

The pdf texcmds package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2009/04/10 v0.4

Abstract

LUA_T_EX provides most of the commands of pdf_T_EX 1.40. However a number of utility functions are removed. This package tries to fill the gap and implements some of the missing primitive using Lua.

Contents

1	Documentation	1
1.1	General principles	2
1.2	Macros	2
1.2.1	Experimental	3
2	Implementation	4
2.1	Reload check and package identification	4
2.2	Catcodes	5
2.3	Load package infwarerr	6
2.4	Without LUA _T _E X	6
2.5	Load module	7
2.6	Lua functions	7
2.7	Lua module	10
3	Test	14
3.1	Catcode checks for loading	14
4	Installation	16
4.1	Download	16
4.2	Bundle installation	16
4.3	Package installation	16
4.4	Refresh file name databases	17
4.5	Some details for the interested	17
5	History	17
[2007/11/11 v0.1]		17
[2007/11/12 v0.2]		17
[2007/12/12 v0.3]		17
[2009/04/10 v0.4]		17
6	Index	18

1 Documentation

Some primitives of pdf_T_EX are not defined by LUA_T_EX. This package implements macro based solutions using Lua code for the following missing pdf_T_EX primitives;

- \pdfstrcmp

- `\pdfunescapehex`
- `\pdfescapehex`
- `\pdfescapename`
- `\pdfescapestring`
- `\pdffilesize`
- `\pdffilemoddate`
- `\pdffiledump`
- `\pdfmdfivesum`
- `\immediate\write18`

The original names of the primitives cannot be used:

- The syntax for their arguments cannot easily simulated by macros. The primitives using key words such as `file` (`\pdfmdfivesum`) or `offset` and `length` (`\pdffiledump`) and uses *⟨general text⟩* for the other arguments. Using token registers assignments, *⟨general text⟩* could be caught. However, the simulated primitives are expandable and register assignments would destroy this important property. (*⟨general text⟩* allows something like `\expandafter\bgroup ...`.)
- The original primitives can be expanded using one expansion step. The new macros need two expansion steps because of the additional macro expansion. Example:

```
\expandafter\foo\pdffilemoddate{file}
vs. \expandafter\expandafter\expandafter\foo\pdf@filemoddate{file}.
```

LUA_TE_X isn't stable yet and thus the status of this package is *experimental*. Feedback is welcome.

1.1 General principles

Naming convention: Usually this package defines a macro `\pdf@⟨cmd⟩` if pdf_TE_X provides `\pdf⟨cmd⟩`.

Arguments: The order of arguments in `\pdf@⟨cmd⟩` is the same as for the corresponding primitive of pdf_TE_X. The arguments are ordinary undelimited T_EX arguments, no *⟨general text⟩* and without additional keywords.

Expandibility: The macro `\pdf@⟨cmd⟩` is expandable if the corresponding pdf_TE_X primitive has this property. Exact two expansion steps are necessary (first is the macro expansion).

Without Lua_TE_X: The macros `\pdf@⟨cmd⟩` are mapped to the commands of pdf_TE_X if they are available. Otherwise they are undefined.

1.2 Macros

`\pdf@strcmp {⟨stringA⟩} {⟨stringB⟩}`

Same as `\pdfstrcmp{⟨stringA⟩}{⟨stringB⟩}`.

`\pdf@unescapehex {⟨string⟩}`

Same as `\pdfunescapehex{⟨string⟩}`. The argument is a byte string given in hexadecimal notation. The result are character tokens from 0 until 255 with catcode 12 and the space with catcode 10.

<code>\pdf@escapehex {⟨string⟩}</code> <code>\pdf@escapestring {⟨string⟩}</code> <code>\pdf@escapename {⟨string⟩}</code>
--

Same as the primitives of pdfTeX. However pdfTeX does not know about characters with codes 256 and larger. Thus the string is treated as byte string, characters with more than eight bits are ignored.

<code>\pdf@filesize {⟨filename⟩}</code>

Same as `\pdffilesize{⟨filename⟩}`.

<code>\pdf@filemoddate {⟨filename⟩}</code>
--

Same as `\pdffilemoddate{⟨filename⟩}`.

<code>\pdf@filedump {⟨offset⟩} {⟨length⟩} {⟨filename⟩}</code>

Same as `\pdffiledump offset ⟨offset⟩ length ⟨length⟩ {⟨filename⟩}`. Both `⟨offset⟩` and `⟨length⟩` must not be empty, but must be a valid TeX number.

