

The pdf_{tex}cmds package

Heiko Oberdiek*

2020-06-04 v0.32

Abstract

Lua_{TeX} provides most of the commands of pdf_{TeX} 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@ifprimitive</code>	6
1.2.6	Experimental	7
2	Implementation	7
2.1	Reload check and package identification	7
2.2	Catcodes	8
2.3	Load packages	9
2.4	Without Lua _{TeX}	10
2.5	<code>\pdf@primitive</code> , <code>\pdf@ifprimitive</code>	11
2.5.1	Using Lua _{TeX} 's <code>tex.enableprimitives</code>	11
2.5.2	Trying various names to find the primitives	12
2.5.3	Result	13
2.6	X _Y _{TeX}	13
2.7	<code>\pdf@ifprimitive</code>	13
2.8	<code>\pdf@draftmode</code>	14
2.9	Load Lua module	16
2.10	Lua functions	17
2.10.1	Helper macros	17
2.10.2	Strings	18
2.10.3	Files	19
2.10.4	Timekeeping	20
2.10.5	Shell escape	21
2.11	Lua module	22
2.11.1	Strings	22
2.11.2	Files	25
2.11.3	Timekeeping	26
2.11.4	Miscellaneous	27

*Please report any issues at <https://github.com/ho-tex/pdf_{tex}cmds/issues>

3	Installation	28
3.1	Download	28
3.2	Bundle installation	28
3.3	Package installation	29
3.4	Refresh file name databases	29
3.5	Some details for the interested	29
4	References	30
5	History	30
	[2007/11/11 v0.1]	30
	[2007/11/12 v0.2]	30
	[2007/12/12 v0.3]	30
	[2009/04/10 v0.4]	30
	[2009/09/22 v0.5]	30
	[2009/09/23 v0.6]	30
	[2009/12/12 v0.7]	30
	[2010/03/01 v0.8]	30
	[2010/04/01 v0.9]	30
	[2010/11/04 v0.10]	31
	[2010/11/11 v0.11]	31
	[2011/01/30 v0.12]	31
	[2011/03/04 v0.13]	31
	[2011/04/10 v0.14]	31
	[2011/04/16 v0.15]	31
	[2011/04/22 v0.16]	31
	[2011/06/29 v0.17]	31
	[2011/07/01 v0.18]	31
	[2011/07/28 v0.19]	31
	[2011/11/29 v0.20]	31
	[2016/05/10 v0.21]	31
	[2016/05/21 v0.22]	32
	[2016/10/02 v0.23]	32
	[2017/01/29 v0.24]	32
	[2017/03/19 v0.25]	32
	[2018/01/21 v0.26]	32
	[2018/01/30 v0.27]	32
	[2018/09/07 v0.28]	32
	[2018/09/10 v0.29]	32
	[2019/07/25 v0.30]	32
	[2019/11/24 v0.31]	32
	[2020-06-04 v0.32]	32
6	Index	32

1 Documentation

Some primitives of pdfTeX [1] are not defined by LuaTeX [2]. 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 be 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 register 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}
```

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@⟨cmd⟩` if pdfTeX provides `\pdf⟨cmd⟩`.

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

Expandability: The macro `\pdf@⟨cmd⟩` 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@⟨cmd⟩` 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 [1, “7.15 Strings”]

`\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 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 [1, “7.18 Files”]

`\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 [1, “7.17 Timekeeping”]

The timekeeping macros are based on Andy Thomas’ work [3].

`\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 [1, “7.21 Miscellaneous”]

`\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 { $\langle true \rangle$ } { $\langle false \rangle$ }`

If `\pdfdraftmode` is available and enabled, $\langle true \rangle$ is called, otherwise $\langle false \rangle$ is executed.

`\pdf@setdraftmode { $\langle value \rangle$ }`

Macro `\pdf@setdraftmode` expects the number zero or one as $\langle value \rangle$. 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{0ops, \string\@@input\space is not the %
  original\string\input}%
}

```

1.2.6 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 documentation says, that the used `io.popen` may not be available on all platforms. Then macro `\pdf@pipe` is undefined.

