

# The pdftextcmds package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2007/11/12 v0.2

## Abstract

LUA<sub>T</sub><sub>E</sub>X provides most of the commands of pdf<sub>T</sub><sub>E</sub>X 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

<b>1</b>	<b>Documentation</b>	<b>1</b>
1.1	General principles . . . . .	2
1.2	Macros . . . . .	2
1.2.1	Experimental . . . . .	3
<b>2</b>	<b>Implementation</b>	<b>4</b>
2.1	Reload check and package identification . . . . .	4
2.2	Catcodes . . . . .	5
2.3	Load package infwarerr . . . . .	5
2.4	Without LUA <sub>T</sub> <sub>E</sub> X . . . . .	5
2.5	Lua functions . . . . .	6
<b>3</b>	<b>Test</b>	<b>14</b>
3.1	Catcode checks for loading . . . . .	14
<b>4</b>	<b>Installation</b>	<b>15</b>
4.1	Download . . . . .	15
4.2	Bundle installation . . . . .	15
4.3	Package installation . . . . .	16
4.4	Refresh file name databases . . . . .	16
4.5	Some details for the interested . . . . .	16
<b>5</b>	<b>History</b>	<b>17</b>
	[2007/11/11 v0.1] . . . . .	17
	[2007/11/12 v0.2] . . . . .	17
<b>6</b>	<b>Index</b>	<b>17</b>

## 1 Documentation

Some primitives of pdf<sub>T</sub><sub>E</sub>X are not defined by LUA<sub>T</sub><sub>E</sub>X. This package implements macro based solutions using Lua code for the following missing pdf<sub>T</sub><sub>E</sub>X 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<sub>T</sub>E<sub>X</sub> 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<sub>T</sub>E<sub>X</sub> provides `\pdf⟨cmd⟩`.

**Arguments:** The order of arguments in `\pdf@⟨cmd⟩` is the same as for the corresponding primitive of pdf<sub>T</sub>E<sub>X</sub>. The arguments are ordinary undelimited T<sub>E</sub>X arguments, no *⟨general text⟩* and without additional keywords.

**Expandibility:** The macro `\pdf@⟨cmd⟩` is expandable if the corresponding pdf<sub>T</sub>E<sub>X</sub> primitive has this property. Exact two expansion steps are necessary (first is the macro expansion).

**Without Lua<sub>T</sub>E<sub>X</sub>:** The macros `\pdf@⟨cmd⟩` are mapped to the commands of pdf<sub>T</sub>E<sub>X</sub> 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.

`\pdf@escapehex {⟨string⟩}`  
`\pdf@escapestring {⟨string⟩}`  
`\pdf@escapename {⟨string⟩}`

Same as the primitives of pdf<sub>T</sub>E<sub>X</sub>. However pdf<sub>T</sub>E<sub>X</sub> 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.

`\pdf@filesize {<filename>}`

Same as `\pdffilesize{<filename>}`.

`\pdf@filemoddate {<filename>}`

Same as `\pdffilemoddate{<filename>}`.

`\pdf@filedump {<offset>} {<length>} {<filename>}`

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

`\pdf@mdfivesum {<string>}`

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

`\pdf@filemdfivesum {<filename>}`

Same as `\pdfmdfivesum file{<filename>}`.

`\pdf@shellescape`

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.

`\pdf@system {<cmdline>}`

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

`\pdf@unescapehexnative {<string>}`  
`\pdf@escapehexnative {<string>}`  
`\pdf@escapenamenative {<string>}`  
`\pdf@mdfivesumnative {<string>}`

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 documenta-

tion 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<sup>A</sup>T<sub>E</sub>X.

```
2 \begingroup
3 \catcode44 12 % ,
4 \catcode45 12 % -
5 \catcode46 12 % .
6 \catcode58 12 % :
7 \catcode64 11 % @
8 \expandafter\let\expandafter\x\csname ver@pdf texcmds.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{pdf texcmds}{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\relax
54      \xdef#1{#3}%
55      \fi
56    }%
57  \fi
58 \expandafter\x\csname ver@pdftexcmds.sty\endcsname
59 \ProvidesPackage{pdftexcmds}%
60 [2007/11/12 v0.2 LuaTeX support for pdfTeX utility functions (HO)]

