

The pdftexcmds package

Heiko Oberdiek*

<heiko.oberdiek at googlemail.com>

2018/09/10 v0.29

Abstract

LuaTeX provides most of the commands of pdfTeX 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	2
1.1	General principles	3
1.2	Macros	3
1.2.1	Strings	3
1.2.2	Files	4
1.2.3	Timekeeping	4
1.2.4	Miscellaneous	5
1.2.5	Additional macro: <code>\pdf@isprimitive</code>	6
1.2.6	Experimental	6
2	Implementation	7
2.1	Reload check and package identification	7
2.2	Catcodes	8
2.3	Load packages	9
2.4	Without LuaTeX	9
2.5	<code>\pdf@primitive</code> , <code>\pdf@ifprimitive</code>	11
2.5.1	Using LuaTeX's <code>tex.enableprimitives</code>	11
2.5.2	Trying various names to find the primitives	11
2.5.3	Result	12
2.6	X _g TeX	12
2.7	<code>\pdf@isprimitive</code>	13
2.8	<code>\pdf@draftmode</code>	14
2.9	Load Lua module	15
2.10	Lua functions	16
2.10.1	Helper macros	16
2.10.2	Strings	17
2.10.3	Files	18
2.10.4	Timekeeping	19
2.10.5	Shell escape	20
2.11	Lua module	21
2.11.1	Strings	21
2.11.2	Files	23
2.11.3	Timekeeping	25
2.11.4	Miscellaneous	25

*Please report any issues at <https://github.com/ho-tex/oberdiek/issues>

3	Test	27
3.1	Catcode checks for loading	27
3.2	Test for <code>\pdf@isprimitive</code>	28
3.3	Test for <code>\pdf@shellescape</code>	29
3.4	Test for escape functions	30
4	Installation	32
4.1	Download	32
4.2	Bundle installation	33
4.3	Package installation	33
4.4	Refresh file name databases	33
4.5	Some details for the interested	33
5	Catalogue	34
6	History	34
	[2007/11/11 v0.1]	34
	[2007/11/12 v0.2]	34
	[2007/12/12 v0.3]	35
	[2009/04/10 v0.4]	35
	[2009/09/22 v0.5]	35
	[2009/09/23 v0.6]	35
	[2009/12/12 v0.7]	35
	[2010/03/01 v0.8]	35
	[2010/04/01 v0.9]	35
	[2010/11/04 v0.10]	35
	[2010/11/11 v0.11]	35
	[2011/01/30 v0.12]	35
	[2011/03/04 v0.13]	35
	[2011/04/10 v0.14]	35
	[2011/04/16 v0.15]	35
	[2011/04/22 v0.16]	36
	[2011/06/29 v0.17]	36
	[2011/07/01 v0.18]	36
	[2011/07/28 v0.19]	36
	[2011/11/29 v0.20]	36
	[2016/05/10 v0.21]	36
	[2016/05/21 v0.22]	36
	[2016/10/02 v0.23]	36
	[2017/01/29 v0.24]	36
	[2017/03/19 v0.25]	36
	[2018/01/21 v0.26]	36
	[2018/01/30 v0.27]	36
	[2018/09/07 v0.28]	36
	[2018/09/10 v0.29]	36
7	Index	37

1 Documentation

Some primitives of pdfTeX [[pdftex-manual](#)] are not defined by LuaTeX [[luatex-manual](#)]. This package implements macro based solutions using Lua code for the following missing pdfTeX primitives;

- `\pdfstrcmp`
- `\pdfunescapehex`
- `\pdfescapehex`
- `\pdfescapename`

- `\pdfescapestring`
- `\pdffilesize`
- `\pdffilemoddate`
- `\pdffiledump`
- `\pdfmdfivesum`
- `\pdfresettimer`
- `\pdfelapsedtime`
- `\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 $\langle general\ text \rangle$ for the other arguments. Using token registers assignments, $\langle general\ text \rangle$ could be caught. However, the simulated primitives are expandable and register assignments would destroy this important property. ($\langle general\ text \rangle$ 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}

```

LuaTeX 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@ $\langle cmd \rangle$` if pdfTeX provides `\pdf $\langle cmd \rangle$` .

Arguments: The order of arguments in `\pdf@ $\langle cmd \rangle$` is the same as for the corresponding primitive of pdfTeX. The arguments are ordinary undelimited TeX arguments, no $\langle general\ text \rangle$ and without additional keywords.

Expandibility: The macro `\pdf@ $\langle cmd \rangle$` is expandable if the corresponding pdfTeX primitive has this property. Exact two expansion steps are necessary (first is the macro expansion) except for `\pdf@primitive` and `\pdf@ifprimitive`. The latter ones are not macros, but have the direct meaning of the primitive.

Without LuaTeX: The macros `\pdf@ $\langle cmd \rangle$` are mapped to the commands of pdfTeX if they are available. Otherwise they are undefined.

Availability: The macros that the packages provides are undefined, if the necessary primitives are not found and cannot be implemented by Lua.

1.2 Macros

1.2.1 Strings [pdftex-manual]

<code>\pdf@strcmp {$\langle stringA \rangle$} {$\langle stringB \rangle$}</code>
--

Same as `\pdfstrcmp{ $\langle stringA \rangle$ }{ $\langle stringB \rangle$ }`.

`\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 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.

1.2.2 Files [pdftex-manual]

`\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⟩}`.

1.2.3 Timekeeping [pdftex-manual]

The timekeeping macros are based on Andy Thomas' work [**AndyThomas:Analog**].

`\pdf@resettimer`

Same as `\pdfresettimer`, it resets the internal timer.

`\pdf@elapsedtime`

Same as `\pdfelapsedtime`. It behaves like a read-only integer. For printing purposes it can be prefixed by `\the` or `\number`. It measures the time in scaled seconds (seconds multiplied with 65536) since the latest call of `\pdf@resettimer` or start of program/package. The resolution, the shortest time interval that can be measured, depends on the program and system.

- pdfTeX with `gettimeofday`: $\geq 1/65536$ s

- pdfTeX with `ftime`: ≥ 1 ms
- pdfTeX with `time`: ≥ 1 s
- LuaTeX: ≥ 10 ms
(`os.clock()` returns a float number with two decimal digits in LuaTeX beta-0.70.1-2011061416 (rev 4277)).

1.2.4 Miscellaneous [pdfTeX-manual]

`\pdf@draftmode`

If the TeX compiler knows `\pdfdraftmode` or `\draftmode` (pdfTeX, LuaTeX), then `\pdf@draftmode` returns, whether this mode is enabled. The result is an implicit number: one means the draft mode is available and enabled. If the value is zero, then the mode is not active or `\pdfdraftmode` is not available. An explicit number is yielded by `\number\pdf@draftmode`. The macro cannot be used to change the mode, see `\pdf@setdraftmode`.

`\pdf@ifdraftmode {<true>} {<false>}`

If `\pdfdraftmode` is available and enabled, `<true>` is called, otherwise `<false>` is executed.

`\pdf@setdraftmode {<value>}`

Macro `\pdf@setdraftmode` expects the number zero or one as `<value>`. Zero deactivates the mode and one enables the draft mode. The macro does not have an effect, if the feature `\pdfdraftmode` is not available.

`\pdf@shellescape`

Same as `\pdfshellescape`. It is or 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.

In LuaTeX before 0.68.0 `\pdf@shellescape` is not available due to a bug in `os.execute()`. The argumentless form crashes in some circumstances with segmentation fault. (It is fixed in version 0.68.0 or revision 4167 of LuaTeX. and packported to some version of 0.67.0).

Hints for usage:

- Before its use `\pdf@shellescape` should be tested, whether it is available. Example with package `ltxcmds` (loaded by package `pdfTeXcmds`):

```
\ltx@ifundefined{pdf@shellescape}{%
  % \pdf@shellescape is undefined
}%{
  % \pdf@shellescape is available
}
```

Use `\ltx@ifundefined` in expandable contexts.

- `\pdf@shellescape` might be a numerical constant, expands to the primitive, or expands to a plain number. Therefore use it in contexts where these differences does not matter.
- Use in comparisons, e.g.:

`\ifnum\pdf@shellescape=0 ...`

- Print the number: `\number\pdf@shellescape`

`\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?

`\pdf@primitive \cmd`

Same as `\pdfprimitive` in pdfTeX or LuaTeX. In XeTeX the primitive is called `\primitive`. Despite the current definition of the command `\cmd`, it's meaning as primitive is used.

`\pdf@ifprimitive \cmd`

Same as `\ifpdfprimitive` in pdfTeX or LuaTeX. XeTeX calls it `\ifprimitive`. It is a switch that checks if the command `\cmd` has it's primitive meaning.

1.2.5 Additional macro: `\pdf@isprimitive`

`\pdf@isprimitive \cmd1 \cmd2 {⟨true⟩} {⟨false⟩}`

If `\cmd1` has the primitive meaning given by the primitive name of `\cmd2`, then the argument `⟨true⟩` is executed, otherwise `⟨false⟩`. The macro `\pdf@isprimitive` is expandable. Internally it checks the result of `\meaning` and is therefore available for all TeX variants, even the original TeX. Example with L^ATeX:

```
\makeatletter
\pdf@isprimitive{@@input}{input}{%
  \typeout{\string\@@input\space is original\string\input}%
}{%
  \typeout{Oops, \string\@@input\space is not the %
    original\string\input}%
}
```

1.2.6 Experimental

`\pdf@unescapehexnative {⟨string⟩}`
`\pdf@escapehexnative {⟨string⟩}`
`\pdf@escapenamename {⟨string⟩}`
`\pdf@cmdfivesumnative {⟨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 documen-

tation 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\catcode61\catcode48\catcode32=10\relax%
3 \catcode13=5 % ^^M
4 \endlinechar=13 %
5 \catcode35=6 % #
6 \catcode39=12 % '
7 \catcode44=12 % ,
8 \catcode45=12 % -
9 \catcode46=12 % .
10 \catcode58=12 % :
11 \catcode64=11 % @
12 \catcode123=1 % {
13 \catcode125=2 % }
14 \expandafter\let\expandafter\x\csname ver@pdftexcmds.sty\endcsname
15 \ifx\x\relax % plain-TeX, first loading
16 \else
17 \def\empty{}%
18 \ifx\x\empty % LaTeX, first loading,
19 % variable is initialized, but \ProvidesPackage not yet seen
20 \else
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{pdftexcmds}{The package is already loaded}%
29 \aftergroup\endinput
30 \fi
31 \fi
32 \endgroup%
```