<code>\pdf@mdfivesum {⟨string⟩}</code>
--

Same as `\pdfmdfivesum{⟨string⟩}`. Keyword `file` is supported by macro `\pdf@filemdfivesum`.

<code>\pdf@filemdfivesum {⟨filename⟩}</code>
--

Same as `\pdfmdfivesum file{⟨filename⟩}`.

<code>\pdf@shellescape</code>

Same as `\pdfshellescape`. It expands to 1 if external commands can be executed and 0 otherwise. In pdfTeX external commands must be enabled first by command line option or configuration option. In LuaTeX option `--safer` disables the execution of external commands.

<code>\pdf@system {⟨cmdline⟩}</code>

It is a wrapper for `\immediate\write18` in pdfTeX or `os.execute` in LuaTeX.

In theory `os.execute` returns a status number. But its meaning is quite undefined. Are there some reliable properties? Does it make sense to provide an user interface to this status exit code?

1.2.1 Experimental

<code>\pdf@unescapehexnative {⟨string⟩}</code> <code>\pdf@escapehexnative {⟨string⟩}</code> <code>\pdf@escapenamenative {⟨string⟩}</code> <code>\pdf@mdfivesumnative {⟨string⟩}</code>

The variants without `native` in the macro name are supposed to be compatible with pdfTeX. However characters with more than eight bits are not supported and are ignored. If LuaTeX is running, then its UTF-8 coded strings are used.

Thus the full unicode character range is supported. However the result differs from pdfTeX for characters with eight or more bits.

`\pdf@pipe {⟨cmdline⟩}`

It calls `⟨cmdline⟩` and returns the output of the external program in the usual manner as byte string (catcode 12, space with catcode 10). The Lua documentation says, that the used `io.popen` may not be available on all platforms. Then macro `\pdf@pipe` is undefined.

2 Implementation

```
1 ⟨*package⟩
```

2.1 Reload check and package identification

Reload check, especially if the package is not used with L^AT_EX.

```
2 \begingroup
3   \catcode44 12 % ,
4   \catcode45 12 % -
5   \catcode46 12 % .
6   \catcode58 12 % :
7   \catcode64 11 % @
8   \expandafter\let\expandafter\x\csname ver@pdfdoccmds.sty\endcsname
9   \ifcase 0%
10    \ifx\x\relax % plain
11    \else
12      \ifx\x\empty % LaTeX
13      \else
14        1%
15      \fi
16    \fi
17  \else
18    \catcode35 6 % #
19    \catcode123 1 % {
20    \catcode125 2 % }
21    \expandafter\ifx\csname PackageInfo\endcsname\relax
22      \def\x#1#2{%
23        \immediate\write-1{Package #1 Info: #2.}%
24      }%
25    \else
26      \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
27    \fi
28    \x{pdfdoccmds}{The package is already loaded}%
29  \endgroup
30  \expandafter\endinput
31  \fi
32 \endgroup
```

Package identification:

```
33 \begingroup
34   \catcode35 6 % #
35   \catcode40 12 % (
36   \catcode41 12 % )
37   \catcode44 12 % ,
38   \catcode45 12 % -
39   \catcode46 12 % .
40   \catcode47 12 % /
41   \catcode58 12 % :
42   \catcode64 11 % @
43   \catcode123 1 % {
```

```

44 \catcode125 2 % }
45 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
46 \def\x#1#2#3[#4]{\endgroup
47 \immediate\write-1{Package: #3 #4}%
48 \xdef#1{#4}%
49 }%
50 \else
51 \def\x#1#2[#3]{\endgroup
52 #2[#3]}%
53 \ifx#1\@undefined
54 \xdef#1{#3}%
55 \fi
56 \ifx#1\relax
57 \xdef#1{#3}%
58 \fi
59 }%
60 \fi
61 \expandafter\x\csname ver@pdftexcmds.sty\endcsname
62 \ProvidesPackage{pdftexcmds}%
63 [2009/04/10 v0.4 LuaTeX support for pdfTeX utility functions (HO)]

