}%
161   }
162 \expandafter\def\csname selectlanguage \endcsname#1{%
163   \bbl@push@language
164   \aftergroup\bbl@pop@language
165   \bbl@set@language{#1}}
166 \def\bbl@set@language#1{%
167   \edef\language\name{%
168     \ifnum\escapechar=\expandafter'\string#1\@empty
169     \else \string#1\@empty\fi}%
170   \select@language{\language\name}%
171   \if@filesw
172     \protected@write\@auxout{}\{\string\select@language{\language\name}}}%
173     \addtocontents{toc}{\xstring\select@language{\language\name}}}%

```

```

174 \addtocontents{lof}{\xstring\select@language{\language}\language}%
175 \addtocontents{lot}{\xstring\select@language{\language}\language}%
176 \fi}
177 \def\select@language#1{%
178 \expandafter\ifx\csname l@#1\endcsname\relax
179 \nolater{#1}%
180 \else
181 \expandafter\ifx\csname date#1\endcsname\relax
182 \noopterr{#1}%
183 \else
184 \bbl@patterns{\language}%
185 \originalTeX
186 \expandafter\def\expandafter\originalTeX
187 \expandafter{\csname noextras#1\endcsname
188 \let\originalTeX\@empty}%
189 \languageshortands{none}%
190 \babel@beginsave
191 \csname captions#1\endcsname
192 \csname date#1\endcsname
193 \csname extras#1\endcsname\relax
194 \babel@savevariable\lefthyphenmin
195 \babel@savevariable\righthyphenmin
196 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
197 \set@hyphenmins\tw@\thr@@\relax
198 \else
199 \expandafter\expandafter\expandafter\set@hyphenmins
200 \csname #1hyphenmins\endcsname\relax
201 \fi
202 \fi
203 \fi}
204 \long\def\otherlanguage#1{%
205 \csname selectlanguage \endcsname{#1}%
206 \ignorespaces
207 }
208 \long\def\endotherlanguage{%
209 \originalTeX
210 \global\@ignoretrue\ignorespaces
211 }
212 \expandafter\def\csname otherlanguage*\endcsname#1{%
213 \foreign@language{#1}%
214 }
215 \expandafter\def\csname endotherlanguage*\endcsname{%
216 \csname noextras\language\endcsname
217 }
218 \def\foreignlanguage{\protect\csname foreignlanguage \endcsname}
219 \expandafter\def\csname foreignlanguage \endcsname#1#2{%
220 \begingroup
221 \originalTeX
222 \foreign@language{#1}%
223 #2%

```

```

224 \csname noextras#1\endcsname
225 \endgroup
226 }
227 \def\foreign@language#1{%
228 \def\language#1}%
229 \expandafter\ifx\csname l@#1\endcsname\relax
230 \nolannerr{#1}%
231 \else
232 \bbl@patterns{\language}%
233 \languageshorthands{none}%
234 \csname extras#1\endcsname
235 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
236 \set@hyphenmins\tw@\thr@@\relax
237 \else
238 \expandafter\expandafter\expandafter\set@hyphenmins
239 \csname #1hyphenmins\endcsname\relax
240 \fi
241 \fi
242 }
243 \def\bbl@patterns#1{%
244 \language=\expandafter\ifx\csname l@#1:\f@encoding\endcsname\relax
245 \csname l@#1\endcsname
246 \else
247 \csname l@#1:\f@encoding\endcsname
248 \fi\relax

```

With Lua<sub>T</sub><sub>E</sub>X, load patterns and exceptions on the fly using functions from the supporting Lua module, unless of course they are already loaded for this language (identified by its number to avoid problems with synonyms).

Also, since this code will be executed at runtime, be careful while testing if we're using Lua<sub>T</sub><sub>E</sub>X.