Package identification:

```
33 \begingroup\catcode61\catcode48\catcode32=10\relax%
34 \catcode13=5 % ^^M
35 \endlinechar=13 %
36 \catcode35=6 % #
37 \catcode39=12 % '
38 \catcode40=12 % (
39 \catcode41=12 % )
40 \catcode44=12 % ,
41 \catcode45=12 % -
42 \catcode46=12 % .
43 \catcode47=12 % /
44 \catcode58=12 % :
45 \catcode64=11 % @
46 \catcode91=12 % [
47 \catcode93=12 % ]
48 \catcode123=1 % {
49 \catcode125=2 % }
50 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
51 \def\x#1#2#3[#4]{\endgroup
```

```

52 \immediate\write-1{Package: #3 #4}%
53 \xdef#1{#4}%
54 }%
55 \else
56 \def\x#1#2[#3]{\endgroup
57 #2[#3]}%
58 \ifx#1\@undefined
59 \xdef#1{#3}%
60 \fi
61 \ifx#1\relax
62 \xdef#1{#3}%
63 \fi
64 }%
65 \fi
66 \expandafter\x\csname ver@pdftexcmds.sty\endcsname
67 \ProvidesPackage{pdftexcmds}%
68 [2018/09/10 v0.29 Utility functions of pdfTeX for LuaTeX (HO)]%

```

2.2 Catcodes

```

69 \begingroup\catcode61\catcode48\catcode32=10\relax%
70 \catcode13=5 % ^^M
71 \endlinechar=13 %
72 \catcode123=1 % {
73 \catcode125=2 % }
74 \catcode64=11 % @
75 \def\x{\endgroup
76 \expandafter\edef\csname pdftexcmds@AtEnd\endcsname{%
77 \endlinechar=\the\endlinechar\relax
78 \catcode13=\the\catcode13\relax
79 \catcode32=\the\catcode32\relax
80 \catcode35=\the\catcode35\relax
81 \catcode61=\the\catcode61\relax
82 \catcode64=\the\catcode64\relax
83 \catcode123=\the\catcode123\relax
84 \catcode125=\the\catcode125\relax
85 }%
86 }%
87 \x\catcode61\catcode48\catcode32=10\relax%
88 \catcode13=5 % ^^M
89 \endlinechar=13 %
90 \catcode35=6 % #
91 \catcode64=11 % @
92 \catcode123=1 % {
93 \catcode125=2 % }
94 \def\TMP@EnsureCode#1#2{%
95 \edef\pdftexcmds@AtEnd{%
96 \pdftexcmds@AtEnd
97 \catcode#1=\the\catcode#1\relax
98 }%
99 \catcode#1=#2\relax
100 }
101 \TMP@EnsureCode{0}{12}%
102 \TMP@EnsureCode{1}{12}%
103 \TMP@EnsureCode{2}{12}%
104 \TMP@EnsureCode{10}{12}% ^^J
105 \TMP@EnsureCode{33}{12}% !
106 \TMP@EnsureCode{34}{12}% "
107 \TMP@EnsureCode{38}{4}% &
108 \TMP@EnsureCode{39}{12}% '
109 \TMP@EnsureCode{40}{12}% (
110 \TMP@EnsureCode{41}{12}% )

```



```

111 \TMP@EnsureCode{42}{12}% *
112 \TMP@EnsureCode{43}{12}% +
113 \TMP@EnsureCode{44}{12}% ,
114 \TMP@EnsureCode{45}{12}% -
115 \TMP@EnsureCode{46}{12}% .
116 \TMP@EnsureCode{47}{12}% /
117 \TMP@EnsureCode{58}{12}% :
118 \TMP@EnsureCode{60}{12}% <
119 \TMP@EnsureCode{62}{12}% >
120 \TMP@EnsureCode{91}{12}% [
121 \TMP@EnsureCode{93}{12}% ]
122 \TMP@EnsureCode{94}{7}% ^ (superscript)
123 \TMP@EnsureCode{95}{12}% _ (other)
124 \TMP@EnsureCode{96}{12}% `
125 \TMP@EnsureCode{126}{12}% ~ (other)
126 \edef\pdftexcmds@AtEnd{%
127   \pdftexcmds@AtEnd
128   \escapechar=\number\escapechar\relax
129   \noexpand\endinput
130 }
131 \escapechar=92 %

```

2.3 Load packages

```

132 \begingroup\expandafter\expandafter\expandafter\endgroup
133 \expandafter\ifx\csname RequirePackage\endcsname\relax
134   \def\TMP@RequirePackage#1[#2]{%
135     \begingroup\expandafter\expandafter\expandafter\endgroup
136     \expandafter\ifx\csname ver@#1.sty\endcsname\relax
137       \input #1.sty\relax
138     \fi
139   }%
140   \TMP@RequirePackage{infwarerr}[2007/09/09]%
141   \TMP@RequirePackage{ifluatex}[2010/03/01]%
142   \TMP@RequirePackage{ltxcmds}[2010/12/02]%
143   \TMP@RequirePackage{ifpdf}[2010/09/13]%
144 \else
145   \RequirePackage{infwarerr}[2007/09/09]%
146   \RequirePackage{ifluatex}[2010/03/01]%
147   \RequirePackage{ltxcmds}[2010/12/02]%
148   \RequirePackage{ifpdf}[2010/09/13]%
149 \fi

```

2.4 Without LuaTeX

```

150 \ifluatex
151 \else
152   \@PackageInfoNoLine{pdftexcmds}{LuaTeX not detected}%
153   \def\pdftexcmds@nopdftex{%
154     \@PackageInfoNoLine{pdftexcmds}{pdfTeX >= 1.30 not detected}%
155     \let\pdftexcmds@nopdftex\relax
156   }%
157   \def\pdftexcmds@temp#1{%
158     \begingroup\expandafter\expandafter\expandafter\endgroup
159     \expandafter\ifx\csname pdf#1\endcsname\relax
160       \pdftexcmds@nopdftex
161     \else
162       \expandafter\def\csname pdf@#1\endcsname
163       \expandafter##\expandafter{%
164         \csname pdf#1\endcsname
165       }%
166     \fi
167   }%
168   \pdftexcmds@temp{strcmp}%

```

```

169 \pdfdoccmds@temp{escapehex}%
170 \let\pdf@escapehexnative\pdf@escapehex
171 \pdfdoccmds@temp{unescapehex}%
172 \let\pdf@unescapehexnative\pdf@unescapehex
173 \pdfdoccmds@temp{escapestring}%
174 \pdfdoccmds@temp{escapename}%
175 \pdfdoccmds@temp{filesize}%
176 \pdfdoccmds@temp{filemoddate}%
177 \begingroup\expandafter\expandafter\expandafter\endgroup
178 \expandafter\ifx\csname pdfshellescape\endcsname\relax
179 \pdfdoccmds@nopdftex
180 \ltx@ifundefined{pdftexversion}{%
181 }{%
182 \ifnum\pdftexversion>120 % 1.21a supports \ifeof18
183 \ifeof18 %
184 \chardef\pdf@shellescape=0 %
185 \else
186 \chardef\pdf@shellescape=1 %
187 \fi
188 \fi
189 }%
190 \else
191 \def\pdf@shellescape{%
192 \pdfshellescape
193 }%
194 \fi
195 \begingroup\expandafter\expandafter\expandafter\endgroup
196 \expandafter\ifx\csname pdffiledump\endcsname\relax
197 \pdfdoccmds@nopdftex
198 \else
199 \def\pdf@filedump#1#2#3{%
200 \pdffiledump offset#1 length#2{#3}%
201 }%
202 \fi
203 \begingroup\expandafter\expandafter\expandafter\endgroup
204 \expandafter\ifx\csname pdfmdfivesum\endcsname\relax
205 \begingroup\expandafter\expandafter\expandafter\endgroup
206 \expandafter\ifx\csname mdfivesum\endcsname\relax
207 \pdfdoccmds@nopdftex
208 \else
209 \def\pdf@mdfivesum#\mdfivesum{%
210 \let\pdf@mdfivesumnative\pdf@mdfivesum
211 \def\pdf@filemdfivesum#\mdfivesum file}%
212 \fi
213 \else
214 \def\pdf@mdfivesum#\pdfmdfivesum{%
215 \let\pdf@mdfivesumnative\pdf@mdfivesum
216 \def\pdf@filemdfivesum#\pdfmdfivesum file}%
217 \fi
218 \def\pdf@system#{%
219 \immediate\write18%
220 }%
221 \def\pdfdoccmds@temp#1{%
222 \begingroup\expandafter\expandafter\expandafter\endgroup
223 \expandafter\ifx\csname pdf#1\endcsname\relax
224 \pdfdoccmds@nopdftex
225 \else
226 \expandafter\let\csname pdf@#1\endcsname
227 \csname pdf#1\endcsname
228 \fi
229 }%
230 \pdfdoccmds@temp{resettimer}%

```

```
231 \pdf texcmds@temp{elapsedtime}%
232 \fi
```

2.5 \pdf@primitive, \pdf@ifprimitive

Since version 1.40.0 pdf_{TEX} has \pdfprimitive and \ifpdfprimitive. And \pdfprimitive was fixed in version 1.40.4.

X_{TEX} provides them under the name \primitive and \ifprimitive. Lua_{TEX} knows both name variants, but they have possibly to be enabled first (tex.enableprimitives).

Depending on the format TeX Live uses a prefix luatex.

Caution: \let must be used for the definition of the macros, especially because of \ifpdfprimitive.