2 Implementation

```

1 <*package>

```

2.1 Reload check and package identification

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

```

2 \begingroup\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{pdf texcmds}{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@pdf texcmds.sty\endcsname
67 \ProvidesPackage{pdf texcmds}%
68 [2020-06-04 v0.32 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{iftex}[2019/11/07]%%
142   \TMP@RequirePackage{ltxcms}[2010/12/02]%
143 \else
144   \RequirePackage{infwarerr}[2007/09/09]%
145   \RequirePackage{iftex}[2019/11/07]%
146   \RequirePackage{ltxcms}[2010/12/02]%
147 \fi

```

2.4 Without LuaTeX

```

148 \ifluatex
149 \else
150   \def\pdf texcmds@nopdftex{%
151     \let\pdf texcmds@nopdftex\relax
152   }%
153   \def\pdf texcmds@temp#1{%
154     \begingroup\expandafter\expandafter\expandafter\endgroup
155     \expandafter\ifx\csname
156       \expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname\relax
157     \pdf texcmds@nopdftex
158   \else
159     \expandafter\def\csname pdf@#1\expandafter\endcsname
160       \expandafter{%
161         \csname\expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname
162       }%
163   \fi
164   }%
165   \pdf texcmds@temp{strcmp}%
166   \pdf texcmds@temp{escapehex}%
167   \let\pdf@escapehexnative\pdf@escapehex
168   \pdf texcmds@temp{unescapehex}%
169   \let\pdf@unescapehexnative\pdf@unescapehex
170   \pdf texcmds@temp{escapestring}%
171   \pdf texcmds@temp{escapename}%
172   \pdf texcmds@temp{filesize}%
173   \pdf texcmds@temp{filemoddate}%
174   \begingroup\expandafter\expandafter\expandafter\endgroup
175   \expandafter\ifx\csname pdfshellescape\endcsname\relax
176     \pdf texcmds@nopdftex
177     \ltx@ifundefined{pdftexversion}{%
178       {%
179         \ifnum\pdftexversion>120 % 1.21a supports \ifeof18
180         \ifeof18 %
181           \chardef\pdf@shellescape=0 %
182         \else
183           \chardef\pdf@shellescape=1 %
184         \fi
185       \fi
186     }%
187   \else

```

```

188 \def\pdf@shellescape{%
189 \pdfshellescape
190 }%
191 \fi
192 \begingroup\expandafter\expandafter\expandafter\endgroup
193 \expandafter\ifx\csname pdf@filedump\endcsname\relax
194 \pdf@texcmds@nopdftex
195 \else
196 \def\pdf@filedump#1#2#3{%
197 \pdf@filedump offset#1 length#2{#3}%
198 }%
199 \fi
200 \begingroup\expandafter\expandafter\expandafter\endgroup
201 \expandafter\ifx\csname pdf@mdfivesum\endcsname\relax
202 \begingroup\expandafter\expandafter\expandafter\endgroup
203 \expandafter\ifx\csname mdfivesum\endcsname\relax
204 \pdf@texcmds@nopdftex
205 \else
206 \def\pdf@mdfivesum#{\mdfivesum}%
207 \let\pdf@mdfivesumnative\pdf@mdfivesum
208 \def\pdf@filemdfivesum#{\mdfivesum file}%
209 \fi
210 \else
211 \def\pdf@mdfivesum#{\pdfmdfivesum}%
212 \let\pdf@mdfivesumnative\pdf@mdfivesum
213 \def\pdf@filemdfivesum#{\pdfmdfivesum file}%
214 \fi
215 \def\pdf@system#{%
216 \immediate\write18%
217 }%
218 \def\pdf@texcmds@temp#1{%
219 \begingroup\expandafter\expandafter\expandafter\endgroup
220 \expandafter\ifx\csname
221 \expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname\relax
222 \pdf@texcmds@nopdftex
223 \else
224 \expandafter\let\csname pdf@#1\expandafter\endcsname
225 \csname\expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname
226 \fi
227 }%
228 \pdf@texcmds@temp{resettimer}%
229 \pdf@texcmds@temp{elapsedtime}%
230 \fi

```

2.5 \pdf@primitive, \pdf@ifprimitive

Since version 1.40.0 pdfTeX has `\pdfprimitive` and `\ifpdfprimitive`. And `\pdfprimitive` was fixed in version 1.40.4.