```

## 2.2 Catcodes

```

61 \begingroup
62   \catcode123 1 % {
63   \catcode125 2 % }
64   \def\x{\endgroup
65     \expandafter\edef\csname pdftexcmds@AtEnd\endcsname{%
66       \catcode35 \the\catcode35\relax
67       \catcode64 \the\catcode64\relax
68       \catcode123 \the\catcode123\relax
69       \catcode125 \the\catcode125\relax
70     }%
71   }%
72 \x
73 \catcode35 6 % #
74 \catcode64 11 % @
75 \catcode123 1 % {
76 \catcode125 2 % }
77 \def\TMP@EnsureCode#1#2{%
78   \edef\pdftexcmds@AtEnd{%
79     \pdftexcmds@AtEnd
80     \catcode#1 \the\catcode#1\relax
81   }%
82   \catcode#1 #2\relax
83 }
84 \TMP@EnsureCode{33}{12}% !
85 \TMP@EnsureCode{34}{12}% "
86 \TMP@EnsureCode{39}{12}% '
87 \TMP@EnsureCode{40}{12}% (
88 \TMP@EnsureCode{41}{12}% )
89 \TMP@EnsureCode{42}{12}% *
90 \TMP@EnsureCode{43}{12}% +
91 \TMP@EnsureCode{44}{12}% ,
92 \TMP@EnsureCode{45}{12}% -
93 \TMP@EnsureCode{46}{12}% .
94 \TMP@EnsureCode{47}{12}% /
95 \TMP@EnsureCode{58}{12}% :
96 \TMP@EnsureCode{60}{12}% <
97 \TMP@EnsureCode{61}{12}% =
98 \TMP@EnsureCode{62}{12}% >
99 \TMP@EnsureCode{95}{12}% _ (other)
100 \TMP@EnsureCode{126}{12}% ~ (other)

```

## 2.3 Load package infwarerr

```

101 \begingroup\expandafter\expandafter\expandafter\endgroup
102 \expandafter\ifx\csname RequirePackage\endcsname\relax
103   \input infwarerr.sty\relax
104 \else
105   \RequirePackage{infwarerr}[2007/09/09]%
106 \fi

```

## 2.4 Without LuaTeX

```

107 \begingroup\expandafter\expandafter\expandafter\endgroup
108 \expandafter\ifx\csname directlua\endcsname\relax
109   \@PackageInfo{pdftexcmds}{LuaTeX not detected}%
110   \def\pdftexcmds@nopdftex{%
111     \@PackageInfoNoLine{pdftexcmds}{pdfTeX >= 1.30 not detected}%
112     \let\pdftexcmds@nopdftex\relax
113   }%
114   \def\pdftexcmds@temp#1{%
115     \begingroup\expandafter\expandafter\expandafter\endgroup
116     \expandafter\ifx\csname pdf#1\endcsname\relax
117       \pdftexcmds@nopdftex
118     \else
119       \expandafter\def\csname pdf@#1\endcsname\expandafter\endcsname
120       \expandafter##\expandafter{%
121         \csname pdf#1\endcsname
122       }%
123     \fi
124   }%
125   \pdftexcmds@temp{strcmp}%
126   \pdftexcmds@temp{escapehex}%
127   \let\pdf@escapehexnative\pdf@escapehex
128   \pdftexcmds@temp{unescapehex}%
129   \let\pdf@unescapehexnative\pdf@unescapehex
130   \pdftexcmds@temp{escapestring}%
131   \pdftexcmds@temp{escapename}%
132   \pdftexcmds@temp{filesize}%
133   \pdftexcmds@temp{filemoddate}%
134   \begingroup\expandafter\expandafter\expandafter\endgroup
135   \expandafter\ifx\csname pdfshellescape\endcsname\relax
136     \pdftexcmds@nopdftex
137   \else
138     \def\pdf@shellescape{%
139       \pdfshellescape
140     }%
141   \fi
142   \begingroup\expandafter\expandafter\expandafter\endgroup
143   \expandafter\ifx\csname pdffiledump\endcsname\relax
144     \pdftexcmds@nopdftex
145   \else
146     \def\pdf@filedump#1#2#3{%
147       \pdffiledump offset#1 length#2{#3}%
148     }%
149   \fi
150   \begingroup\expandafter\expandafter\expandafter\endgroup
151   \expandafter\ifx\csname pdfmdfivesum\endcsname\relax
152     \pdftexcmds@nopdftex
153   \else
154     \def\pdf@mdfivesum#{\pdfmdfivesum}%
155     \let\pdf@mdfivesumnative\pdf@mdfivesum
156     \def\pdf@filemdfivesum#{\pdfmdfivesum file}%
157   \fi
158   \def\pdf@system#{%
159     \immediate\write18%
160   }%
161   \pdftexcmds@AtEnd
162   \expandafter\endinput
163 \fi