2.5.1 Using Lua_{TEX}'s tex.enableprimitives

```
233 \ifluatex

\pdf texcmds@directlua

234 \ifnum\luatexversion<36 %
235 \def\pdf texcmds@directlua{\directlua0 }%
236 \else
237 \let\pdf texcmds@directlua\directlua
238 \fi

239 \begingroup
240 \newlinechar=10 %
241 \endlinechar=\newlinechar
242 \pdf texcmds@directlua{%
243 if tex.enableprimitives then
244 tex.enableprimitives(
245 'pdf@',
246 {'primitive', 'ifprimitive', 'pdfdraftmode', 'draftmode'}
247 )
248 tex.enableprimitives('', {'luaescapestring'})
249 end
250 }%
251 \endgroup %

252 \fi
```

2.5.2 Trying various names to find the primitives

```
\pdf texcmds@strip@prefix

253 \def\pdf texcmds@strip@prefix#1>{}

254 \def\pdf texcmds@temp#1#2#3{%
255 \begingroup\expandafter\expandafter\expandafter\endgroup
256 \expandafter\ifx\csname pdf@#1\endcsname\relax
257 \begingroup
258 \def\x{#3}%
259 \edef\x{\expandafter\pdf texcmds@strip@prefix\meaning\x}%
260 \escapechar=-1 %
261 \edef\y{\expandafter\meaning\csname#2\endcsname}%
262 \expandafter\endgroup
263 \ifx\x\y
264 \expandafter\let\csname pdf@#1\expandafter\endcsname
265 \csname #2\endcsname
266 \fi
267 \fi
268 }
```

\pdf@primitive

```

269 \pdftexcmds@temp{primitive}{pdfprimitive}{pdfprimitive}% pdfTeX, oldLuaTeX
270 \pdftexcmds@temp{primitive}{primitive}{primitive}% XeTeX, luatex
271 \pdftexcmds@temp{primitive}{luatexprimitive}{pdfprimitive}% oldLuaTeX
272 \pdftexcmds@temp{primitive}{luatexpdfprimitive}{pdfprimitive}% oldLuaTeX

```

`\pdf@ifprimitive`

```

273 \pdftexcmds@temp{ifprimitive}{ifpdfprimitive}{ifpdfprimitive}% pdfTeX, oldLu-
    aTeX
274 \pdftexcmds@temp{ifprimitive}{ifprimitive}{ifprimitive}% XeTeX, luatex
275 \pdftexcmds@temp{ifprimitive}{luatexifprimitive}{ifpdfprimitive}% oldLuaTeX
276 \pdftexcmds@temp{ifprimitive}{luatexpdfprimitive}{ifpdfprimitive}% oldLuaTeX

```

Disable broken `\pdfprimitive`.

```

277 \ifuatex\else
278 \begingroup
279 \expandafter\ifx\csname pdf@primitive\endcsname\relax
280 \else
281 \expandafter\ifx\csname pdftexversion\endcsname\relax
282 \else
283 \ifnum\pdftexversion=140 %
284 \expandafter\ifx\csname pdftexrevision\endcsname\relax
285 \else
286 \ifnum\pdftexrevision<4 %
287 \endgroup
288 \let\pdf@primitive\@undefined
289 \@PackageInfoNoLine{pdftexcmds}{%
290 \string\pdf@primitive\space disabled, %
291 because\MessageBreak
292 \string\pdfprimitive\space is broken until pdfTeX 1.40.4%
293 }%
294 \begingroup
295 \fi
296 \fi
297 \fi
298 \fi
299 \fi
300 \endgroup
301 \fi

```

2.5.3 Result

```

302 \begingroup
303 \@PackageInfoNoLine{pdftexcmds}{%
304 \string\pdf@primitive\space is %
305 \expandafter\ifx\csname pdf@primitive\endcsname\relax not \fi
306 available%
307 }%
308 \@PackageInfoNoLine{pdftexcmds}{%
309 \string\pdf@ifprimitive\space is %
310 \expandafter\ifx\csname pdf@ifprimitive\endcsname\relax not \fi
311 available%
312 }%
313 \endgroup

```

2.6 X_gTeX

Look for primitives `\shellescape`, `\stricmp`.

```

314 \def\pdftexcmds@temp#1{%
315 \begingroup\expandafter\expandafter\expandafter\endgroup
316 \expandafter\ifx\csname pdf@#1\endcsname\relax
317 \begingroup
318 \escapechar=-1 %
319 \edef\x{\expandafter\meaning\csname#1\endcsname}%

```

```

320 \def\y{#1}%
321 \def\z##1->{}%
322 \edef\y{\expandafter\z\meaning\y}%
323 \expandafter\endgroup
324 \ifx\x\y
325 \expandafter\def\csname pdf@#1\expandafter\endcsname
326 \expandafter{%
327 \csname#1\endcsname
328 }%
329 \fi
330 \fi
331 }%
332 \pdfdocmds@temp{shellescape}%
333 \pdfdocmds@temp{strcmp}%

```

2.7 \pdf@isprimitive

```

334 \def\pdf@isprimitive{%
335 \begingroup\expandafter\expandafter\expandafter\endgroup
336 \expandafter\ifx\csname pdf@strcmp\endcsname\relax
337 \long\def\pdf@isprimitive###1{%
338 \expandafter\pdfdocmds@isprimitive\expandafter{\meaning###1}%
339 }%
340 \long\def\pdfdocmds@isprimitive##1##2{%
341 \expandafter\pdfdocmds@isprimitive\expandafter{\string##2}{##1}%
342 }%
343 \def\pdfdocmds@isprimitive##1##2{%
344 \ifnum0\pdfdocmds@equal##1\delimiter##2\delimiter=1 %
345 \expandafter\ltx@firstoftwo
346 \else
347 \expandafter\ltx@secondoftwo
348 \fi
349 }%
350 \def\pdfdocmds@equal##1##2\delimiter##3##4\delimiter{%
351 \if##1##3%
352 \ifx\relax##2##4\relax
353 1%
354 \else
355 \ifx\relax##2\relax
356 \else
357 \ifx\relax##4\relax
358 \else
359 \pdfdocmds@equalcont{##2}{##4}%
360 \fi
361 \fi
362 \fi
363 \fi
364 }%
365 \def\pdfdocmds@equalcont###1{%
366 \def\pdfdocmds@equalcont####1####2##1##1##1##1##1{%
367 ##1##1##1##1%
368 \pdfdocmds@equal####1\delimiter####2\delimiter
369 }%
370 }%
371 \expandafter\pdfdocmds@equalcont\csname fi\endcsname
372 \else
373 \long\def\pdf@isprimitive##1##2{%
374 \ifnum\pdf@strcmp{\meaning##1}{\string##2}=0 %
375 \expandafter\ltx@firstoftwo
376 \else
377 \expandafter\ltx@secondoftwo
378 \fi

```

```

379 }%
380 \fi
381 }
382 \ifuatex
383 \ifx\pdfdraftmode\@undefined
384 \let\pdfdraftmode\draftmode
385 \fi
386 \else
387 \pdf@isprimitive
388 \fi

```

2.8 \pdf@draftmode

```

389 \let\pdftexcmds@temp\ltx@zero %
390 \ltx@ifUndefined{pdfdraftmode}{%
391 \@PackageInfoNoLine{pdftexcmds}{\ltx@backslashchar pdfdraftmode not found}%
392 }{%
393 \ifpdf
394 \let\pdftexcmds@temp\ltx@one
395 \@PackageInfoNoLine{pdftexcmds}{\ltx@backslashchar pdfdraftmode found}%
396 \else
397 \@PackageInfoNoLine{pdftexcmds}{%
398 \ltx@backslashchar pdfdraftmode is ignored in DVI mode%
399 }%
400 \fi
401 }
402 \ifcase\pdftexcmds@temp

```

\pdf@draftmode

```
403 \let\pdf@draftmode\ltx@zero
```

\pdf@ifdraftmode

```
404 \let\pdf@ifdraftmode\ltx@secondoftwo
```

\pdftexcmds@setdraftmode

```
405 \def\pdftexcmds@setdraftmode#1{%

```

```
406 \else
```

\pdftexcmds@draftmode

```
407 \let\pdftexcmds@draftmode\pdfdraftmode
```

\pdf@ifdraftmode

```

408 \def\pdf@ifdraftmode{%
409 \ifnum\pdftexcmds@draftmode=\ltx@one
410 \expandafter\ltx@firstoftwo
411 \else
412 \expandafter\ltx@secondoftwo
413 \fi
414 }%

```

\pdf@draftmode

```

415 \def\pdf@draftmode{%
416 \ifnum\pdftexcmds@draftmode=\ltx@one
417 \expandafter\ltx@one
418 \else
419 \expandafter\ltx@zero
420 \fi
421 }%

```

\pdftexcmds@setdraftmode

```

422 \def\pdftexcmds@setdraftmode#1{%
423 \pdftexcmds@draftmode=#1\relax
424 }%

```

425 \fi

\pdf@setdraftmode

```
426 \def\pdf@setdraftmode#1{%
427   \begingroup
428   \count\ltx@cclv=#1\relax
429   \edef\x{\endgroup
430     \noexpand\pdftexcmds@@setdraftmode{\the\count\ltx@cclv}%
431   }%
432   \x
433 }
```

\pdftexcmds@@setdraftmode

```
434 \def\pdftexcmds@@setdraftmode#1{%
435   \ifcase#1 %
436     \pdftexcmds@setdraftmode{#1}%
437   \or
438     \pdftexcmds@setdraftmode{#1}%
439   \else
440     \@PackageWarning{pdftexcmds}{%
441       \string\pdf@setdraftmode: Ignoring\MessageBreak
442       invalid value `#1'%
443     }%
444   \fi
445 }
```

2.9 Load Lua module

```
446 \ifluatex
447 \else
448   \expandafter\pdftexcmds@AtEnd
449 \fi%

450 \ifnum\luatexversion<80
451   \begingroup\expandafter\expandafter\expandafter\endgroup
452   \expandafter\ifx\csname RequirePackage\endcsname\relax
453     \def\TMP@RequirePackage#1[#2]{%
454       \begingroup\expandafter\expandafter\expandafter\endgroup
455       \expandafter\ifx\csname ver@#1.sty\endcsname\relax
456         \input #1.sty\relax
457       \fi
458     }%
459     \TMP@RequirePackage{luatex-loader}[2009/04/10]%
460   \else
461     \RequirePackage{luatex-loader}[2009/04/10]%
462   \fi
463 \fi
464 \pdftexcmds@directlua{%
465   require("pdftexcmds")%
466 }
467 \ifnum\luatexversion>37 %
468   \ifnum0%
469     \pdftexcmds@directlua{%
470       if status.ini_version then %
471         tex.write("1")%
472       end%
473     }>0 %
474   \everyjob\expandafter{%
475     \the\everyjob
476     \pdftexcmds@directlua{%
477       require("pdftexcmds")%
478     }%
479   }%
```

```

480 \fi
481 \fi
482 \begingroup
483 \def\x{2018/09/10 v0.29}%
484 \ltx@onelevel@sanitize\x
485 \edef\y{%
486 \pdfTexcmds@directlua{%
487   if oberdiek.pdfTexcmds.getVersion then %
488     oberdiek.pdfTexcmds.getVersion()%
489   end%
490 }%
491 }%
492 \ifx\x\y
493 \else
494 \@PackageError{pdfTexcmds}{%
495   Wrong version of lua module.\MessageBreak
496   Package version: \x\MessageBreak
497   Lua module: \y
498 } \@ehc
499 \fi
500 \endgroup