```

2.2 Catcodes

```

64 \begingroup
65 \catcode123 1 % {
66 \catcode125 2 % }
67 \def\x{\endgroup
68 \expandafter\edef\csname pdftexcmds@AtEnd\endcsname{%
69 \catcode35 \the\catcode35\relax
70 \catcode64 \the\catcode64\relax
71 \catcode123 \the\catcode123\relax
72 \catcode125 \the\catcode125\relax
73 }%
74 }%
75 \x
76 \catcode35 6 % #
77 \catcode64 11 % @
78 \catcode123 1 % {
79 \catcode125 2 % }
80 \def\TMP@EnsureCode#1#2{%
81 \edef\pdftexcmds@AtEnd{%
82 \pdftexcmds@AtEnd
83 \catcode#1 \the\catcode#1\relax
84 }%
85 \catcode#1 #2\relax
86 }
87 \TMP@EnsureCode{10}{12}% ^^J
88 \TMP@EnsureCode{33}{12}% !
89 \TMP@EnsureCode{34}{12}% "
90 \TMP@EnsureCode{39}{12}% '
91 \TMP@EnsureCode{40}{12}% (
92 \TMP@EnsureCode{41}{12}% )
93 \TMP@EnsureCode{42}{12}% *
94 \TMP@EnsureCode{43}{12}% +
95 \TMP@EnsureCode{44}{12}% ,
96 \TMP@EnsureCode{45}{12}% -
97 \TMP@EnsureCode{46}{12}% .
98 \TMP@EnsureCode{47}{12}% /
99 \TMP@EnsureCode{58}{12}% :
100 \TMP@EnsureCode{60}{12}% <
101 \TMP@EnsureCode{61}{12}% =
102 \TMP@EnsureCode{62}{12}% >

```

```

103 \TMP@EnsureCode{94}{7}% ^ (superscript)
104 \TMP@EnsureCode{95}{12}% _ (other)
105 \TMP@EnsureCode{96}{12}% '
106 \TMP@EnsureCode{126}{12}% ~ (other)

```

2.3 Load package infwarerr

```

107 \begingroup\expandafter\expandafter\expandafter\endgroup
108 \expandafter\ifx\csname RequirePackage\endcsname\relax
109   \input infwarerr.sty\relax
110   \input ifluatex.sty\relax
111 \else
112   \RequirePackage{infwarerr}[2007/09/09]%
113   \RequirePackage{ifluatex}[2009/04/10]%
114 \fi

```

2.4 Without LuaTeX

```

115 \ifluatex
116 \else
117   \@PackageInfo{pdftexcmds}{LuaTeX not detected}%
118   \def\pdftexcmds@nopdftex{%
119     \@PackageInfoNoLine{pdftexcmds}{pdfTeX >= 1.30 not detected}%
120     \let\pdftexcmds@nopdftex\relax
121   }%
122   \def\pdftexcmds@temp#1{%
123     \begingroup\expandafter\expandafter\expandafter\endgroup
124     \expandafter\ifx\csname pdf#1\endcsname\relax
125       \pdftexcmds@nopdftex
126     \else
127       \expandafter\def\csname pdf@#1\endcsname\expandafter\endcsname
128       \expandafter##\expandafter{%
129         \csname pdf#1\endcsname
130       }%
131     \fi
132   }%
133   \pdftexcmds@temp{strcmp}%
134   \pdftexcmds@temp{escapehex}%
135   \let\pdf@escapehexnative\pdf@escapehex
136   \pdftexcmds@temp{unescapehex}%
137   \let\pdf@unescapehexnative\pdf@unescapehex
138   \pdftexcmds@temp{escapestring}%
139   \pdftexcmds@temp{escapename}%
140   \pdftexcmds@temp{filesize}%
141   \pdftexcmds@temp{filemoddate}%
142   \begingroup\expandafter\expandafter\expandafter\endgroup
143   \expandafter\ifx\csname pdfshellescape\endcsname\relax
144     \pdftexcmds@nopdftex
145   \else
146     \def\pdf@shellescape{%
147       \pdfshellescape
148     }%
149   \fi
150   \begingroup\expandafter\expandafter\expandafter\endgroup
151   \expandafter\ifx\csname pdffiledump\endcsname\relax
152     \pdftexcmds@nopdftex
153   \else
154     \def\pdf@filedump#1#2#3{%
155       \pdffiledump offset#1 length#2{#3}%
156     }%
157   \fi
158   \begingroup\expandafter\expandafter\expandafter\endgroup
159   \expandafter\ifx\csname pdfmdfivesum\endcsname\relax
160     \pdftexcmds@nopdftex

```

```

161 \else
162   \def\pdf@mdfivesum#\pdfmdfivesum}%
163   \let\pdf@mdfivesumnative\pdf@mdfivesum
164   \def\pdf@filemdfivesum#\pdfmdfivesum file}%
165 \fi
166 \def\pdf@system#{%
167   \immediate\write18%
168 }%
169 \pdfTexcmds@AtEnd
170 \expandafter\endinput
171 \fi

```

\pdfTexcmds@directlua

```

172 \ifnum\luatexversion<36 %
173   \def\pdfTexcmds@directlua{\directlua0 }%
174 \else
175   \let\pdfTexcmds@directlua\directlua
176 \fi