```

## 2.5 Lua functions

\pdftexcmds@luastate

```

164 \chardef\pdftexcmds@luastate=0 %

```

\pdftexcmds@toks

```
165 \begingroup\expandafter\expandafter\expandafter\endgroup
166 \expandafter\ifx\csname newtoks\endcsname\relax
167   \toksdef\pdftexcmds@toks=0 %
168 \else
169   \csname newtoks\endcsname\pdftexcmds@toks
170 \fi

171 \begingroup
172   \escapechar=92 %
173   \edef\{\{\string\}%
174   \catcode35=12 % #
175   \catcode37=12\relax
176   \directlua\pdftexcmds@luastate{
177     pdftexcmds = {}
178     function pdftexcmds.strcmp(A, B)
179       if A == B then
180         tex.write("0")
181       elseif A < B then
182         tex.write("-1")
183       else
184         tex.write("1")
185       end
186     end
187     function pdftexcmds.utf8_to_byte(str)
188       local i = 0
189       local n = string.len(str)
190       local t = {}
191       while i < n do
192         i = i + 1
193         local a = string.byte(str, i)
194         if a < 128 then
195           table.insert(t, string.char(a))
196         else
197           if a >= 192 and i < n then
198             i = i + 1
199             local b = string.byte(str, i)
200             if b < 128 or b >= 192 then
201               i = i - 1
202             elseif a == 194 then
203               table.insert(t, string.char(b))
204             elseif a == 195 then
205               table.insert(t, string.char(b + 64))
206             end
207           end
208         end
209       end
210       return table.concat(t)
211     end
212     function pdftexcmds.escapehex(str, mode)
213       if mode == "byte" then
214         str = pdftexcmds.utf8_to_byte(str)
215       end
216       tex.write((string.gsub(str, ".",
217         function (ch)
218           return string.format("%02X", string.byte(ch))
219         end
220       )))
221     end
```

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

```
222   function pdftexcmds.unescapehex(str, mode)
```

```

223     local a = 0
224     local first = true
225     local result = {}
226     for i = 1, string.len(str), 1 do
227         local ch = string.byte(str, i)
228         if ch >= 48 and ch <= 57 then
229             ch = ch - 48
230         elseif ch >= 65 and ch <= 70 then
231             ch = ch - 55
232         elseif ch >= 97 and ch <= 102 then
233             ch = ch - 87
234         else
235             ch = nil
236         end
237         if ch then
238             if first then
239                 a = ch * 16
240                 first = false
241             else
242                 table.insert(result, a + ch)
243                 first = true
244             end
245         end
246     end
247     if not first then
248         table.insert(result, a)
249     end
250     if mode == "byte" then
251         local utf8 = {}
252         for i, a in ipairs(result) do
253             if a < 128 then
254                 table.insert(utf8, a)
255             else
256                 if a < 192 then
257                     table.insert(utf8, 194)
258                     a = a - 128
259                 else
260                     table.insert(utf8, 195)
261                     a = a - 192
262                 end
263                 table.insert(utf8, a + 128)
264             end
265         end
266         result = utf8
267     end
268     tex.settoks(pdftecmds.toks, string.char(unpack(result)))
269 end