```

2.10 Lua functions

2.10.1 Helper macros

`\pdfTexcmds@toks`

```

501 \begingroup\expandafter\expandafter\expandafter\endgroup
502 \expandafter\ifx\csname newtoks\endcsname\relax
503 \toksdef\pdfTexcmds@toks=0 %
504 \else
505 \csname newtoks\endcsname\pdfTexcmds@toks
506 \fi

```

`\pdfTexcmds@Patch`

```

507 \def\pdfTexcmds@Patch{0}
508 \ifnum\luatexversion>40 %
509 \ifnum\luatexversion<66 %
510 \def\pdfTexcmds@Patch{1}%
511 \fi
512 \fi

513 \ifcase\pdfTexcmds@Patch
514 \catcode`\&=14 %
515 \else
516 \catcode`\&=9 %

```

`\pdfTexcmds@PatchDecode`

```

517 \def\pdfTexcmds@PatchDecode#1\@nil{%
518 \pdfTexcmds@DecodeA#1^^A^^A\@nil{}%
519 }%

```

`\pdfTexcmds@DecodeA`

```

520 \def\pdfTexcmds@DecodeA#1^^A^^A#2\@nil#3{%
521 \ifx\relax#2\relax
522 \ltx@ReturnAfterElseFi{%
523 \pdfTexcmds@DecodeB#3#1^^A^^B\@nil{}%
524 }%
525 \else
526 \ltx@ReturnAfterFi{%
527 \pdfTexcmds@DecodeA#2\@nil{#3#1^^@}%
528 }%
529 \fi
530 }%

```


`\pdfTexcmds@DecodeB`

```
531 \def\pdfTexcmds@DecodeB#1^^A^^B#2\@nil#3{%
532   \ifx\relax#2\relax%
533     \ltx@ReturnAfterElseFi{%
534       \ltx@zero
535       #3#1%
536     }%
537   \else
538     \ltx@ReturnAfterFi{%
539       \pdfTexcmds@DecodeB#2\@nil{#3#1^^A}%
540     }%
541   \fi
542 }%

543 \fi

544 \ifnum\luatexversion<36 %
545 \else
546   \catcode`\0=9 %
547 \fi
```

2.10.2 Strings [pdfTeX-manual]

`\pdf@strcmp`

```
548 \long\def\pdf@strcmp#1#2{%
549   \directlua0{%
550     oberdiek.pdfTexcmds.strcmp("\luaescapestring{#1}",%
551       "\luaescapestring{#2}")%
552   }%
553 }%

554 \pdf@isprimitive
```

`\pdf@escapehex`

```
555 \long\def\pdf@escapehex#1{%
556   \directlua0{%
557     oberdiek.pdfTexcmds.escapehex("\luaescapestring{#1}", "byte")%
558   }%
559 }%
```

`\pdf@escapehexnative`

```
560 \long\def\pdf@escapehexnative#1{%
561   \directlua0{%
562     oberdiek.pdfTexcmds.escapehex("\luaescapestring{#1}")%
563   }%
564 }%
```

`\pdf@unescapehex`

```
565 \def\pdf@unescapehex#1{%
566   & \romannumeral\expandafter\pdfTexcmds@PatchDecode
567   \the\expandafter\pdfTexcmds@toks
568   \directlua0{%
569     oberdiek.pdfTexcmds.toks="pdfTexcmds@toks"%
570     oberdiek.pdfTexcmds.unescapehex("\luaescapestring{#1}", "byte", \pdfTex-
571       cmds@Patch)%
572   }%
573   & \@nil
574 }%
```

`\pdf@unescapehexnative`

```
574 \def\pdf@unescapehexnative#1{%
575   & \romannumeral\expandafter\pdfTexcmds@PatchDecode
576   \the\expandafter\pdfTexcmds@toks
```

```

577 \directlua0{%
578   oberdiek.pdfdoccmds.toks="pdfdoccmds@toks"%
579   oberdiek.pdfdoccmds.unescapehex("\luaescapestring{#1}", \pdfdoccmds@Patch)%
580 }%
581 & \@nil
582 }%

```

\pdf@escapestring

```

583 \long\def\pdf@escapestring#1{%
584   \directlua0{%
585     oberdiek.pdfdoccmds.escapestring("\luaescapestring{#1}", "byte")%
586   }%
587 }

```

\pdf@escapename

```

588 \long\def\pdf@escapename#1{%
589   \directlua0{%
590     oberdiek.pdfdoccmds.escapename("\luaescapestring{#1}", "byte")%
591   }%
592 }

```

\pdf@escapenamename

```

593 \long\def\pdf@escapenamename#1{%
594   \directlua0{%
595     oberdiek.pdfdoccmds.escapename("\luaescapestring{#1}")%
596   }%
597 }

```

2.10.3 Files [pdfdoc-manual]

\pdf@filesize

```

598 \def\pdf@filesize#1{%
599   \directlua0{%
600     oberdiek.pdfdoccmds.filesize("\luaescapestring{#1}")%
601   }%
602 }

```

\pdf@filemoddate

```

603 \def\pdf@filemoddate#1{%
604   \directlua0{%
605     oberdiek.pdfdoccmds.filemoddate("\luaescapestring{#1}")%
606   }%
607 }

```

\pdf@filedump

```

608 \def\pdf@filedump#1#2#3{%
609   \directlua0{%
610     oberdiek.pdfdoccmds.filedump("\luaescapestring{\number#1}",%
611       "\luaescapestring{\number#2}",%
612       "\luaescapestring{#3}")%
613   }%
614 }%

```

\pdf@mdfivesum

```

615 \long\def\pdf@mdfivesum#1{%
616   \directlua0{%
617     oberdiek.pdfdoccmds.mdfivesum("\luaescapestring{#1}", "byte")%
618   }%
619 }%

```

`\pdf@mdfivesumnative`

```
620 \long\def\pdf@mdfivesumnative#1{%
621   \directlua0{%
622     oberdiek.pdfTEXcmds.mdfivesum("\luaescapestring{#1}")%
623   }%
624 }%
```

`\pdf@filemdfivesum`

```
625 \def\pdf@filemdfivesum#1{%
626   \directlua0{%
627     oberdiek.pdfTEXcmds.filemdfivesum("\luaescapestring{#1}")%
628   }%
629 }%
```

2.10.4 Timekeeping [pdfTeX-manual]

`\protected`

```
630 \let\pdfTEXcmds@temp=Y%
631 \begingroup\expandafter\expandafter\expandafter\endgroup
632 \expandafter\ifx\csname protected\endcsname\relax
633   \pdfTEXcmds@directlua0{%
634     if tex.enableprimitives then %
635       tex.enableprimitives(',{protected}')%
636     end%
637   }%
638 \fi
639 \begingroup\expandafter\expandafter\expandafter\endgroup
640 \expandafter\ifx\csname protected\endcsname\relax
641   \let\pdfTEXcmds@temp=N%
642 \fi
```

`\numexpr`

```
643 \begingroup\expandafter\expandafter\expandafter\endgroup
644 \expandafter\ifx\csname numexpr\endcsname\relax
645   \pdfTEXcmds@directlua0{%
646     if tex.enableprimitives then %
647       tex.enableprimitives(',{numexpr}')%
648     end%
649   }%
650 \fi
651 \begingroup\expandafter\expandafter\expandafter\endgroup
652 \expandafter\ifx\csname numexpr\endcsname\relax
653   \let\pdfTEXcmds@temp=N%
654 \fi
655 \ifx\pdfTEXcmds@temp N%
656   \@PackageWarningNoLine{pdfTEXcmds}{%
657     Definitions of \ltx@backslashchar pdf@resettimer and%
658     \MessageBreak
659     \ltx@backslashchar pdf@elapsedtime are skipped, because%
660     \MessageBreak
661     e-TeX's \ltx@backslashchar protected or %
662     \ltx@backslashchar numexpr are missing%
663   }%
664 \else
```

`\pdf@resettimer`

```
665 \protected\def\pdf@resettimer{%
666   \pdfTEXcmds@directlua0{%
667     oberdiek.pdfTEXcmds.resettimer()%
668   }%
669 }%
```

`\pdf@elapsedtime`

```
670 \protected\def\pdf@elapsedtime{%
671   \numexpr
672   \pdfcmds@directlua{%
673     oberdiek.pdfcmds.elapsedtime()}%
674   }%
675   \relax
676 }%

677 \fi
```

2.10.5 Shell escape

`\pdf@shellescape`

```
678 \ifnum\luatexversion<68 %
679 \else
680 \protected\edef\pdf@shellescape{%
681   \numexpr\directlua{tex.sprint(%
682     \number\catcodetable@string,status.shell_escape)}\relax}
683 \fi
```

`\pdf@system`

```
684 \def\pdf@system#1{%
685   \directlua{%
686     oberdiek.pdfcmds.system("\luaescapestring{#1}")%
687   }%
688 }
```

`\pdf@lastsystemstatus`

```
689 \def\pdf@lastsystemstatus{%
690   \directlua{%
691     oberdiek.pdfcmds.lastsystemstatus()}%
692   }%
693 }
```

`\pdf@lastsystemexit`

```
694 \def\pdf@lastsystemexit{%
695   \directlua{%
696     oberdiek.pdfcmds.lastsystemexit()}%
697   }%
698 }
```

```
699 \catcode`\0=12 %
```

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

```
700 \ifnum0%
701   \pdfcmds@directlua{%
702     if io.popen then %
703       tex.write("1")%
704     end%
705   }%
706   =1 %
707   \def\pdf@pipe#1{%
708 & \romannumeral\expandafter\pdfcmds@PatchDecode
709   \the\expandafter\pdfcmds@toks
710   \pdfcmds@directlua{%
711     oberdiek.pdfcmds.toks="pdfcmds@toks"%
712     oberdiek.pdfcmds.pipe("\luaescapestring{#1}", \pdfcmds@Patch)%
713   }%
714 & \@nil
715   }%
716 \fi
```

```
717 \pdfdoccmds@AtEnd%
718 </package>
```

2.11 Lua module

```
719 <*lua>
720 module("oberdiek.pdfdoccmds", package.seeall)
721 local systemexitstatus
722 function getversion()
723   tex.write("2018/09/10 v0.29")
724 end
```