X_YTeX provides them under the name `\primitive` and `\ifprimitive`. LuaTeX 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 LuaTeX's `tex.enableprimitives`

```

231 \ifluatex

\pdftexcmds@directlua

232 \ifnum\luatexversion<36 %
233 \def\pdftexcmds@directlua{\directlua0}%
234 \else
235 \let\pdftexcmds@directlua\directlua
236 \fi

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

250 \fi

```

2.5.2 Trying various names to find the primitives

```

\pdftexcmds@strip@prefix

251 \def\pdftexcmds@strip@prefix#1>{}

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

\pdf@primitive

267 \pdftexcmds@temp{primitive}{pdfprimitive}{pdfprimitive}% pdfTeX, oldLuaTeX
268 \pdftexcmds@temp{primitive}{primitive}{primitive}% XeTeX, luatex
269 \pdftexcmds@temp{primitive}{luatexpprimitive}{pdfprimitive}% oldLuaTeX
270 \pdftexcmds@temp{primitive}{luatexpdfprimitive}{pdfprimitive}% oldLuaTeX

\pdf@ifprimitive

271 \pdftexcmds@temp{ifprimitive}{ifpdfprimitive}{ifpdfprimitive}% pdfTeX, oldLuaTeX
272 \pdftexcmds@temp{ifprimitive}{ifprimitive}{ifprimitive}% XeTeX, luatex
273 \pdftexcmds@temp{ifprimitive}{luatexifprimitive}{ifpdfprimitive}% oldLuaTeX
274 \pdftexcmds@temp{ifprimitive}{luatexifpdfprimitive}{ifpdfprimitive}% oldLuaTeX

```

Disable broken \pdfprimitive.

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

2.5.3 Result

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

2.6 X_TTEX

Look for primitives \shellescape, \strcmp.

```
312 \def\pdftexcmds@temp#1{%
313   \begingroup\expandafter\expandafter\expandafter\endgroup
314   \expandafter\ifx\csname pdf@#1\endcsname\relax
315     \begingroup
316       \escapechar=-1 %
317       \edef\x{\expandafter\meaning\csname#1\endcsname}%
318       \def\y{#1}%
319       \def\z##1->{%}%
320       \edef\y{\expandafter\z\meaning\y}%
321     \expandafter\endgroup
322   \ifx\x\y
323     \expandafter\def\csname pdf@#1\expandafter\endcsname
324     \expandafter{%
```

```

325         \csname#1\endcsname
326     }%
327     \fi
328 \fi
329 }%
330 \pdfdocmds@temp{shellescape}%
331 \pdfdocmds@temp{strcmp}%