```

249 \ifx\directlua@\undefined\else
250 \ifx\directlua\relax\else
251 \ifcsname lu@texhyphen@loaded@the\language\endcsname \else
252 \global\@namedef{lu@texhyphen@loaded@the\language}{}%
253 \directlua{
254     if not luatexhyphen then
255         dofile(assert(kpse.find_file("luatex-hyphen.lua")))
256     end
257     luatexhyphen.loadlanguage("\luatexluaescapestring{#1}",
258         \the\language)}%
259 \fi
260 \fi
261 \fi
262 }
263 \def\hyphenrules#1{%
264 \expandafter\ifx\csname l@#1\endcsname@\undefined
265 \nolannerr{#1}%
266 \else
267 \bbl@patterns{#1}%

```



```

268 \languageshorthands{none}%
269 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
270 \set@hyphenmins\tw@\thr@@\relax
271 \else
272 \expandafter\expandafter\expandafter\set@hyphenmins
273 \csname #1hyphenmins\endcsname\relax
274 \fi
275 \fi
276 }
277 \def\endhyphenrules{}
278 \def\providehyphenmins#1#2{%
279 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
280 \@namedef{#1hyphenmins}{#2}%
281 \fi}
282 \def\set@hyphenmins#1#2{\lefthyphenmin#1\righthyphenmin#2}
283 \def\LdfInit{%
284 \chardef\atcatcode=\catcode'\@
285 \catcode'\@=11\relax
286 \input babel.def\relax
287 \catcode'\@=\atcatcode \let\atcatcode\relax
288 \LdfInit}
289 \ifx\originalTeX\@undefined\let\originalTeX\@empty\fi
290 \ifx\babel@beginsave\@undefined\let\babel@beginsave\relax\fi
291 \ifx\PackageError\@undefined
292 \def\@nolanerr#1{%
293 \errhelp{Your command will be ignored, type <return> to proceed}%
294 \errmessage{You haven't defined the language #1\space yet}}
295 \def\@nopatterns#1{%
296 \message{No hyphenation patterns were loaded for}%
297 \message{the language '#1'}%
298 \message{I will use the patterns loaded for \string\language=0
299 instead}}
300 \def\@noopterr#1{%
301 \errmessage{The option #1 was not specified in \string\usepackage}
302 \errhelp{You may continue, but expect unexpected results}}
303 \def\@activated#1{%
304 \wlog{Package babel Info: Making #1 an active character}}
305 \else
306 \newcommand*{\@nolanerr}[1]{%
307 \PackageError{babel}%
308 {You haven't defined the language #1\space yet}%
309 {Your command will be ignored, type <return> to proceed}}
310 \newcommand*{\@nopatterns}[1]{%
311 \PackageWarningNoLine{babel}%
312 {No hyphenation patterns were loaded for\MessageBreak
313 the language '#1'\MessageBreak
314 I will use the patterns loaded for \string\language=0
315 instead}}
316 \newcommand*{\@noopterr}[1]{%
317 \PackageError{babel}%

```

```

318             {You haven't loaded the option #1\space yet}%
319             {You may proceed, but expect unexpected results}}
320 \newcommand*{\@activated}[1]{%
321   \PackageInfo{babel}{%
322     Making #1 an active character}}
323 \fi
324 \def\process@line#1#2 #3/{%
325   \ifx=#1
326     \process@synonym#2 /
327   \else
328     \process@language#1#2 #3/%
329   \fi
330 }
331 \toks@{}
332 \def\process@synonym#1 /{%
333   \ifnum\last@language=\m@ne
334     \expandafter\chardef\csname l@#1\endcsname\relax
335     \wlog{\string\l@#1=\string\language0}
336     \toks@\expandafter{\the\toks@
337       \expandafter\let\csname #1hyphenmins\expandafter\endcsname
338       \csname\language\hyphenmins\endcsname}%
339   \else
340     \expandafter\chardef\csname l@#1\endcsname\last@language
341     \wlog{\string\l@#1=\string\language\the\last@language}
342     \expandafter\let\csname #1hyphenmins\expandafter\endcsname
343     \csname\language\hyphenmins\endcsname
344   \fi
345 }
346 \def\process@language#1 #2 #3/{%
347   \expandafter\addlanguage\csname l@#1\endcsname
348   \expandafter\language\csname l@#1\endcsname
349   \def\language{#1}%

```

In the Lua<sub>TeX</sub> case, we have to decide whether to load the language now. Remember our choice, since we'll need it two times more.

If we choose to load the language now, mark it as loaded. This is done using  $\TeX$  macros in order to survive the format dumping-loading cycle, which would not be as straightforward using Lua objects.