2.11.1 Strings [pdfdoc-manual]

```
725 function strcmp(A, B)
726   if A == B then
727     tex.write("0")
728   elseif A < B then
729     tex.write("-1")
730   else
731     tex.write("1")
732   end
733 end
734 local function utf8_to_byte(str)
735   local i = 0
736   local n = string.len(str)
737   local t = {}
738   while i < n do
739     i = i + 1
740     local a = string.byte(str, i)
741     if a < 128 then
742       table.insert(t, string.char(a))
743     else
744       if a >= 192 and i < n then
745         i = i + 1
746         local b = string.byte(str, i)
747         if b < 128 or b >= 192 then
748           i = i - 1
749         elseif a == 194 then
750           table.insert(t, string.char(b))
751         elseif a == 195 then
752           table.insert(t, string.char(b + 64))
753         end
754       end
755     end
756   end
757   return table.concat(t)
758 end
759 function escapehex(str, mode)
760   if mode == "byte" then
761     str = utf8_to_byte(str)
762   end
763   tex.write((string.gsub(str, ".",
764     function (ch)
765       return string.format("%02X", string.byte(ch))
766     end
767   )))
768 end
```

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

```
769 function unescapehex(str, mode, patch)
770   local a = 0
```

```

771 local first = true
772 local result = {}
773 for i = 1, string.len(str), 1 do
774   local ch = string.byte(str, i)
775   if ch >= 48 and ch <= 57 then
776     ch = ch - 48
777   elseif ch >= 65 and ch <= 70 then
778     ch = ch - 55
779   elseif ch >= 97 and ch <= 102 then
780     ch = ch - 87
781   else
782     ch = nil
783   end
784   if ch then
785     if first then
786       a = ch * 16
787       first = false
788     else
789       table.insert(result, a + ch)
790       first = true
791     end
792   end
793 end
794 if not first then
795   table.insert(result, a)
796 end
797 if patch == 1 then
798   local temp = {}
799   for i, a in ipairs(result) do
800     if a == 0 then
801       table.insert(temp, 1)
802       table.insert(temp, 1)
803     else
804       if a == 1 then
805         table.insert(temp, 1)
806         table.insert(temp, 2)
807       else
808         table.insert(temp, a)
809       end
810     end
811   end
812   result = temp
813 end
814 if mode == "byte" then
815   local utf8 = {}
816   for i, a in ipairs(result) do
817     if a < 128 then
818       table.insert(utf8, a)
819     else
820       if a < 192 then
821         table.insert(utf8, 194)
822         a = a - 128
823       else
824         table.insert(utf8, 195)
825         a = a - 192
826       end
827       table.insert(utf8, a + 128)
828     end
829   end
830   result = utf8
831 end

```

this next line added for current luatex; this is the only change in the file. eroux,

28apr13. (v 0.21)

```
832 local unpack = _G["unpack"] or table.unpack
833 tex.settoks(toks, string.char(unpack(result)))
834 end
```

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

```
835 function escapestring(str, mode)
836   if mode == "byte" then
837     str = utf8_to_byte(str)
838   end
839   tex.write((string.gsub(str, ".",
840     function (ch)
841       local b = string.byte(ch)
842       if b < 33 or b > 126 then
843         return string.format("\\%.3o", b)
844       end
845       if b == 40 or b == 41 or b == 92 then
846         return "\\\" .. ch
847       end
848     end)))
849 end
```

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

```
848   return nil
849 end
850 )))
851 end
```

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

```
852 function escapename(str, mode)
853   if mode == "byte" then
854     str = utf8_to_byte(str)
855   end
856   tex.write((string.gsub(str, ".",
857     function (ch)
858       local b = string.byte(ch)
859       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.

```
860     return ""
861   end
862   if b <= 32 or b >= 127
863     or b == 35 or b == 37 or b == 40 or b == 41
864     or b == 47 or b == 60 or b == 62 or b == 91
865     or b == 93 or b == 123 or b == 125 then
866     return string.format("#%.2X", b)
867   else
```

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

```
868   return nil
869 end
870 end
871 )))
872 end
```

2.11.2 Files [pdftex-manual]

```
873 function filesize(filename)
874   local foundfile = kpse.find_file(filename, "tex", true)
875   if foundfile then
876     local size = lfs.attributes(foundfile, "size")
877     if size then
878       tex.write(size)
879     end
880   end
881 end
```

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

```

882 function filemoddate(filename)
883   local foundfile = kpse.find_file(filename, "tex", true)
884   if foundfile then
885     local date = lfs.attributes(foundfile, "modification")
886     if date then
887       local d = os.date("!*t", date)
888       if d.sec >= 60 then
889         d.sec = 59
890       end
891       local u = os.date("!*t", date)
892       local off = 60 * (d.hour - u.hour) + d.min - u.min
893       if d.year ~= u.year then
894         if d.year > u.year then
895           off = off + 1440
896         else
897           off = off - 1440
898         end
899       elseif d.yday ~= u.yday then
900         if d.yday > u.yday then
901           off = off + 1440
902         else
903           off = off - 1440
904         end
905       end
906       local timezone
907       if off == 0 then
908         timezone = "Z"
909       else
910         local hours = math.floor(off / 60)
911         local mins = math.abs(off - hours * 60)
912         timezone = string.format("%+03d%02d'", hours, mins)
913       end
914       tex.write(string.format("D:%04d%02d%02d%02d%02d%s",
915         d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
916     end
917   end
918 end
919 function filedump(offset, length, filename)
920   length = tonumber(length)
921   if length and length > 0 then
922     local foundfile = kpse.find_file(filename, "tex", true)
923     if foundfile then
924       offset = tonumber(offset)
925       if not offset then
926         offset = 0
927       end
928       local filehandle = io.open(foundfile, "rb")
929       if filehandle then
930         if offset > 0 then
931           filehandle:seek("set", offset)
932         end
933         local dump = filehandle:read(length)
934         escapehex(dump)
935         filehandle:close()
936       end
937     end
938   end
939 end
940 function mdffivesum(str, mode)
941   if mode == "byte" then
942     str = utf8_to_byte(str)
943   end

```



```

944 escapehex(md5.sum(str))
945 end
946 function filemdfivesum(filename)
947 local foundfile = kpse.find_file(filename, "tex", true)
948 if foundfile then
949     local filehandle = io.open(foundfile, "rb")
950     if filehandle then
951         local contents = filehandle:read("*a")
952         escapehex(md5.sum(contents))
953         filehandle:close()
954     end
955 end
956 end

```

2.11.3 Timekeeping [pdftex-manual]

The functions for timekeeping are based on Andy Thomas' work [AndyThomas:Analog]. Changes:

- Overflow check is added.
- `string.format` is used to avoid exponential number representation for sure.
- `tex.write` is used instead of `tex.print` to get tokens with catcode 12 and without appended `\endlinechar`.

```

957 local basetime = 0
958 function resettimer()
959     basetime = os.clock()
960 end
961 function elapsedtime()
962     local val = (os.clock() - basetime) * 65536 + .5
963     if val > 2147483647 then
964         val = 2147483647
965     end
966     tex.write(string.format("%d", val))
967 end

```

2.11.4 Miscellaneous [pdftex-manual]

```

968 function shellescape()
969     if os.execute then
970         if status
971             and status.luatex_version
972             and status.luatex_version >= 68 then
973             tex.write(os.execute())
974         else
975             local result = os.execute()
976             if result == 0 then
977                 tex.write("0")
978             else
979                 if result == nil then
980                     tex.write("0")
981                 else
982                     tex.write("1")
983                 end
984             end
985         end
986     else
987         tex.write("0")
988     end
989 end
990 function system(cmdline)
991     systemexitstatus = nil
992     texio.write_nl("log", "system(" .. cmdline .. ") ")

```

```

993 if os.execute then
994   texio.write("log", "executed.")
995   systemexitstatus = os.execute(cmdline)
996 else
997   texio.write("log", "disabled.")
998 end
999 end
1000 function lastsystmstatus()
1001   local result = tonumber(systemexitstatus)
1002   if result then
1003     local x = math.floor(result / 256)
1004     tex.write(result - 256 * math.floor(result / 256))
1005   end
1006 end
1007 function lastsystmexit()
1008   local result = tonumber(systemexitstatus)
1009   if result then
1010     tex.write(math.floor(result / 256))
1011   end
1012 end
1013 function pipe(cmdline, patch)
1014   local result
1015   systemexitstatus = nil
1016   texio.write_nl("log", "pipe(" .. cmdline .. ") ")
1017   if io.popen then
1018     texio.write("log", "executed.")
1019     local handle = io.popen(cmdline, "r")
1020     if handle then
1021       result = handle:read("*a")
1022       handle:close()
1023     end
1024   else
1025     texio.write("log", "disabled.")
1026   end
1027   if result then
1028     if patch == 1 then
1029       local temp = {}
1030       for i, a in ipairs(result) do
1031         if a == 0 then
1032           table.insert(temp, 1)
1033           table.insert(temp, 1)
1034         else
1035           if a == 1 then
1036             table.insert(temp, 1)
1037             table.insert(temp, 2)
1038           else
1039             table.insert(temp, a)
1040           end
1041         end
1042       end
1043       result = temp
1044     end
1045     tex.settoks(toks, result)
1046   else
1047     tex.settoks(toks, "")
1048   end
1049 end
1050  $\langle$ /lua $\rangle$ 

```

3 Test

3.1 Catcode checks for loading

```
1051 ⟨*test1⟩
1052 \catcode`\{=1 %
1053 \catcode`\}=2 %
1054 \catcode`\#=6 %
1055 \catcode`\@=11 %
1056 \expandafter\ifx\csname count@\endcsname\relax
1057 \countdef\count@=255 %
1058 \fi
1059 \expandafter\ifx\csname @gobble\endcsname\relax
1060 \long\def\@gobble#1{}%
1061 \fi
1062 \expandafter\ifx\csname @firstofone\endcsname\relax
1063 \long\def\@firstofone#1{#1}%
1064 \fi
1065 \expandafter\ifx\csname loop\endcsname\relax
1066 \expandafter\@firstofone
1067 \else
1068 \expandafter\@gobble
1069 \fi
1070 {%
1071 \def\loop#1\repeat{%
1072 \def\body{#1}%
1073 \iterate
1074 }%
1075 \def\iterate{%
1076 \body
1077 \let\next\iterate
1078 \else
1079 \let\next\relax
1080 \fi
1081 \next
1082 }%
1083 \let\repeat=\fi
1084 }%
1085 \def\RestoreCatcodes{}
1086 \count@=0 %
1087 \loop
1088 \edef\RestoreCatcodes{%
1089 \RestoreCatcodes
1090 \catcode\the\count@=\the\catcode\count@\relax
1091 }%
1092 \ifnum\count@<255 %
1093 \advance\count@ 1 %
1094 \repeat
1095
1096 \def\RangeCatcodeInvalid#1#2{%
1097 \count@=#1\relax
1098 \loop
1099 \catcode\count@=15 %
1100 \ifnum\count@<#2\relax
1101 \advance\count@ 1 %
1102 \repeat
1103 }
1104 \def\RangeCatcodeCheck#1#2#3{%
1105 \count@=#1\relax
1106 \loop
1107 \ifnum#3=\catcode\count@
1108 \else
```