```

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

```

270 function pdftecmds.escapestring(str, mode)
271     if mode == "byte" then
272         str = pdftecmds.utf8_to_byte(str)
273     end
274     tex.write((string.gsub(str, ".",
275         function (ch)
276             local b = string.byte(ch)
277             if b < 33 or b > 126 then
278                 return string.format("\\%.3o", b)
279             end
280             if b == 40 or b == 41 or b == 92 then
281                 return "\\\" .. ch
282             end

```

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



```

283         return nil
284     end
285 )))
286 end

```

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

```

287 function pdftexcmds.escapename(str, mode)
288     if mode == "byte" then
289         str = pdftexcmds.utf8_to_byte(str)
290     end
291     tex.write((string.gsub(str, ".",
292         function (ch)
293             local b = string.byte(ch)
294             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.

```

295         return ""
296     end
297     if b <= 32 or b >= 127
298         or b == 35 or b == 37 or b == 40 or b == 41
299         or b == 47 or b == 60 or b == 62 or b == 91
300         or b == 93 or b == 123 or b == 125 then
301         return string.format("#%.2X", b)
302     else

```

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

```

303         return nil
304     end
305 end
306 )))
307 end
308 function pdftexcmds.filesize(filename)
309     local foundfile = kpse.find_file(filename, "tex", true)
310     if foundfile then
311         local size = lfs.attributes(foundfile, "size")
312         if size then
313             tex.write(size)
314         end
315     end
316 end

```

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

```

317 function pdftexcmds.filemoddate(filename)
318     local foundfile = kpse.find_file(filename, "tex", true)
319     if foundfile then
320         local date = lfs.attributes(foundfile, "modification")
321         if date then
322             local d = os.date("!*t", date)
323             if d.sec >= 60 then
324                 d.sec = 59
325             end
326             local u = os.date("!*t", date)
327             local off = 60 * (d.hour - u.hour) + d.min - u.min
328             if d.year ~= u.year then
329                 if d.year > u.year then
330                     off = off + 1440
331                 else
332                     off = off - 1440
333                 end
334             elseif d.yday ~= u.yday then
335                 if d.yday > u.yday then
336                     off = off + 1440
337                 else

```

```

338         off = off - 1440
339     end
340 end
341 local timezone
342 if off == 0 then
343     timezone = "Z"
344 else
345     local hours = math.floor(off / 60)
346     local mins = math.abs(off - hours * 60)
347     timezone = string.format("%+03d'%02d'", hours, mins)
348 end
349 tex.write(string.format("D:%04d%02d%02d%02d%02d%02d%s",
350     d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
351 end
352 end
353 end
354 function pdftexcmds.filedump(offset, length, filename)
355     length = tonumber(length)
356     if length and length > 0 then
357         local foundfile = kpse.find_file(filename, "tex", true)
358         if foundfile then
359             offset = tonumber(offset)
360             if not offset then
361                 offset = 0
362             end
363             local filehandle = io.open(foundfile, "r")
364             if filehandle then
365                 if offset > 0 then
366                     filehandle:seek("set", offset)
367                 end
368                 local dump = filehandle:read(length)
369                 pdftexcmds.escapehex(dump)
370             end
371         end
372     end
373 end
374 function pdftexcmds.md5sum(str, mode)
375     if mode == "byte" then
376         str = pdftexcmds.utf8_to_byte(str)
377     end
378     pdftexcmds.escapehex(md5.sum(str))
379 end
380 function pdftexcmds.filemd5sum(filename)
381     local foundfile = kpse.find_file(filename, "tex", true)
382     if foundfile then
383         local filehandle = io.open(foundfile, "r")
384         if filehandle then
385             local contents = filehandle:read("*a")
386             pdftexcmds.escapehex(md5.sum(contents))
387         end
388     end
389 end
390 function pdftexcmds.shellescape()
391     if os.execute then
392         tex.write("1")
393     else
394         tex.write("0")
395     end
396 end
397 function pdftexcmds.system(cmdline)
398     pdftexcmds.systemexitstatus = nil
399     texio.write_nl("log", "system(" .. cmdline .. ") ")

```