```

2.7 \pdf@isprimitive

```

332 \def\pdf@isprimitive{%
333     \begingroup\expandafter\expandafter\expandafter\endgroup
334     \expandafter\ifx\csname pdf@strcmp\endcsname\relax
335         \long\def\pdf@isprimitive##1{%
336             \expandafter\pdfdocmds@isprimitive\expandafter{\meaning##1}%
337         }%
338         \long\def\pdfdocmds@isprimitive##1##2{%
339             \expandafter\pdfdocmds@@isprimitive\expandafter{\string##2}{##1}%
340         }%
341         \def\pdfdocmds@@isprimitive##1##2{%
342             \ifnum0\pdfdocmds@equal##1\delimiter##2\delimiter=1 %
343                 \expandafter\ltx@firstoftwo
344             \else
345                 \expandafter\ltx@secondoftwo
346             \fi
347         }%
348         \def\pdfdocmds@equal##1##2\delimiter##3##4\delimiter{%
349             \ifx##1##3%
350                 \ifx\relax##2##4\relax
351                     1%
352                 \else
353                     \ifx\relax##2\relax
354                         \else
355                             \ifx\relax##4\relax
356                                 \else
357                                     \pdfdocmds@equalcont{##2}{##4}%
358                                 \fi
359                             \fi
360                         \fi
361                     \fi
362                 }%
363             \def\pdfdocmds@equalcont##1{%
364                 \def\pdfdocmds@equalcont####1####2##1##1##1##1{%
365                     ##1##1##1##1%
366                     \pdfdocmds@equal####1\delimiter####2\delimiter
367                 }%
368             }%
369             \expandafter\pdfdocmds@equalcont\csname fi\endcsname
370         \else
371             \long\def\pdf@isprimitive##1##2{%
372                 \ifnum\pdf@strcmp{\meaning##1}{\string##2}=0 %
373                     \expandafter\ltx@firstoftwo
374                 \else
375                     \expandafter\ltx@secondoftwo
376                 \fi
377             }%
378         \fi
379 }

```

```

380 \ifluatex
381 \ifx\pdfdraftmode\@undefined
382   \let\pdfdraftmode\draftmode
383 \fi
384 \else
385   \pdf@isprimitive
386 \fi

```

2.8 \pdf@draftmode

```

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

```

\pdf@draftmode

```

401   \let\pdf@draftmode\ltx@zero

```

\pdf@ifdraftmode

```

402   \let\pdf@ifdraftmode\ltx@secondoftwo

```

\pdftexcmds@setdraftmode

```

403   \def\pdftexcmds@setdraftmode#1{%

```

```

404 \else

```

\pdftexcmds@draftmode

```

405   \let\pdftexcmds@draftmode\pdfdraftmode

```

\pdf@ifdraftmode

```

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

```

\pdf@draftmode

```

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

```

\pdftexcmds@setdraftmode

```
420 \def\pdftexcmds@setdraftmode#1{%
421   \pdftexcmds@draftmode=#1\relax
422 }%
```

```
423 \fi
```

\pdf@setdraftmode

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

\pdftexcmds@@setdraftmode

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

2.9 Load Lua module

```
444 \ifluatex
445 \else
446   \expandafter\pdftexcmds@AtEnd
447 \fi%

448 \pdftexcmds@directlua{%
449   require("pdf texcmds")%
450 }
451 \ifnum\luatexversion>37 %
452   \ifnum0%
453     \pdftexcmds@directlua{%
454       if status.ini_version then %
455         tex.write("1")%
456       end%
457     }>0 %
458   \everyjob\expandafter{%
459     \the\everyjob
460     \pdftexcmds@directlua{%
461       require("pdf texcmds")%
462     }%
463   }%
464   \fi
465 \fi
466 \begingroup
467 \def\x{2020-06-04 v0.32}%
```



```

468 \ltx@onelevel@sanitize\x
469 \edef\y{%
470   \pdftexcmds@directlua{%
471     if oberdiek.pdftexcmds.getversion then %
472       oberdiek.pdftexcmds.getversion()%
473     end%
474   }%
475 }%
476 \ifx\x\y
477 \else
478   \@PackageError{pdftexcmds}{%
479     Wrong version of lua module.\MessageBreak
480     Package version: \x\MessageBreak
481     Lua module: \y
482   }\@ehc
483 \fi
484 \endgroup

```

2.10 Lua functions

2.10.1 Helper macros

\pdftexcmds@toks

```

485 \beginingroup\expandafter\expandafter\expandafter\endgroup
486 \expandafter\ifx\csname newtoks\endcsname\relax
487   \toksdef\pdftexcmds@toks=0 %
488 \else
489   \csname newtoks\endcsname\pdftexcmds@toks
490 \fi