```

2.5 Load module

```

177 \begingroup\expandafter\expandafter\expandafter\endgroup
178 \expandafter\ifx\csname RequirePackage\endcsname\relax
179   \input luatex-loader.sty\relax
180 \else
181   \RequirePackage{luatex-loader}[2009/04/10]%
182 \fi
183 \pdfTexcmds@directlua{%
184   require("oberdiek.pdfTexcmds")%
185 }

```

2.6 Lua functions

\pdfTexcmds@toks

```

186 \begingroup\expandafter\expandafter\expandafter\endgroup
187 \expandafter\ifx\csname newtoks\endcsname\relax
188   \toksdef\pdfTexcmds@toks=0 %
189 \else
190   \csname newtoks\endcsname\pdfTexcmds@toks
191 \fi

192 \ifnum\luatexversion<36 %
193 \else
194   \catcode'\0=9 %
195 \fi

```

\pdf@strcmp

```

196 \long\def\pdf@strcmp#1#2{%
197   \directlua0{%
198     oberdiek.pdfTexcmds.strcmp("\luaescapestring{#1}",%
199       "\luaescapestring{#2}")%
200   }%
201 }%

```

\pdf@escapehex

```

202 \long\def\pdf@escapehex#1{%
203   \directlua0{%
204     oberdiek.pdfTexcmds.escapehex("\luaescapestring{#1}", "byte")%
205   }%
206 }%

```

\pdf@escapehexnative

```

207 \long\def\pdf@escapehexnative#1{%
208   \directlua0{%
209     oberdiek.pdfdocmds.escapehex("\luaescapestring{#1}")%
210   }%
211 }%

```

\pdf@unescapehex

```

212 \def\pdf@unescapehex#1{%
213   \the\expandafter\pdfdocmds@toks
214   \directlua0{%
215     oberdiek.pdfdocmds.toks="pdfdocmds@toks"%
216     oberdiek.pdfdocmds.unescapehex("\luaescapestring{#1}", "byte")%
217   }%
218 }%

```

\pdf@unescapehexnative

```

219 \def\pdf@unescapehexnative#1{%
220   \the\expandafter\pdfdocmds@toks
221   \directlua0{%
222     oberdiek.pdfdocmds.toks="pdfdocmds@toks"%
223     oberdiek.pdfdocmds.unescapehex("\luaescapestring{#1}")%
224   }%
225 }%

```

\pdf@escapestring

```

226 \long\def\pdf@escapestring#1{%
227   \directlua0{%
228     oberdiek.pdfdocmds.escapestring("\luaescapestring{#1}", "byte")%
229   }%
230 }

```

\pdf@escapename

```

231 \long\def\pdf@escapename#1{%
232   \directlua0{%
233     oberdiek.pdfdocmds.escapename("\luaescapestring{#1}", "byte")%
234   }%
235 }

```

\pdf@escapenamenative

```

236 \long\def\pdf@escapenamenative#1{%
237   \directlua0{%
238     oberdiek.pdfdocmds.escapename("\luaescapestring{#1}")%
239   }%
240 }

```

\pdf@filesize

```

241 \def\pdf@filesize#1{%
242   \directlua0{%
243     oberdiek.pdfdocmds.filesize("\luaescapestring{#1}")%
244   }%
245 }

```

\pdf@filemoddate

```

246 \def\pdf@filemoddate#1{%
247   \directlua0{%
248     oberdiek.pdfdocmds.filemoddate("\luaescapestring{#1}")%
249   }%
250 }

```



```

\pdf@filedump
251 \def\pdf@filedump#1#2#3{%
252   \directlua0{%
253     oberdiek.pdfcmds.filedump("\luaescapestring{\number#1}",%
254       "\luaescapestring{\number#2}",%
255       "\luaescapestring{#3}")%
256   }%
257 }%

\pdf@mdfivesum
258 \long\def\pdf@mdfivesum#1{%
259   \directlua0{%
260     oberdiek.pdfcmds.mdfivesum("\luaescapestring{#1}", "byte")%
261   }%
262 }%

\pdf@mdfivesumnative
263 \long\def\pdf@mdfivesumnative#1{%
264   \directlua0{%
265     oberdiek.pdfcmds.mdfivesum("\luaescapestring{#1}")%
266   }%
267 }%

\pdf@filemdfivesum
268 \def\pdf@filemdfivesum#1{%
269   \directlua0{%
270     oberdiek.pdfcmds.filemdfivesum("\luaescapestring{#1}")%
271   }%
272 }%

\pdf@shellescape
273 \def\pdf@shellescape{%
274   \directlua0{%
275     oberdiek.pdfcmds.shellescape()%
276   }%
277 }

\pdf@system
278 \def\pdf@system#1{%
279   \directlua0{%
280     oberdiek.pdfcmds.system("\luaescapestring{#1}")%
281   }%
282 }

\pdf@lastsystemstatus
283 \def\pdf@lastsystemstatus{%
284   \directlua0{%
285     oberdiek.pdfcmds.lastsystemstatus()%
286   }%
287 }

\pdf@lastsystemexit
288 \def\pdf@lastsystemexit{%
289   \directlua0{%
290     oberdiek.pdfcmds.lastsystemexit()%
291   }%
292 }

293 \catcode'\0=12 %