```

400     if os.execute then
401         texio.write("log", "executed.")
402         pdftexcmds.systemexitstatus = os.execute(cmdline)
403     else
404         texio.write("log", "disabled.")
405     end
406 end
407 function pdftexcmds.lastsystemstatus()
408     local result = tonumber(pdftexcmds.systemexitstatus)
409     if result then
410         local x = math.floor(result / 256)
411         tex.write(result - 256 * math.floor(result / 256))
412     end
413 end
414 function pdftexcmds.lastsystemexit()
415     local result = tonumber(pdftexcmds.systemexitstatus)
416     if result then
417         tex.write(math.floor(result / 256))
418     end
419 end
420 function pdftexcmds.pipe(cmdline)
421     local result
422     pdftexcmds.systemexitstatus = nil
423     texio.write_nl("log", "pipe(" .. cmdline .. ") ")
424     if io.popen then
425         texio.write("log", "executed.")
426         local handle = io.popen(cmdline, "r")
427         if handle then
428             result = handle:read("*a")
429             handle:close()
430         end
431     else
432         texio.write("log", "disabled.")
433     end
434     if result then
435         tex.settoks(pdftexcmds.toks, result)
436     else
437         tex.settoks(pdftexcmds.toks, "")
438     end
439 end
440 }
441 \endgroup

```

\pdf@strcmp

```

442 \long\def\pdf@strcmp#1#2{%
443     \directlua\pdftexcmds@luastate{%
444         pdftexcmds strcmp("\luaescapestring{#1}",%
445             "\luaescapestring{#2}")%
446     }%
447 }%

```

\pdf@escapehex

```

448 \long\def\pdf@escapehex#1{%
449     \directlua\pdftexcmds@luastate{%
450         pdftexcmds.escapehex("\luaescapestring{#1}", "byte")%
451     }%
452 }%

```

\pdf@escapehexnative

```

453 \long\def\pdf@escapehexnative#1{%
454     \directlua\pdftexcmds@luastate{%
455         pdftexcmds.escapehex("\luaescapestring{#1}")%

```

```

456 }%
457 }%

\pdf@unescapehex
458 \def\pdf@unescapehex#1{%
459 \the\expandafter\pdf texcmds@toks
460 \directlua\pdf texcmds@luastate{%
461   pdf texcmds.toks="pdf texcmds@toks"%
462   pdf texcmds.unescapehex("\luaescapestring{#1}", "byte")%
463 }%
464 }%

\pdf@unescapehexnative
465 \def\pdf@unescapehexnative#1{%
466 \the\expandafter\pdf texcmds@toks
467 \directlua\pdf texcmds@luastate{%
468   pdf texcmds.toks="pdf texcmds@toks"%
469   pdf texcmds.unescapehex("\luaescapestring{#1}")%
470 }%
471 }%

\pdf@escapestring
472 \long\def\pdf@escapestring#1{%
473 \directlua\pdf texcmds@luastate{%
474   pdf texcmds.escapestring("\luaescapestring{#1}", "byte")%
475 }%
476 }

\pdf@escapename
477 \long\def\pdf@escapename#1{%
478 \directlua\pdf texcmds@luastate{%
479   pdf texcmds.escapename("\luaescapestring{#1}", "byte")%
480 }%
481 }

\pdf@escapenamenative
482 \long\def\pdf@escapenamenative#1{%
483 \directlua\pdf texcmds@luastate{%
484   pdf texcmds.escapename("\luaescapestring{#1}")%
485 }%
486 }