```

\pdftexcmds@Patch

```

491 \def\pdftexcmds@Patch{0}
492 \ifnum\luatexversion>40 %
493   \ifnum\luatexversion<66 %
494     \def\pdftexcmds@Patch{1}%
495   \fi
496 \fi

497 \ifcase\pdftexcmds@Patch
498   \catcode'\&=14 %
499 \else
500   \catcode'\&=9 %

```

\pdftexcmds@PatchDecode

```

501 \def\pdftexcmds@PatchDecode#1\@nil{%
502   \pdftexcmds@DecodeA#1^^A^^A\@nil{%
503   }%

```

\pdftexcmds@DecodeA

```

504 \def\pdftexcmds@DecodeA#1^^A^^A#2\@nil#3{%
505   \ifx\relax#2\relax
506     \ltx@ReturnAfterElseFi{%
507       \pdftexcmds@DecodeB#3#1^^A^^B\@nil{%
508       }%
509     \else
510       \ltx@ReturnAfterFi{%
511         \pdftexcmds@DecodeA#2\@nil{#3#1^^@}%
512       }%
513     \fi
514   }%

```

\pdftexcmds@DecodeB

```
515 \def\pdftexcmds@DecodeB#1^^A^^B#2\@nil#3{%
516   \ifx\relax#2\relax%
517     \ltx@ReturnAfterElseFi{%
518       \ltx@zero
519       #3#1%
520     }%
521   \else
522     \ltx@ReturnAfterFi{%
523       \pdftexcmds@DecodeB#2\@nil{#3#1^^A}%
524     }%
525   \fi
526 }%

527 \fi

528 \ifnum\luatexversion<36 %
529 \else
530   \catcode'\0=9 %
531 \fi
```

2.10.2 Strings [1, “7.15 Strings”]

\pdf@strcmp

```
532 \long\def\pdf@strcmp#1#2{%
533   \directlua0{%
534     oberdiek.pdf texcmds.strptime("\luaescapestring{#1}",%
535       "\luaescapestring{#2}")%
536   }%
537 }%

538 \pdf@isprimitive
```

\pdf@escapehex

```
539 \long\def\pdf@escapehex#1{%
540   \directlua0{%
541     oberdiek.pdf texcmds.escapehex("\luaescapestring{#1}", "byte")%
542   }%
543 }%
```

\pdf@escapehexnative

```
544 \long\def\pdf@escapehexnative#1{%
545   \directlua0{%
546     oberdiek.pdf texcmds.escapehex("\luaescapestring{#1}")%
547   }%
548 }%
```

\pdf@unescapehex

```
549 \def\pdf@unescapehex#1{%
550 & \romannumeral\expandafter\pdf texcmds@PatchDecode
551 \the\expandafter\pdf texcmds@toks
552 \directlua0{%
553   oberdiek.pdf texcmds.toks="pdf texcmds@toks"%
554   oberdiek.pdf texcmds.unescapehex("\luaescapestring{#1}", "byte", \pdf texcmds@Patch)%
555 }%
556 & \@nil
557 }%
```

\pdf@unescapehexnative

```
558 \def\pdf@unescapehexnative#1{%
559 & \romannumeral\expandafter\pdf texcmds@PatchDecode
560 \the\expandafter\pdf texcmds@toks
561 \directlua0{%
562 oberdiek.pdf texcmds.toks="pdf texcmds@toks"%
563 oberdiek.pdf texcmds.unescapehex("\luaescapestring{#1}", \pdf texcmds@Patch)%
564 }%
565 & \@nil
566 }%
```

\pdf@escapestring

```
567 \long\def\pdf@escapestring#1{%
568 \directlua0{%
569 oberdiek.pdf texcmds.escapestring("\luaescapestring{#1}")%
570 }%
571 }
```

\pdf@escapename

```
572 \long\def\pdf@escapename#1{%
573 \directlua0{%
574 oberdiek.pdf texcmds.escapename("\luaescapestring{#1}", "byte")%
575 }%
576 }
```