```

`\pdf@pipe` Check availability of `io.popen` first.

```
294 \ifnum0%
295   \pdftexcmds@directlua{%
296     if io.popen then %
297       tex.write("1")%
298     end%
299   }%
300   =1 %
301 \def\pdf@pipe#1{%
302   \the\expandafter\pdftexcmds@toks
303   \pdftexcmds@directlua{%
304     oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
305     oberdiek.pdftexcmds.pipe("\luaescapestring{#1}")%
306   }%
307 }%
308 \fi

309 \pdftexcmds@AtEnd
310 \end{package}
```

2.7 Lua module

```
311 \lua
312 module("oberdiek.pdftexcmds", package.seeall)
313 local systemexitstatus
314 function strcmp(A, B)
315   if A == B then
316     tex.write("0")
317   elseif A < B then
318     tex.write("-1")
319   else
320     tex.write("1")
321   end
322 end
323 local function utf8_to_byte(str)
324   local i = 0
325   local n = string.len(str)
326   local t = {}
327   while i < n do
328     i = i + 1
329     local a = string.byte(str, i)
330     if a < 128 then
331       table.insert(t, string.char(a))
332     else
333       if a >= 192 and i < n then
334         i = i + 1
335         local b = string.byte(str, i)
336         if b < 128 or b >= 192 then
337           i = i - 1
338         elseif a == 194 then
339           table.insert(t, string.char(b))
340         elseif a == 195 then
341           table.insert(t, string.char(b + 64))
342         end
343       end
344     end
345   end
346   return table.concat(t)
347 end
348 function escapehex(str, mode)
349   if mode == "byte" then
350     str = utf8_to_byte(str)
```

```

351 end
352 tex.write((string.gsub(str, ".",
353   function (ch)
354     return string.format("%02X", string.byte(ch))
355   end
356 )))
357 end

```

See procedure `unescapehex` in file `utils.c` of pdfTeX. Caution: `tex.write` ignores leading spaces.

```

358 function unescapehex(str, mode)
359   local a = 0
360   local first = true
361   local result = {}
362   for i = 1, string.len(str), 1 do
363     local ch = string.byte(str, i)
364     if ch >= 48 and ch <= 57 then
365       ch = ch - 48
366     elseif ch >= 65 and ch <= 70 then
367       ch = ch - 55
368     elseif ch >= 97 and ch <= 102 then
369       ch = ch - 87
370     else
371       ch = nil
372     end
373     if ch then
374       if first then
375         a = ch * 16
376         first = false
377       else
378         table.insert(result, a + ch)
379         first = true
380       end
381     end
382   end
383   if not first then
384     table.insert(result, a)
385   end
386   if mode == "byte" then
387     local utf8 = {}
388     for i, a in ipairs(result) do
389       if a < 128 then
390         table.insert(utf8, a)
391       else
392         if a < 192 then
393           table.insert(utf8, 194)
394           a = a - 128
395         else
396           table.insert(utf8, 195)
397           a = a - 192
398         end
399         table.insert(utf8, a + 128)
400       end
401     end
402     result = utf8
403   end
404   tex.settoks(toks, string.char(unpack(result)))
405 end

```

See procedure `escapestring` in file `utils.c` of pdfTeX.

```

406 function escapestring(str, mode)
407   if mode == "byte" then
408     str = utf8_to_byte(str)
409   end

```

```

410 tex.write((string.gsub(str, ".",
411     function (ch)
412         local b = string.byte(ch)
413         if b < 33 or b > 126 then
414             return string.format("\\%.3o", b)
415         end
416         if b == 40 or b == 41 or b == 92 then
417             return "\\" .. ch
418         end

```

Lua 5.1 returns the match in case of return value nil.

```

419         return nil
420     end
421 )))
422 end