\pdf@filesize
487 \def\pdf@filesize#1{%
488 \directlua\pdf texcmds@luastate{%
489   pdf texcmds.filesize("\luaescapestring{#1}")%
490 }%
491 }

\pdf@filemoddate
492 \def\pdf@filemoddate#1{%
493 \directlua\pdf texcmds@luastate{%
494   pdf texcmds.filemoddate("\luaescapestring{#1}")%
495 }%
496 }

\pdf@filedump
497 \def\pdf@filedump#1#2#3{%
498 \directlua\pdf texcmds@luastate{%
499   pdf texcmds.filedump("\luaescapestring{\number#1}",%
500     "\luaescapestring{\number#2}",%

```

```

501         "\luaescapestring{#3}")%
502     }%
503 }%

\pdf@mdfivesum
504 \long\def\pdf@mdfivesum#1{%
505     \directlua\pdfTexcmds@luastate{%
506         pdfTexcmds.mdfivesum("\luaescapestring{#1}", "byte")%
507     }%
508 }%

\pdf@mdfivesumnative
509 \long\def\pdf@mdfivesumnative#1{%
510     \directlua\pdfTexcmds@luastate{%
511         pdfTexcmds.mdfivesum("\luaescapestring{#1}")%
512     }%
513 }%

\pdf@filemdfivesum
514 \def\pdf@filemdfivesum#1{%
515     \directlua\pdfTexcmds@luastate{%
516         pdfTexcmds.filemdfivesum("\luaescapestring{#1}")%
517     }%
518 }%

\pdf@shellescape
519 \def\pdf@shellescape{%
520     \directlua\pdfTexcmds@luastate{%
521         pdfTexcmds.shellescape()%
522     }%
523 }

\pdf@system
524 \def\pdf@system#1{%
525     \directlua\pdfTexcmds@luastate{%
526         pdfTexcmds.system("\luaescapestring{#1}")%
527     }%
528 }

\pdf@lastsystemstatus
529 \def\pdf@lastsystemstatus{%
530     \directlua\pdfTexcmds@luastate{%
531         pdfTexcmds.lastsystemstatus()%
532     }%
533 }

\pdf@lastsystemexit
534 \def\pdf@lastsystemexit{%
535     \directlua\pdfTexcmds@luastate{%
536         pdfTexcmds.lastsystemexit()%
537     }%
538 }

\pdf@pipe Check availability of io.popen first.
539 \ifnum0%
540     \directlua\pdfTexcmds@luastate{%
541         if io.popen then %
542             tex.write("1")%
543         end%
544     }%
545     =1 %

```

```

546 \def\pdf@pipe#1{%
547   \the\expandafter\pdftexcmds@toks
548   \directlua\pdftexcmds@luastate{%
549     pdftexcmds.toks="pdftexcmds@toks"%
550     pdftexcmds.pipe("\luaescapestring{#1}")%
551   }%
552 }%
553 \fi

554 \pdftexcmds@AtEnd
555 </package>

```

## 3 Test

### 3.1 Catcode checks for loading

```

556 <test1>

557 \catcode'\{=1 %
558 \catcode'\}=2 %
559 \catcode'\#=6 %
560 \catcode'\@=11 %
561 \expandafter\ifx\csname count@\endcsname\relax
562   \countdef\count@=255 %
563 \fi
564 \expandafter\ifx\csname @gobble\endcsname\relax
565   \long\def\@gobble#1{%
566 \fi
567 \expandafter\ifx\csname @firstofone\endcsname\relax
568   \long\def\@firstofone#1{#1}%
569 \fi
570 \expandafter\ifx\csname loop\endcsname\relax
571   \expandafter\@firstofone
572 \else
573   \expandafter\@gobble
574 \fi
575 {%
576   \def\loop#1\repeat{%
577     \def\body{#1}%
578     \iterate
579   }%
580   \def\iterate{%
581     \body
582     \let\next\iterate
583   \else
584     \let\next\relax
585   \fi
586   \next
587 }%
588 \let\repeat=\fi
589 }%
590 \def\RestoreCatcodes{}
591 \count@=0 %
592 \loop
593   \edef\RestoreCatcodes{%
594     \RestoreCatcodes
595     \catcode\the\count@=\the\catcode\count@\relax
596   }%
597 \ifnum\count@<255 %
598   \advance\count@ 1 %
599 \repeat
600
601 \def\RangeCatcodeInvalid#1#2{%

```