\pdf@escapenamenative

```
577 \long\def\pdf@escapenamenative#1{%
578 \directlua0{%
579 oberdiek.pdf texcmds.escapename("\luaescapestring{#1}")%
580 }%
581 }
```

2.10.3 Files [1, “7.18 Files”]

\pdf@filesize

```
582 \def\pdf@filesize#1{%
583 \directlua0{%
584 oberdiek.pdf texcmds.filesize("\luaescapestring{#1}")%
585 }%
586 }
```

\pdf@filemoddate

```
587 \def\pdf@filemoddate#1{%
588 \directlua0{%
589 oberdiek.pdf texcmds.filemoddate("\luaescapestring{#1}")%
590 }%
591 }
```

\pdf@filedump

```
592 \def\pdf@filedump#1#2#3{%
593 \directlua0{%
594 oberdiek.pdf texcmds.filedump("\luaescapestring{\number#1}",%
595 "\luaescapestring{\number#2}",%
596 "\luaescapestring{\number#3}")%
597 }%
598 }%
```

\pdf@mdfivesum

```
599 \long\def\pdf@mdfivesum#1{%
600   \directlua0{%
601     oberdiek.pdf texcmds.mdfivesum("\luaescapestring{#1}", "byte")%
602   }%
603 }%
```

\pdf@mdfivesumnative

```
604 \long\def\pdf@mdfivesumnative#1{%
605   \directlua0{%
606     oberdiek.pdf texcmds.mdfivesum("\luaescapestring{#1}")%
607   }%
608 }%
```

\pdf@filemdfivesum

```
609 \def\pdf@filemdfivesum#1{%
610   \directlua0{%
611     oberdiek.pdf texcmds.filemdfivesum("\luaescapestring{#1}")%
612   }%
613 }%
```

2.10.4 Timekeeping [1, “7.17 Timekeeping”]

\protected

```
614 \let\pdf texcmds@temp=Y%
615 \begingroup\expandafter\expandafter\expandafter\endgroup
616 \expandafter\ifx\csname protected\endcsname\relax
617   \pdf texcmds@directlua0{%
618     if tex.enableprimitives then %
619       tex.enableprimitives('', {'protected'})%
620     end%
621   }%
622 \fi
623 \begingroup\expandafter\expandafter\expandafter\endgroup
624 \expandafter\ifx\csname protected\endcsname\relax
625   \let\pdf texcmds@temp=N%
626 \fi
```

\numexpr

```
627 \begingroup\expandafter\expandafter\expandafter\endgroup
628 \expandafter\ifx\csname numexpr\endcsname\relax
629   \pdf texcmds@directlua0{%
630     if tex.enableprimitives then %
631       tex.enableprimitives('', {'numexpr'})%
632     end%
633   }%
634 \fi
635 \begingroup\expandafter\expandafter\expandafter\endgroup
636 \expandafter\ifx\csname numexpr\endcsname\relax
637   \let\pdf texcmds@temp=N%
638 \fi

639 \ifx\pdf texcmds@temp N%
640   \@PackageWarningNoLine{pdf texcmds}{%
641     Definitions of \ltx@backslashchar pdf@resettimer and%
642     \MessageBreak
643     \ltx@backslashchar pdf@elapsed time are skipped, because%
644     \MessageBreak
```

```

645     e-TeX's \ltx@backslashchar protected or %
646     \ltx@backslashchar numexpr are missing%
647 }%
648 \else

```

`\pdf@resettimer`

```

649 \protected\def\pdf@resettimer{%
650   \pdftexcmds@directlua{%
651     oberdiek.pdftexcmds.resettimer()}%
652   }%
653 }%

```

`\pdf@elapsedtime`

```

654 \protected\def\pdf@elapsedtime{%
655   \numexpr
656   \pdftexcmds@directlua{%
657     oberdiek.pdftexcmds.elapsedtime()}%
658   }%
659   \relax
660 }%

661 \fi