```

See procedure `escapename` in file `utils.c` of pdfTeX.

```

423 function escapename(str, mode)
424     if mode == "byte" then
425         str = utf8_to_byte(str)
426     end
427     tex.write((string.gsub(str, ".",
428         function (ch)
429             local b = string.byte(ch)
430             if b == 0 then

```

In Lua 5.0 nil could be used for the empty string, But nil returns the match in Lua 5.1, thus we use the empty string explicitly.

```

431         return ""
432     end
433     if b <= 32 or b >= 127
434         or b == 35 or b == 37 or b == 40 or b == 41
435         or b == 47 or b == 60 or b == 62 or b == 91
436         or b == 93 or b == 123 or b == 125 then
437         return string.format("#%.2X", b)
438     else

```

Lua 5.1 returns the match in case of return value nil.

```

439         return nil
440     end
441 end
442 )))
443 end
444 function filesize(filename)
445     local foundfile = kpse.find_file(filename, "tex", true)
446     if foundfile then
447         local size = lfs.attributes(foundfile, "size")
448         if size then
449             tex.write(size)
450         end
451     end
452 end

```

See procedure `makepdftime` in file `utils.c` of pdfTeX.

```

453 function filemoddate(filename)
454     local foundfile = kpse.find_file(filename, "tex", true)
455     if foundfile then
456         local date = lfs.attributes(foundfile, "modification")
457         if date then
458             local d = os.date("*t", date)
459             if d.sec >= 60 then
460                 d.sec = 59
461             end
462             local u = os.date("!*t", date)
463             local off = 60 * (d.hour - u.hour) + d.min - u.min
464             if d.year ~= u.year then

```

```

465         if d.year > u.year then
466             off = off + 1440
467         else
468             off = off - 1440
469         end
470     elseif d.yday ~= u.yday then
471         if d.yday > u.yday then
472             off = off + 1440
473         else
474             off = off - 1440
475         end
476     end
477     local timezone
478     if off == 0 then
479         timezone = "Z"
480     else
481         local hours = math.floor(off / 60)
482         local mins = math.abs(off - hours * 60)
483         timezone = string.format("%+03d'%02d'", hours, mins)
484     end
485     tex.write(string.format("D:%04d%02d%02d%02d%02d%s",
486         d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
487 end
488 end
489 end
490 function filedump(offset, length, filename)
491     length = tonumber(length)
492     if length and length > 0 then
493         local foundfile = kpse.find_file(filename, "tex", true)
494         if foundfile then
495             offset = tonumber(offset)
496             if not offset then
497                 offset = 0
498             end
499             local filehandle = io.open(foundfile, "r")
500             if filehandle then
501                 if offset > 0 then
502                     filehandle:seek("set", offset)
503                 end
504                 local dump = filehandle:read(length)
505                 escapehex(dump)
506             end
507         end
508     end
509 end
510 function md5sum(str, mode)
511     if mode == "byte" then
512         str = utf8_to_byte(str)
513     end
514     escapehex(md5.sum(str))
515 end
516 function filemd5sum(filename)
517     local foundfile = kpse.find_file(filename, "tex", true)
518     if foundfile then
519         local filehandle = io.open(foundfile, "r")
520         if filehandle then
521             local contents = filehandle:read("*a")
522             escapehex(md5.sum(contents))
523         end
524     end
525 end
526 function shellescape()

```

```

527 if os.execute then
528   tex.write("1")
529 else
530   tex.write("0")
531 end
532 end
533 function system(cmdline)
534   systemexitstatus = nil
535   texio.write_nl("log", "system(" .. cmdline .. ") ")
536   if os.execute then
537     texio.write("log", "executed.")
538     systemexitstatus = os.execute(cmdline)
539   else
540     texio.write("log", "disabled.")
541   end
542 end
543 function lastsystemstatus()
544   local result = tonumber(systemexitstatus)
545   if result then
546     local x = math.floor(result / 256)
547     tex.write(result - 256 * math.floor(result / 256))
548   end
549 end
550 function lastsystemexit()
551   local result = tonumber(systemexitstatus)
552   if result then
553     tex.write(math.floor(result / 256))
554   end
555 end
556 function pipe(cmdline)
557   local result
558   systemexitstatus = nil
559   texio.write_nl("log", "pipe(" .. cmdline .. ") ")
560   if io.popen then
561     texio.write("log", "executed.")
562     local handle = io.popen(cmdline, "r")
563     if handle then
564       result = handle:read("*a")
565       handle:close()
566     end
567   else
568     texio.write("log", "disabled.")
569   end
570   if result then
571     tex.settoks(toks, result)
572   else
573     tex.settoks(toks, "")
574   end
575 end
576  $\langle$ /lua $\rangle$ 