```

602 \count@=#1\relax
603 \loop
604   \catcode\count@=15 %
605   \ifnum\count@<#2\relax
606     \advance\count@ 1 %
607   \repeat
608 }
609 \def\Test{%
610   \RangeCatcodeInvalid{0}{47}%
611   \RangeCatcodeInvalid{58}{64}%
612   \RangeCatcodeInvalid{91}{96}%
613   \RangeCatcodeInvalid{123}{255}%
614   \catcode'\@=12 %
615   \catcode'\=0 %
616   \catcode'\{=1 %
617   \catcode'\}=2 %
618   \catcode'\#=6 %
619   \catcode'\[=12 %
620   \catcode'\]=12 %
621   \catcode'\%=14 %
622   \catcode'\_ =10 %
623   \catcode13=5 %
624   \input pdftexcmds.sty\relax
625   \RestoreCatcodes
626 }
627 \Test
628 \csname @@end\endcsname
629 \end
630 </test1>

```

## 4 Installation

### 4.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

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

*TDS* refers to the standard “A Directory Structure for  $\text{\TeX}$  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):

---

<sup>1</sup><http://ftp.ctan.org/tex-archive/>

```
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- $\text{\TeX}$ :

```
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
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.

### 4.4 Refresh file name databases

If your  $\text{\TeX}$  distribution (te $\text{\TeX}$ , mik $\text{\TeX}$ , ...) relies on file name databases, you must refresh these. For example, te $\text{\TeX}$  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  $\text{\LaTeX}$ .** The `.dtx` chooses its action depending on the format:

**plain- $\text{\TeX}$ :** Run docstrip and extract the files.

**$\text{\LaTeX}$ :** Generate the documentation.

If you insist on using  $\text{\LaTeX}$  for docstrip (really, docstrip does not need  $\text{\LaTeX}$ ), 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 $\text{\LaTeX}$ :

```
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.