```

1109   \errmessage{%
1110     Character \the\count@\space
1111     with wrong catcode \the\catcode\count@\space
1112     instead of \number#3%
1113   }%
1114   \fi
1115   \ifnum\count@<#2\relax
1116     \advance\count@ 1 %
1117   \repeat
1118 }
1119 \def\space{ }
1120 \expandafter\ifx\csname LoadCommand\endcsname\relax
1121 \def\LoadCommand{\input pdftexcmds.sty\relax}%
1122 \fi
1123 \def\Test{%
1124   \RangeCatcodeInvalid{0}{47}%
1125   \RangeCatcodeInvalid{58}{64}%
1126   \RangeCatcodeInvalid{91}{96}%
1127   \RangeCatcodeInvalid{123}{255}%
1128   \catcode`\@=12 %
1129   \catcode`\|=0 %
1130   \catcode`\%=14 %
1131   \LoadCommand
1132   \RangeCatcodeCheck{0}{36}{15}%
1133   \RangeCatcodeCheck{37}{37}{14}%
1134   \RangeCatcodeCheck{38}{47}{15}%
1135   \RangeCatcodeCheck{48}{57}{12}%
1136   \RangeCatcodeCheck{58}{63}{15}%
1137   \RangeCatcodeCheck{64}{64}{12}%
1138   \RangeCatcodeCheck{65}{90}{11}%
1139   \RangeCatcodeCheck{91}{91}{15}%
1140   \RangeCatcodeCheck{92}{92}{0}%
1141   \RangeCatcodeCheck{93}{96}{15}%
1142   \RangeCatcodeCheck{97}{122}{11}%
1143   \RangeCatcodeCheck{123}{255}{15}%
1144   \RestoreCatcodes
1145 }
1146 \Test
1147 \csname @@end\endcsname
1148 \end
1149 </test1>

```

3.2 Test for \pdf@isprimitive

```

1150 <*test2>
1151 \catcode`\{=1 %
1152 \catcode`\}=2 %
1153 \catcode`\#=6 %
1154 \catcode`\@=11 %
1155 \input pdftexcmds.sty\relax
1156 \def\msg#1{%
1157   \begingroup
1158     \escapechar=92 %
1159     \immediate\write16{#1}%
1160   \endgroup
1161 }
1162 \long\def\test#1#2#3#4{%
1163   \begingroup
1164     #4%
1165   \def\str{%
1166     Test \string\pdf@isprimitive
1167     {\string #1}{\string #2}{...}: %
1168   }%

```

```

1169 \pdf@isprimitive{#1}{#2}{%
1170 \ifx#3Y%
1171 \msg{\str true ==> OK.}%
1172 \else
1173 \errmessage{\str false ==> FAILED}%
1174 \fi
1175 }{%
1176 \ifx#3Y%
1177 \errmessage{\str true ==> FAILED}%
1178 \else
1179 \msg{\str false ==> OK.}%
1180 \fi
1181 }%
1182 \endgroup
1183 }
1184 \test\relax\relax Y{}
1185 \test\foobar\relax Y{\let\foobar\relax}
1186 \test\foobar\relax N{}
1187 \test\hbox\hbox Y{}
1188 \test\foobar@hbox\hbox Y{\let\foobar@hbox\hbox}
1189 \test\if\if Y{}
1190 \test\if\ifx N{}
1191 \test\ifx\if N{}
1192 \test\par\par Y{}
1193 \test\hbox\par N{}
1194 \test\par\hbox N{}
1195 \csname @@end\endcsname\end
1196 </test2>

```

3.3 Test for \pdf@shellescape

```

1197 <*test-shell>
1198 \catcode`\{=1 %
1199 \catcode`\}=2 %
1200 \catcode`\#=6 %
1201 \catcode`\@=11 %
1202 \input pdftexcmds.sty\relax
1203 \def\msg#{\immediate\write16}
1204 \def\MaybeEnd{}
1205 \ifx\luatexversion\UnDeFiNeD
1206 \else
1207 \ifnum\luatexversion<68 %
1208 \ifx\pdf@shellescape\@undefined
1209 \msg{SHELL=U}%
1210 \msg{OK (LuaTeX < 0.68)}%
1211 \else
1212 \msg{SHELL=defined}%
1213 \errmessage{Failed (LuaTeX < 0.68)}%
1214 \fi
1215 \def\MaybeEnd{\csname @@end\endcsname\end}%
1216 \fi
1217 \fi
1218 \MaybeEnd
1219 \ifx\pdf@shellescape\@undefined
1220 \msg{SHELL=U}%
1221 \else
1222 \msg{SHELL=\number\pdf@shellescape}%
1223 \fi
1224 \ifx\expected\@undefined
1225 \else
1226 \ifx\expected\relax
1227 \msg{EXPECTED=U}%
1228 \ifx\pdf@shellescape\@undefined

```

```

1229   \msg{OK}%
1230   \else
1231   \errmessage{Failed}%
1232   \fi
1233 \else
1234   \msg{EXPECTED=\number\expected}%
1235   \ifnum\pdf@shellescape=\expected\relax
1236   \msg{OK}%
1237   \else
1238   \errmessage{Failed}%
1239   \fi
1240 \fi
1241 \fi
1242 \csname @@end\endcsname\end
1243 </test-shell)

```

3.4 Test for escape functions

```

1244 <*test-escape)
1245 \catcode`\{=1 %
1246 \catcode`\}=2 %
1247 \catcode`\#=6 %
1248 \catcode`\^=7 %
1249 \catcode`\@=11 %
1250 \errorcontextlines=1000 %
1251 \input pdftexcmds.sty\relax
1252 \def\msg#1{%
1253   \begingroup
1254   \escapechar=92 %
1255   \immediate\write16{#1}%
1256   \endgroup
1257 }

1258 \begingroup
1259 \catcode`\@=11 %
1260 \countdef\count@=255 %
1261 \def\space{ }%
1262 \long\def\@whilenum#1\do #2{%
1263   \ifnum #1\relax
1264     #2\relax
1265   \@iwhilenum{#1\relax#2\relax}%
1266   \fi
1267 }%
1268 \long\def\@iwhilenum#1{%
1269   \ifnum #1%
1270     \expandafter\@iwhilenum
1271   \else
1272     \expandafter\ltx@gobble
1273   \fi
1274   {#1}%
1275 }%
1276 \gdef\AllBytes{%
1277 \count@=0 %
1278 \catcode0=12 %
1279 \@whilenum\count@<256 \do{%
1280 \lccode0=\count@
1281 \ifnum\count@=32 %
1282 \xdef\AllBytes{\AllBytes\space}%
1283 \else
1284 \lowercase{%
1285 \xdef\AllBytes{\AllBytes^^@}%
1286 }%
1287 \fi
1288 \advance\count@ by 1 %

```

```

1289 }%
1290 \endgroup

1291 \def\AllBytesHex{%
1292 000102030405060708090A0B0C0D0E0F%
1293 101112131415161718191A1B1C1D1E1F%
1294 202122232425262728292A2B2C2D2E2F%
1295 303132333435363738393A3B3C3D3E3F%
1296 404142434445464748494A4B4C4D4E4F%
1297 505152535455565758595A5B5C5D5E5F%
1298 606162636465666768696A6B6C6D6E6F%
1299 707172737475767778797A7B7C7D7E7F%
1300 808182838485868788898A8B8C8D8E8F%
1301 909192939495969798999A9B9C9D9E9F%
1302 A0A1A2A3A4A5A6A7A8A9AAABACADAEAF%
1303 B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF%
1304 C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF%
1305 D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
1306 E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
1307 F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF%
1308 }
1309 \ltx@onelevel@sanitize\AllBytesHex
1310 \expandafter\lowercase\expandafter{%
1311 \expandafter\def\expandafter\AllBytesHexLC
1312 \expandafter{\AllBytesHex}%
1313 }
1314 \begingroup
1315 \catcode`\#=12 %
1316 \xdef\AllBytesName{%
1317 #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
1318 #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
1319 #20!"#23$#25&'#28#29*+,-.#2F%
1320 0123456789:;#3C=#3E?%
1321 @ABCDEFGHIJKLMNO%
1322 PQRSTUVWXYZ#5B\ltx@backslashchar#5D^_%
1323 `abcdefghijklmnopqrstuvwxyz
1324 pqrstuvwxyz#7B|#7D\string~#7F%
1325 #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
1326 #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
1327 #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
1328 #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
1329 #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
1330 #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
1331 #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
1332 #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
1333 }%
1334 \endgroup
1335 \ltx@onelevel@sanitize\AllBytesName
1336 \edef\AllBytesFromName{\expandafter\ltx@gobble\AllBytes}
1337 \begingroup
1338 \def\{|}%
1339 \edef%\{\ltx@percentchar}%
1340 \catcode`\|=0 %
1341 \catcode`\#=12 %
1342 \catcode`\~=12 %
1343 \catcode`\|=12 %
1344 |xdef|AllBytesString{%
1345 \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
1346 \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
1347 \040!"#$%&'(\)*+,-./%
1348 0123456789:;<=>?%
1349 @ABCDEFGHIJKLMNO%
1350 PQRSTUVWXYZ[\]^_%

```