```

2.10.5 Shell escape

`\pdf@shellescape`

```

662 \ifnum\luatexversion<68 %
663 \else
664   \protected\edef\pdf@shellescape{%
665     \numexpr\directlua{tex.sprint(%
666       \number\catcodetable@string,status.shell_escape)}\relax}
667 \fi

```

`\pdf@system`

```

668 \def\pdf@system#1{%
669   \directlua{%
670     oberdiek.pdftexcmds.system("\luaescapestring{#1}")%
671   }%
672 }

```

`\pdf@lastsystemstatus`

```

673 \def\pdf@lastsystemstatus{%
674   \directlua{%
675     oberdiek.pdftexcmds.lastsystemstatus()}%
676   }%
677 }

```

`\pdf@lastsystemexit`

```

678 \def\pdf@lastsystemexit{%
679   \directlua{%
680     oberdiek.pdftexcmds.lastsystemexit()}%
681   }%
682 }

683 \catcode'\0=12 %

```

```

\pdf@pipe Check availability of io.popen first.
684 \ifnum0%
685   \pdftexcmds@directlua{%
686     if io.popen then %
687       tex.write("1")%
688     end%
689   }%
690   =1 %
691 \def\pdf@pipe#1{%
692 &   \romannumeral\expandafter\pdftexcmds@PatchDecode
693 \the\expandafter\pdftexcmds@toks
694 \pdftexcmds@directlua{%
695   oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
696   oberdiek.pdftexcmds.pipe("\luaescapestring{#1}", \pdftexcmds@Patch)%
697 }%
698 &   \@nil
699 }%
700 \fi

701 \pdftexcmds@AtEnd%
702 \</package>

```

2.11 Lua module

```

703 <*lua>

704 oberdiek = oberdiek or {}
705 local pdftexcmds = oberdiek.pdftexcmds or {}
706 oberdiek.pdftexcmds = pdftexcmds
707 local systemexitstatus
708 function pdftexcmds.getversion()
709   tex.write("2020-06-04 v0.32")
710 end

```

2.11.1 Strings [1, “7.15 Strings”]

```

711 function pdftexcmds.strcmp(A, B)
712   if A == B then
713     tex.write("0")
714   elseif A < B then
715     tex.write("-1")
716   else
717     tex.write("1")
718   end
719 end

720 local function utf8_to_byte(str)
721   local i = 0
722   local n = string.len(str)
723   local t = {}
724   while i < n do
725     i = i + 1
726     local a = string.byte(str, i)
727     if a < 128 then
728       table.insert(t, string.char(a))
729     else
730       if a >= 192 and i < n then
731         i = i + 1
732         local b = string.byte(str, i)
733         if b < 128 or b >= 192 then
734           i = i - 1

```

```

735         elseif a == 194 then
736             table.insert(t, string.char(b))
737         elseif a == 195 then
738             table.insert(t, string.char(b + 64))
739         end
740     end
741 end
742 end
743 return table.concat(t)
744 end
745 function pdftexcmds.escapehex(str, mode)
746     if mode == "byte" then
747         str = utf8_to_byte(str)
748     end
749     tex.write((string.gsub(str, ".",
750         function (ch)
751             return string.format("%02X", string.byte(ch))
752         end
753     )))
754 end

```

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

```

755 function pdftexcmds.unescapehex(str, mode, patch)
756     local a = 0
757     local first = true
758     local result = {}
759     for i = 1, string.len(str), 1 do
760         local ch = string.byte(str, i)
761         if ch >= 48 and ch <= 57 then
762             ch = ch - 48
763         elseif ch >= 65 and ch <= 70 then
764             ch = ch - 55
765         elseif ch >= 97 and ch <= 102 then
766             ch = ch - 87
767         else
768             ch = nil
769         end
770         if ch then
771             if first then
772                 a = ch * 16
773                 first = false
774             else
775                 table.insert(result, a + ch)
776                 first = true
777             end
778         end
779     end
780     if not first then
781         table.insert(result, a)
782     end
783     if patch == 1 then
784         local temp = {}
785         for i, a in ipairs(result) do
786             if a == 0 then
787                 table.insert(temp, 1)
788                 table.insert(temp, 1)
789             else
790                 if a == 1 then