## 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	
<code>\#</code> .....	559, 618
<code>\%</code> .....	621
<code>\@</code> .....	560, 614
<code>\@PackageInfo</code> .....	109
<code>\@PackageInfoNoLine</code> .....	111
<code>\@firstofone</code> .....	568, 571
<code>\@gobble</code> .....	565, 573
<code>\[</code> .....	619
<code>\]</code> .....	173, 278, 281, 615
<code>\{</code> .....	557, 616
<code>\}</code> .....	558, 617
<code>\]</code> .....	620
<code>\_</code> .....	622
A	
<code>\advance</code> .....	598, 606
B	
<code>\body</code> .....	577, 581
C	
<code>\catcode</code> .....	3, 4, 5, 6, 7, 18, 19, 20, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 62, 63, 66, 67, 68, 69, 73, 74, 75, 76, 80, 82, 174, 175, 557, 558, 559, 560, 595, 604, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623
<code>\chardef</code> .....	164
<code>\count@</code> .....	562, 591, 595, 597, 598, 602, 604, 605, 606
<code>\countdef</code> .....	562
<code>\csname</code> .	8, 21, 45, 58, 65, 102, 108, 116, 119, 121, 135, 143, 151, 166, 169, 561, 564, 567, 570, 628
D	
<code>\directlua</code> .....	176, 443, 449, 454, 460, 467, 473, 478, 483, 488, 493, 498, 505, 510, 515, 520, 525, 530, 535, 540, 548
E	
<code>\empty</code> .....	12
<code>\end</code> .....	629
<code>\endcsname</code>	8, 21, 45, 58, 65, 102, 108, 116, 119, 121, 135, 143, 151, 166, 169, 561, 564, 567, 570, 628
<code>\endinput</code> .....	30, 162
<code>\escapechar</code> .....	172
I	
<code>\ifcase</code> .....	9
<code>\ifnum</code> .....	539, 597, 605
<code>\ifx</code> .....	10, 12, 21, 45, 53, 102, 108, 116, 135, 143, 151, 166, 561, 564, 567, 570
<code>\immediate</code> .....	23, 47, 159
<code>\input</code> .....	103, 624
<code>\iterate</code> .....	578, 580, 582
L	
<code>\loop</code> .....	576, 592, 603
<code>\luaescapestring</code> .....	. 444, 445, 450, 455, 462, 469, 474, 479, 484, 489, 494, 499, 500, 501, 506, 511, 516, 526, 550
N	
<code>\next</code> .....	582, 584, 586
<code>\number</code> .....	499, 500
P	
<code>\PackageInfo</code> .....	26
<code>\pdf@escapehex</code> .....	2, 127, <u>448</u>
<code>\pdf@escapehexnative</code> .....	127, <u>453</u>
<code>\pdf@escapename</code> .....	<u>477</u>
<code>\pdf@escapenamenative</code> .....	<u>482</u>
<code>\pdf@escapestring</code> .....	<u>472</u>
<code>\pdf@filedump</code> .....	3, 146, <u>497</u>
<code>\pdf@filemdfivesum</code> .....	3, 156, <u>514</u>
<code>\pdf@filemoddate</code> .....	3, <u>492</u>
<code>\pdf@filesize</code> .....	3, <u>487</u>
<code>\pdf@lastsystemexit</code> .....	<u>534</u>
<code>\pdf@lastsystemstatus</code> .....	<u>529</u>
<code>\pdf@mdfivesum</code> .....	3, 154, 155, <u>504</u>
<code>\pdf@mdfivesumnative</code> .....	155, <u>509</u>
<code>\pdf@pipe</code> .....	3, <u>539</u>
<code>\pdf@shellescape</code> .....	3, 138, <u>519</u>
<code>\pdf@strcmp</code> .....	2, <u>442</u>
<code>\pdf@system</code> .....	3, 158, <u>524</u>

<code>\pdf@unescapehex</code> . . . . .	2, 129, 458	601, 610, 611, 612, 613
<code>\pdf@unescapehexnative</code> . .	3, 129, 465	<code>\repeat</code> . . . . . 576, 588, 599, 607
<code>\pdffiledump</code> . . . . .	147	<code>\RequirePackage</code> . . . . . 105
<code>\pdfmdfivesum</code> . . . . .	154, 156	<code>\RestoreCatcodes</code> . . 590, 593, 594, 625
<code>\pdfshellescape</code> . . . . .	139	
<code>\pdftexcmds@AtEnd</code> . . .	78, 79, 161, 554	
<code>\pdftexcmds@luastate</code> 164, 176, 443,		<b>T</b>
449, 454, 460, 467, 473, 478,		<code>\Test</code> . . . . . 609, 627
483, 488, 493, 498, 505, 510,		<code>\the</code> 66, 67, 68, 69, 80, 459, 466, 547, 595
515, 520, 525, 530, 535, 540, 548		<code>\TMP@EnsureCode</code> . . . . . 77,
<code>\pdftexcmds@nopdftex</code> . . . . .		84, 85, 86, 87, 88, 89, 90, 91,
. . . . . 110, 112, 117, 136, 144, 152		92, 93, 94, 95, 96, 97, 98, 99, 100
<code>\pdftexcmds@temp</code> . . . . . 114,		<code>\toksdef</code> . . . . . 167
125, 126, 128, 130, 131, 132, 133		
<code>\pdftexcmds@toks</code> . . 165, 459, 466, 547		<b>W</b>
<code>\ProvidesPackage</code> . . . . . 59		<code>\write</code> . . . . . 23, 47, 159
<b>R</b>		<b>X</b>
<code>\RangeCatcodeInvalid</code> . . . . .		<code>\x</code> 8, 10, 12, 22, 26, 28, 46, 51, 58, 64, 72