```

1351 `abcdefghijklmnop%
1352 pqrstuvwxyz{|}~\177%
1353 \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
1354 \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
1355 \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
1356 \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
1357 \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
1358 \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
1359 \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
1360 \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
1361 }%
1362 |endgroup
1363 \ltx@onelevel@sanitize\AllBytesString
1364 \def\Test#1#2#3{%
1365 \begingroup
1366 \expandafter\expandafter\expandafter\def
1367 \expandafter\expandafter\expandafter\TestResult
1368 \expandafter\expandafter\expandafter{%
1369 #1{#2}%
1370 }%
1371 \ifx\TestResult#3%
1372 \else
1373 \newlinechar=10 %
1374 \msg{Expect:^^J#3}%
1375 \msg{Result:^^J\TestResult}%
1376 \errmessage{\string#2 -\string#1-> \string#3}%
1377 \fi
1378 \endgroup
1379 }
1380 \def\test#1#2#3{%
1381 \edef\TestFrom{#2}%
1382 \edef\TestExpect{#3}%
1383 \ltx@onelevel@sanitize\TestExpect
1384 \Test#1\TestFrom\TestExpect
1385 }
1386 \test\pdf@unescapehex{74657374}{test}
1387 \begingroup
1388 \catcode0=12 %
1389 \catcode1=12 %
1390 \test\pdf@unescapehex{740074017400740174}{t^^@t^^At^^@t^^At}%
1391 \endgroup
1392 \Test\pdf@escapehex\AllBytes\AllBytesHex
1393 \Test\pdf@unescapehex\AllBytesHex\AllBytes
1394 \Test\pdf@escapename\AllBytes\AllBytesName
1395 \Test\pdf@escapestring\AllBytes\AllBytesString
1396 \csname @@end\endcsname\end
1397 </test-escape>

```

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.

¹<http://ctan.org/pkg/pdftexcmds>

[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):

<code>pdftexcmds.sty</code>	→ <code>tex/generic/oberdiek/pdftexcmds.sty</code>
<code>oberdiek.pdftexcmds.lua</code>	→ <code>scripts/oberdiek/oberdiek.pdftexcmds.lua</code>
<code>pdftexcmds.lua</code>	→ <code>scripts/oberdiek/pdftexcmds.lua</code>
<code>pdftexcmds.pdf</code>	→ <code>doc/latex/oberdiek/pdftexcmds.pdf</code>
<code>test/pdftexcmds-test1.tex</code>	→ <code>doc/latex/oberdiek/test/pdftexcmds-test1.tex</code>
<code>test/pdftexcmds-test2.tex</code>	→ <code>doc/latex/oberdiek/test/pdftexcmds-test2.tex</code>
<code>test/pdftexcmds-test-shell.tex</code>	→ <code>doc/latex/oberdiek/test/pdftexcmds-test-shell.tex</code>
<code>test/pdftexcmds-test-escape.tex</code>	→ <code>doc/latex/oberdiek/test/pdftexcmds-test-escape.tex</code>
<code>pdftexcmds.dtx</code>	→ <code>source/latex/oberdiek/pdftexcmds.dtx</code>

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 T_EX distribution (teT_EX, mikT_EX, ...) relies on file name databases, you must refresh these. For example, teT_EX users run `texhash` or `mktextlsr`.

4.5 Some details for the interested

Unpacking with L^AT_EX. The `.dtx` chooses its action depending on the format:

plain T_EX: Run `docstrip` and extract the files.

L^AT_EX: Generate the documentation.

If you insist on using L^AT_EX for `docstrip` (really, `docstrip` does not need L^AT_EX), 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 pdfL^AT_EX:

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

5 Catalogue

The following XML file can be used as source for the [T_EX Catalogue](#). The elements `caption` and `description` are imported from the original XML file from the Catalogue. The name of the XML file in the Catalogue is `pdftexcmds.xml`.