```

3 Test

3.1 Catcode checks for loading

```

577  $\langle$ *test1 $\rangle$ 
578 \catcode'\{=1 %
579 \catcode'\}=2 %
580 \catcode'\#=6 %
581 \catcode'\@=11 %
582 \expandafter\ifx\csname count@\endcsname\relax
583   \countdef\count@=255 %

```

```

584 \fi
585 \expandafter\ifx\csname @gobble\endcsname\relax
586   \long\def@gobble#1{%
587 \fi
588 \expandafter\ifx\csname @firstofone\endcsname\relax
589   \long\def@firstofone#1{#1}%
590 \fi
591 \expandafter\ifx\csname loop\endcsname\relax
592   \expandafter@firstofone
593 \else
594   \expandafter@gobble
595 \fi
596 {%
597   \def\loop#1\repeat{%
598     \def\body{#1}%
599     \iterate
600   }%
601   \def\iterate{%
602     \body
603     \let\next\iterate
604   \else
605     \let\next\relax
606   \fi
607   \next
608 }%
609 \let\repeat=\fi
610 }%
611 \def\RestoreCatcodes{}
612 \count@=0 %
613 \loop
614   \edef\RestoreCatcodes{%
615     \RestoreCatcodes
616     \catcode\the\count@=\the\catcode\count@\relax
617   }%
618 \ifnum\count@<255 %
619   \advance\count@ 1 %
620 \repeat
621
622 \def\RangeCatcodeInvalid#1#2{%
623   \count@=#1\relax
624   \loop
625     \catcode\count@=15 %
626     \ifnum\count@<#2\relax
627       \advance\count@ 1 %
628     \repeat
629 }
630 \expandafter\ifx\csname LoadCommand\endcsname\relax
631   \def\LoadCommand{\input pdftexcmds.sty\relax}%
632 \fi
633 \def\Test{%
634   \RangeCatcodeInvalid{0}{47}%
635   \RangeCatcodeInvalid{58}{64}%
636   \RangeCatcodeInvalid{91}{96}%
637   \RangeCatcodeInvalid{123}{255}%
638   \catcode'\@=12 %
639   \catcode'\=0 %
640   \catcode'\{=1 %
641   \catcode'\}=2 %
642   \catcode'\#=6 %
643   \catcode'\[=12 %
644   \catcode'\]=12 %
645   \catcode'\%=14 %

```

```

646 \catcode'\ =10 %
647 \catcode13=5 %
648 \LoadCommand
649 \RestoreCatcodes
650 }
651 \Test
652 \csname @@end\endcsname
653 \end
654 </test1>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/oberdiek/pdftexcmds.dtx](#) The source file.

[CTAN:macros/latex/contrib/oberdiek/pdftexcmds.pdf](#) Documentation.

Bundle. All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

[CTAN:install/macros/latex/contrib/oberdiek.tds.zip](#)

TDS refers to the standard “A Directory Structure for T_EX Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

4.2 Bundle installation

Unpacking. Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```

chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/

```

4.3 Package installation

Unpacking. The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain-T_EX:

```
tex pdftexcmds.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

```

pdftexcmds.sty      → tex/generic/oberdiek/pdftexcmds.sty
oberdiek.pdftexcmds.lua → scripts/oberdiek/oberdiek.pdftexcmds.lua
pdftexcmds.lua      → scripts/oberdiek/pdftexcmds.lua
pdftexcmds.pdf      → doc/latex/oberdiek/pdftexcmds.pdf
pdftexcmds.dtx      → source/latex/oberdiek/pdftexcmds.dtx

```

If you have a `docstrip.cfg` that configures and enables `docstrip`’s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

¹<http://ftp.ctan.org/tex-archive/>

4.4 Refresh file name databases

If your $\text{T}_{\text{E}}\text{X}$ distribution (te $\text{T}_{\text{E}}\text{X}$, mik $\text{T}_{\text{E}}\text{X}$, ...) relies on file name databases, you must refresh these. For example, te $\text{T}_{\text{E}}\text{X}$ users run `texhash` or `mktextlsr`.

4.5 Some details for the interested

Attached source. The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk pdftexcmds.pdf unpack_files output .
```

Unpacking with $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. The `.dtx` chooses its action depending on the format:

plain- $\text{T}_{\text{E}}\text{X}$: Run `docstrip` and extract the files.

$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$: Generate the documentation.