```

350 \ifx\directlua@\undefined
351   \global\toks8\expandafter{\the\toks8#1, }%
352 \else
353   \directlua{
354     if not luatexhyphen then
355       dofile(assert(kpse.find_file("luatex-hyphen.lua")))
356     end
357     processnow = (tex.language == 0) or
358       (luatexhyphen.lookupname("\luatexluaescapestring{#1}") == nil)}%
359   \ifnum0=\directlua{tex.sprint(processnow and "0" or "1")}\relax
360     \global\toks8\expandafter{\the\toks8#1, }%
361     \global\@namedef{lu@texhyphen@loaded@the\language}{}%

```

```

362 \fi
363 \fi
364 \begingroup
365 \bbl@get@enc#1:\@@@
366 \ifx\bbl@hyph@enc\@empty
367 \else
368 \fontencoding{\bbl@hyph@enc}\selectfont
369 \fi
370 \lefthyphenmin\m@ne

```

Conditionally input the patterns file.

```

371 \ifx\directlua\@undefined
372 \input #2\relax
373 \else
374 \ifnum0=\directlua{tex.sprint(processnow and "0" or "1")}\relax
375 \input #2\relax
376 \fi
377 \fi

```

```

378 \ifnum\lefthyphenmin=\m@ne
379 \else
380 \expandafter\xdef\csname #1hyphenmins\endcsname{%
381 \the\lefthyphenmin\the\righthyphenmin}%
382 \fi
383 \endgroup
384 \ifnum\the\language=\z@
385 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
386 \set@hyphenmins\tw@\thr@@\relax
387 \else
388 \expandafter\expandafter\expandafter\set@hyphenmins
389 \csname #1hyphenmins\endcsname
390 \fi
391 \the\toks@
392 \fi
393 \toks@{}%
394 \def\bbl@tempa{#3}%
395 \ifx\bbl@tempa\@empty
396 \else
397 \ifx\bbl@tempa\space
398 \else

```

Conditionnaly input the exceptions file.

```

399 \ifx\directlua\@undefined
400 \input #3\relax
401 \else
402 \ifnum0=\directlua{tex.sprint(processnow and "0" or "1")}\relax
403 \input #3\relax
404 \fi
405 \directlua{processnow = nil}%
406 \fi
407 \fi

```

```

408 \fi
409 }
410 \def\bbl@get@enc#1:#2\@@@{%
411 \def\bbl@tempa{#1}%
412 \def\bbl@tempb{#2}%
413 \ifx\bbl@tempb\@empty
414 \let\bbl@hyph@enc\@empty
415 \else
416 \bbl@get@enc#2\@@@
417 \edef\bbl@hyph@enc{\bbl@tempa}%
418 \fi}
419 \openin1 = language.dat
420 \ifeof1
421 \message{I couldn't find the file language.dat,\space
422 I will try the file hyphen.tex}
423 \input hyphen.tex\relax
424 \else
425 \last@language@m@ne
426 \loop
427 \endlinechar@m@ne
428 \read1 to \bbl@line
429 \endlinechar'\^M
430 \ifx\bbl@line\@empty
431 \else
432 \edef\bbl@line{\bbl@line\space/}%
433 \expandafter\process@line\bbl@line
434 \fi
435 \iftrue \csname fi\endcsname
436 \csname if\ifeof1 false\else true\fi\endcsname
437 \repeat
438 \language=0
439 \fi
440 \closein1
441 \let\process@language\@undefined
442 \let\process@synonym\@undefined
443 \let\process@line\@undefined
444 \let\bbl@tempa\@undefined
445 \let\bbl@tempb\@undefined
446 \let\bbl@eq\@undefined
447 \let\bbl@line\@undefined
448 \let\bbl@get@enc\@undefined
449 \ifx\addto@hook\@undefined
450 \else
451 \expandafter\addto@hook\expandafter\everyjob\expandafter{%
452 \expandafter\typeout\expandafter{\the\toks8 loaded.}}
453 \fi
454 \</hyphen>

```