```
1398 (*catalogue)
1399 <?xml version='1.0' encoding='us-ascii'?>
1400 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
1401 <entry datestamp='$Date$' modifier='$Author$' id='pdftexcmds'>
1402   <name>pdftexcmds</name>
1403   <caption>LuaTeX support for pdfTeX utility functions.</caption>
1404   <authorref id='auth:oberdiek' />
1405   <copyright owner='Heiko Oberdiek' year='2007,2009-2011' />
1406   <license type='lppl1.3' />
1407   <version number='0.20' />
1408   <description>
1409     LuaTeX provides most of the commands of
1410     <xref refid='pdftex'>pdfTeX</xref> 1.40. However, a number of
1411     utility functions are not available. This package tries to fill
1412     the gap and implements some of the missing primitives using Lua.
1413     <p />
1414     The package is part of the <xref refid='oberdiek'>oberdiek</xref>
1415     bundle.
1416   </description>
1417   <documentation details='Package documentation'
1418     href='ctan:/macros/latex/contrib/oberdiek/pdftexcmds.pdf' />
1419   <ctan file='true' path='/macros/latex/contrib/oberdiek/pdftexcmds.dtx' />
1420   <miktex location='oberdiek' />
1421   <texlive location='oberdiek' />
1422   <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip' />
1423 </entry>
1424 </catalogue>
```

6 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 LuaTeX 0.36.

[2009/09/22 v0.5]

- `\pdf@primitive`, `\pdf@ifprimitive` added.
- X_YTeX's variants are detected for `\pdf@shellescape`, `\pdf@stricmp`, `\pdf@primitive`, `\pdf@ifprimitive`.

[2009/09/23 v0.6]

- Macro `\pdf@isprimitive` added.

[2009/12/12 v0.7]

- Short info shortened.

[2010/03/01 v0.8]

- Required date for package `ifluatex` updated.

[2010/04/01 v0.9]

- Use `\ifeof18` for defining `\pdf@shellescape` between pdfTeX 1.21a (inclusive) and 1.30.0 (exclusive).

[2010/11/04 v0.10]

- `\pdf@draftmode`, `\pdf@ifdraftmode` and `\pdf@setdraftmode` added.

[2010/11/11 v0.11]

- Missing `\RequirePackage` for package `ifpdf` added.

[2011/01/30 v0.12]

- Already loaded package files are not input in plain TeX.

[2011/03/04 v0.13]

- Improved Lua function `shellescape` that also uses the result of `os.execute()` (thanks to Philipp Stephani).

[2011/04/10 v0.14]

- Version check of loaded module added.
- Patch for bug in LuaTeX between 0.40.6 and 0.65 that is fixed in revision 4096.

[2011/04/16 v0.15]

- LuaTeX: `\pdf@shellescape` is only supported for version 0.70.0 and higher due to a bug, `os.execute()` crashes in some circumstances. Fixed in LuaTeX beta-0.70.0, revision 4167.

[2011/04/22 v0.16]

- Previous fix was not working due to a wrong catcode of digit zero (due to easily support the old `\directlua0`). The version border is lowered to 0.68, because some beta-0.67.0 seems also to work.

[2011/06/29 v0.17]

- Documentation addition to `\pdf@shellescape`.

[2011/07/01 v0.18]

- Add Lua module loading in `\everyjob` for `iniTeX` (LuaTeX only).

[2011/07/28 v0.19]

- Missing space in an info message added (Martin Münch).

[2011/11/29 v0.20]

- `\pdf@resettimer` and `\pdf@elapsedtime` added (thanks Andy Thomas).

[2016/05/10 v0.21]

- local `unpack` added (thanks Élie Roux).

[2016/05/21 v0.22]

- adjust `\textbackslash` usage in bib file for biber bug.

[2016/10/02 v0.23]

- add `file.close` to lua filehandles (github pull request).

[2017/01/29 v0.24]

- Avoid loading `luatex-loader` for current `luatex`. (Use `pdftexcmds.lua` not `oberdiek.pdftexcmds.lua` to simplify file search with standard require)

[2017/03/19 v0.25]

- New `\pdf@shellescape` for LuaTeX, see github issue 20.

[2018/01/21 v0.26]

- use `rb` not `r` mode for file open github issue 34.

[2018/01/30 v0.27]

- `\pdf@mdfivesum` for `X3TeX`

[2018/09/07 v0.28]

- Fix catcode regime in `luatex sprint` for `\pdf@shellescape` GH issue 45

[2018/09/10 v0.29]

- Actually do the fix described above in the code, not just document it.

7 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; plain numbers refer to the code lines where the entry is used.

Symbols	
<code>\#</code>	1054, 1153, 1200, 1247, 1315, 1341
<code>\%</code>	1130, 1339
<code>\&</code>	514, 516
<code>\(</code>	1347
<code>\)</code>	1347
<code>\@</code>	1055, 1128, 1154, 1201, 1249, 1259
<code>\@PackageError</code>	494
<code>\@PackageInfoNoLine</code>	152, 154, 289, 303, 308, 391, 395, 397
<code>\@PackageWarning</code>	440
<code>\@PackageWarningNoLine</code>	656
<code>\@ehc</code>	498
<code>\@firstofone</code>	1063, 1066
<code>\@gobble</code>	1060, 1068
<code>\@iwhilenum</code>	1265, 1268, 1270
<code>\@nil</code>	517, 518, 520, 523, 527, 531, 539, 572, 581, 714
<code>\@undefined</code>	58, 288, 383, 1208, 1219, 1224, 1228
<code>\@whilenum</code>	1262, 1279
<code>\\</code>	843, 846, 1129, 1343, 1350
<code>\{</code>	1052, 1151, 1198, 1245
<code>\}</code>	1053, 1152, 1199, 1246
<code>\^</code>	1248
<code>\ </code>	1338, 1340
<code>\~</code>	1342
Numbers	
<code>\0</code>	546, 699, 1345, 1346, 1347
<code>\1</code>	1352
<code>\2</code>	1353, 1354, 1355, 1356
<code>\3</code>	1357, 1358, 1359, 1360
A	
<code>\advance</code>	1093, 1101, 1116, 1288
<code>\aftergroup</code>	29
<code>\AllBytes</code>	1276, 1282, 1285, 1336, 1392, 1393, 1394, 1395
<code>\AllBytesFromName</code>	1336
<code>\AllBytesHex</code> 1291, 1309, 1312, 1392, 1393	
<code>\AllBytesHexLC</code>	1311
<code>\AllBytesName</code>	1316, 1335, 1394
<code>\AllBytesString</code>	1363, 1395
B	
<code>\body</code>	1072, 1076
C	
<code>\catcode</code>	2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 69, 70, 72, 73, 74, 78, 79, 80, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 97, 99, 514, 516, 546, 699, 1052, 1053, 1054, 1055, 1090, 1099, 1107, 1111, 1128, 1129, 1130, 1151, 1152, 1153, 1154, 1198, 1199, 1200, 1201, 1245, 1246, 1247, 1248, 1249, 1259, 1278, 1315, 1340, 1341, 1342, 1343, 1388, 1389
<code>\catcodetable@string</code>	682
<code>\chardef</code>	184, 186
<code>\count</code>	428, 430
<code>\count@</code> 1057, 1086, 1090, 1092, 1093, 1097, 1099, 1100, 1101, 1105, 1107, 1110, 1111, 1115, 1116, 1260, 1277, 1279, 1280, 1281, 1288	
<code>\countdef</code>	1057, 1260
<code>\csname</code>	14, 21, 50, 66, 76, 133, 136, 159, 162, 164, 178, 196, 204, 206, 223, 226, 227, 256, 261, 264, 265, 279, 281, 284, 305, 310, 316, 319, 325, 327, 336, 371, 452, 455, 502, 505, 632, 640, 644, 652, 1056, 1059, 1062, 1065, 1120, 1147, 1195, 1215, 1242, 1396
D	
<code>\delimiter</code>	344, 350, 368
<code>\directlua</code>	235, 237, 549, 556, 561, 568, 577, 584, 589, 594, 599, 604, 609, 616, 621, 626, 681, 685, 690, 695
<code>\do</code>	1262, 1279
<code>\draftmode</code>	384
E	
<code>\empty</code>	17, 18
<code>\end</code>	1148, 1195, 1215, 1242, 1396
<code>\endcsname</code>	14, 21, 50, 66, 76, 133, 136, 159, 162, 164, 178, 196, 204, 206, 223, 226, 227, 256, 261, 264, 265, 279, 281, 284, 305, 310, 316, 319, 325, 327, 336, 371, 452, 455, 502, 505, 632, 640, 644, 652, 1056, 1059, 1062, 1065, 1120, 1147, 1195, 1215, 1242, 1396
<code>\endinput</code>	29, 129
<code>\endlinechar</code>	4, 35, 71, 77, 89, 241
<code>\errmessage</code>	1109, 1173, 1177, 1213, 1231, 1238, 1376
<code>\errorcontextlines</code>	1250
<code>\escapechar</code> 128, 131, 260, 318, 1158, 1254	
<code>\everyjob</code>	474, 475
<code>\expected</code>	1224, 1226, 1234, 1235
F	
<code>\foobar</code>	1185, 1186
<code>\foobar@hbox</code>	1188

	G	
<code>\gdef</code>		1276
	H	
<code>\hbox</code>		1187, 1188, 1193, 1194
	I	
<code>\if</code>		1189, 1190, 1191
<code>\ifcase</code>		402, 435, 513
<code>\ifeof</code>		182, 183
<code>\ifluatex</code>		150, 233, 277, 382, 446
<code>\ifnum</code>		182, 234, 283, 286, 344, 374, 409, 416, 450, 467, 468, 508, 509, 544, 678, 700, 1092, 1100, 1107, 1115, 1207, 1235, 1263, 1269, 1281
<code>\ifpdf</code>		393
<code>\ifx</code>		15, 18, 21, 50, 58, 61, 133, 136, 159, 178, 196, 204, 206, 223, 256, 263, 279, 281, 284, 305, 310, 316, 324, 336, 351, 352, 355, 357, 383, 452, 455, 492, 502, 521, 532, 632, 640, 644, 652, 655, 1056, 1059, 1062, 1065, 1120, 1170, 1176, 1190, 1191, 1205, 1208, 1219, 1224, 1226, 1228, 1371
<code>\immediate</code>		23, 52, 219, 1159, 1203, 1255
<code>\input</code>		137, 456, 1121, 1155, 1202, 1251
<code>\iterate</code>		1073, 1075, 1077
	L	
<code>\lccode</code>		1280
<code>\LoadCommand</code>		1121, 1131
<code>\loop</code>		1071, 1087, 1098, 1106
<code>\lowercase</code>		1284, 1310
<code>\ltx@backslashchar</code>		391, 395, 398, 657, 659, 661, 662, 1322
<code>\ltx@cclv</code>		428, 430
<code>\ltx@firstoftwo</code>		345, 375, 410
<code>\ltx@gobble</code>		1272, 1336
<code>\ltx@ifUndefined</code>		180, 390
<code>\ltx@one</code>		394, 409, 416, 417
<code>\ltx@onelevel@sanitize</code>		484, 1309, 1335, 1363, 1383
<code>\ltx@percentchar</code>		1339
<code>\ltx@ReturnAfterElseFi</code>		522, 533
<code>\ltx@ReturnAfterFi</code>		526, 538
<code>\ltx@secondoftwo</code>		347, 377, 404, 412
<code>\ltx@zero</code>		389, 403, 419, 534
<code>\luaescapestring</code>		550, 551, 557, 562, 570, 579, 585, 590, 595, 600, 605, 610, 611, 612, 617, 622, 627, 686, 712
<code>\luatexversion</code>		234, 450, 467, 508, 509, 544, 678, 1205, 1207
	M	
<code>\MaybeEnd</code>		1204, 1215, 1218
<code>\mdfivesum</code>		209, 211
<code>\meaning</code> ..		259, 261, 319, 322, 338, 374
<code>\MessageBreak</code>		291, 441, 495, 496, 658, 660
<code>\msg</code> ..		1156, 1171, 1179, 1203, 1209, 1210, 1212, 1220, 1222, 1227, 1229, 1234, 1236, 1252, 1374, 1375
	N	
<code>\newlinechar</code>		240, 241, 1373
<code>\next</code>		1077, 1079, 1081
<code>\number</code>		128, 610, 611, 682, 1112, 1222, 1234
<code>\numexpr</code>		643, 671, 681
	P	
<code>\PackageInfo</code>		26
<code>\par</code>		1192, 1193, 1194
<code>\pdf@draftmode</code>		5, 403, 415
<code>\pdf@elapsedtime</code>		4, 670
<code>\pdf@escapehex</code>		4, 170, 555, 1392
<code>\pdf@escapehexnative</code>		170, 560
<code>\pdf@escapename</code>		588, 1394
<code>\pdf@escapenamename</code>		593
<code>\pdf@escapestring</code>		583, 1395
<code>\pdf@filedump</code>		4, 199, 608
<code>\pdf@filemdfivesum</code>		4, 211, 216, 625
<code>\pdf@filemoddate</code>		4, 603
<code>\pdf@filesize</code>		4, 598
<code>\pdf@ifdraftmode</code>		5, 404, 408
<code>\pdf@ifprimitive</code>		6, 273, 309
<code>\pdf@isprimitive</code>		6, 334, 337, 373, 387, 554, 1166, 1169
<code>\pdf@lastsystemexit</code>		694
<code>\pdf@lastsystemstatus</code>		689
<code>\pdf@mdfivesum</code> ..		4, 209, 210, 214, 215, 615
<code>\pdf@mdfivesumnative</code> ..		210, 215, 620
<code>\pdf@pipe</code>		6, 700
<code>\pdf@primitive</code> ..		6, 269, 288, 290, 304
<code>\pdf@resettimer</code>		4, 665
<code>\pdf@setdraftmode</code>		5, 426, 441
<code>\pdf@shellescape</code> ..		5, 184, 186, 191, 678, 1208, 1219, 1222, 1228, 1235
<code>\pdf@strcmp</code>		3, 374, 548
<code>\pdf@system</code>		6, 218, 684
<code>\pdf@unescapehex</code>		4, 172, 565, 1386, 1390, 1393
<code>\pdf@unescapehexnative</code> ..		6, 172, 574
<code>\pdf@draftmode</code>		383, 384, 407
<code>\pdf@filedump</code>		200
<code>\pdf@mdfivesum</code>		214, 216
<code>\pdf@primitive</code>		292
<code>\pdf@shellescape</code>		192
<code>\pdf@texcmds@isprimitive</code> ..		341, 343
<code>\pdf@texcmds@setdraftmode</code> ..		430, 434
<code>\pdf@texcmds@AtEnd</code>		95, 96, 126, 127, 448, 717
<code>\pdf@texcmds@DecodeA</code>		518, 520
<code>\pdf@texcmds@DecodeB</code>		523, 531
<code>\pdf@texcmds@directlua</code>		234, 242, 464, 469, 476, 486, 633, 645, 666, 672, 701, 710
<code>\pdf@texcmds@draftmode</code>		407, 409, 416, 423
<code>\pdf@texcmds@equal</code>		344, 350, 368
<code>\pdf@texcmds@equalcont</code>		359, 365, 366, 371
<code>\pdf@texcmds@isprimitive</code> ..		338, 340

- \pdfTeXcmds@nopdfTeX 153, 155, 160, 179, 197, 207, 224
- \pdfTeXcmds@Patch 507, 513, 570, 579, 712
- \pdfTeXcmds@PatchDecode 517, 566, 575, 708
- \pdfTeXcmds@setdraftmode 405, 422, 436, 438
- \pdfTeXcmds@strip@prefix 253, 259
- \pdfTeXcmds@temp 157, 168, 169, 171, 173, 174, 175, 176, 221, 230, 231, 254, 269, 270, 271, 272, 273, 274, 275, 276, 314, 332, 333, 389, 394, 402, 630, 641, 653, 655
- \pdfTeXcmds@toks 501, 567, 576, 709
- \pdfTeXrevision 286
- \pdfTeXversion 182, 283
- \protected 630, 665, 670, 680
- \ProvidesPackage 19, 67

R

- \RangeCatcodeCheck 1104, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143
- \RangeCatcodeInvalid 1096, 1124, 1125, 1126, 1127
- \repeat 1071, 1083, 1094, 1102, 1117
- \RequirePackage 145, 146, 147, 148, 461
- \RestoreCatcodes 1085, 1088, 1089, 1144
- \romannumeral 566, 575, 708

S

- \space 290, 292, 304, 309, 1110, 1111, 1119, 1261, 1282
- \str 1165, 1171, 1173, 1177, 1179

T

- \Test 1123, 1146, 1364, 1384, 1392, 1393, 1394, 1395
- \test 1162, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1380, 1386, 1390
- \TestExpect 1382, 1383, 1384
- \TestFrom 1381, 1384
- \TestResult 1367, 1371, 1375
- \the 77, 78, 79, 80, 81, 82, 83, 84, 97, 430, 475, 567, 576, 709, 1090, 1110, 1111
- \TMP@EnsureCode 94, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125
- \TMP@RequirePackage 134, 140, 141, 142, 143, 453, 459
- \toksdef 503

U

- \UnDeFiNeD 1205

W

- \write 23, 52, 219, 1159, 1203, 1255

X

- \x 14, 15, 18, 22, 26, 28, 51, 56, 66, 75, 87, 258, 259, 263, 319, 324, 429, 432, 483, 484, 492, 496

Y

- \y 261, 263, 320, 322, 324, 485, 492, 497

Z

- \z 321, 322