If you insist on using $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ for `docstrip` (really, `docstrip` does not need $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{pdftexcmds.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdf $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$:

```
pdflatex pdftexcmds.dtx
makeindex -s gind.ist pdftexcmds.idx
pdflatex pdftexcmds.dtx
makeindex -s gind.ist pdftexcmds.idx
pdflatex pdftexcmds.dtx
```

5 History

[2007/11/11 v0.1]

- First version.

[2007/11/12 v0.2]

- Short description fixed.

[2007/12/12 v0.3]

- Organization of Lua code as module.

[2009/04/10 v0.4]

- Adaptation for syntax change of `\directlua` in $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 0.36.

6 Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols		I	
\#	580, 642	\ifcase	9
\%	645	\ifluatex	115
\@	581, 638	\ifnum	172, 192, 294, 618, 626
\@PackageInfo	117	\ifx	10, 12, 21, 45, 53, 56, 108, 124, 143, 151, 159, 178, 187, 582, 585, 588, 591, 630
\@PackageInfoNoLine	119	\immediate	23, 47, 167
\@firstofone	589, 592	\input	109, 110, 179, 631
\@gobble	586, 594	\iterate	599, 601, 603
\@undefined	53		
\[643	L	
\\	414, 417, 639	\LoadCommand	631, 648
\{	578, 640	\loop	597, 613, 624
\}	579, 641	\luaescapestring	198, 199, 204, 209, 216, 223, 228, 233, 238, 243, 248, 253, 254, 255, 260, 265, 270, 280, 305
\]	644	\luatexversion	172, 192
Numbers			
\0	194, 293	N	
_	646	\next	603, 605, 607
A		\number	253, 254
\advance	619, 627	P	
B		\PackageInfo	26
\body	598, 602	\pdf@escapehex	3, 135, 202
C		\pdf@escapehexnative	135, 207
\catcode	3, 4, 5, 6, 7, 18, 19, 20, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 65, 66, 69, 70, 71, 72, 76, 77, 78, 79, 83, 85, 194, 293, 578, 579, 580, 581, 616, 625, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647	\pdf@escapename	231
\count@	583, 612, 616, 618, 619, 623, 625, 626, 627	\pdf@escapenamenative	236
\countdef	583	\pdf@escapestring	226
\csname	8, 21, 45, 61, 68, 108, 124, 127, 129, 143, 151, 159, 178, 187, 190, 582, 585, 588, 591, 630, 652	\pdf@filedump	3, 154, 251
D		\pdf@filemdfivesum	3, 164, 268
\directlua	173, 175, 197, 203, 208, 214, 221, 227, 232, 237, 242, 247, 252, 259, 264, 269, 274, 279, 284, 289	\pdf@filemoddate	3, 246
E		\pdf@filesize	3, 241
\empty	12	\pdf@lastsystemexit	288
\end	653	\pdf@lastsystemstatus	283
\endcsname	8, 21, 45, 61, 68, 108, 124, 127, 129, 143, 151, 159, 178, 187, 190, 582, 585, 588, 591, 630, 652	\pdf@mdfivesum	3, 162, 163, 258
\endinput	30, 170	\pdf@mdfivesumnative	163, 263
		\pdf@pipe	4, 294
		\pdf@shellescape	3, 146, 273
		\pdf@strcmp	2, 196
		\pdf@system	3, 166, 278
		\pdf@unescapehex	2, 137, 212
		\pdf@unescapehexnative	3, 137, 219
		\pdf@filedump	155
		\pdf@mdfivesum	162, 164
		\pdf@shellescape	147
		\pdf@texcmds@AtEnd	81, 82, 169, 309
		\pdf@texcmds@directlua	172, 183, 295, 303
		\pdf@texcmds@nopdf@tex	118, 120, 125, 144, 152, 160
		\pdf@texcmds@temp	122, 133, 134, 136, 138, 139, 140, 141
		\pdf@texcmds@toks	186, 213, 220, 302
		\ProvidesPackage	62

R	\TMP@EnsureCode . . . 80, 87, 88, 89,
\RangeCatcodeInvalid	90, 91, 92, 93, 94, 95, 96, 97, 98,
. 622, 634, 635, 636, 637	99, 100, 101, 102, 103, 104, 105, 106
\repeat 597, 609, 620, 628	\toksdef 188
\RequirePackage 112, 113, 181	
\RestoreCatcodes . . 611, 614, 615, 649	W
	\write 23, 47, 167
T	
\Test 633, 651	X
\the 69, 70, 71, 72, 83, 213, 220, 302, 616	\x 8, 10, 12, 22, 26, 28, 46, 51, 61, 67, 75