```

```

791         table.insert(temp, 1)
792         table.insert(temp, 2)
793     else
794         table.insert(temp, a)
795     end
796 end
797 end
798 result = temp
799 end
800 if mode == "byte" then
801     local utf8 = {}
802     for i, a in ipairs(result) do
803         if a < 128 then
804             table.insert(utf8, a)
805         else
806             if a < 192 then
807                 table.insert(utf8, 194)
808                 a = a - 128
809             else
810                 table.insert(utf8, 195)
811                 a = a - 192
812             end
813             table.insert(utf8, a + 128)
814         end
815     end
816     result = utf8
817 end

```

this next line added for current luatex; this is the only change in the file. eroux, 28apr13. (v 0.21)

```

818 local unpack = _G["unpack"] or table.unpack
819 tex.settoks(pdftexcmds.toks, string.char(unpack(result)))
820 end

```

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

```

821 function pdftexcmds.escapestring(str, mode)
822     if mode == "byte" then
823         str = utf8_to_byte(str)
824     end
825     tex.write((string.gsub(str, ".",
826         function (ch)
827             local b = string.byte(ch)
828             if b < 33 or b > 126 then
829                 return string.format("\\%.3o", b)
830             end
831             if b == 40 or b == 41 or b == 92 then
832                 return "\\\" .. ch
833             end
834         end
835     )))
836 end

```

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

```

834         return nil
835     end
836 )))
837 end

```

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

```

838 function pdftexcmds.escapename(str, mode)
839     if mode == "byte" then
840         str = utf8_to_byte(str)
841     end
842     tex.write((string.gsub(str, ".",

```



```

843 function (ch)
844     local b = string.byte(ch)
845     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.

```

846         return ""
847     end
848     if b <= 32 or b >= 127
849         or b == 35 or b == 37 or b == 40 or b == 41
850         or b == 47 or b == 60 or b == 62 or b == 91
851         or b == 93 or b == 123 or b == 125 then
852         return string.format("#%.2X", b)
853     else

```

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

```

854         return nil
855     end
856 end
857 )))
858 end

```

2.11.2 Files [1, “7.18 Files”]

```

859 function pdftexcmds.filesize(filename)
860     local foundfile = kpse.find_file(filename, "tex", true)
861     if foundfile then
862         local size = lfs.attributes(foundfile, "size")
863         if size then
864             tex.write(size)
865         end
866     end
867 end

```

See procedure makepdftime in file utils.c of pdfTEX.

```

868 function pdftexcmds.filemoddate(filename)
869     local foundfile = kpse.find_file(filename, "tex", true)
870     if foundfile then
871         local date = lfs.attributes(foundfile, "modification")
872         if date then
873             local d = os.date("!*t", date)
874             if d.sec >= 60 then
875                 d.sec = 59
876             end
877             local u = os.date("!*t", date)
878             local off = 60 * (d.hour - u.hour) + d.min - u.min
879             if d.year ~= u.year then
880                 if d.year > u.year then
881                     off = off + 1440
882                 else
883                     off = off - 1440
884                 end
885             elseif d.yday ~= u.yday then
886                 if d.yday > u.yday then
887                     off = off + 1440
888                 else
889                     off = off - 1440
890                 end
891             end
892             local timezone
893             if off == 0 then

```

```

894         timezone = "Z"
895     else
896         local hours = math.floor(off / 60)
897         local mins = math.abs(off - hours * 60)
898         timezone = string.format("%+03d'%02d'", hours, mins)
899     end
900     tex.write(string.format("D:%04d%02d%02d%02d%02d%s",
901         d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
902 end
903 end
904 end
905 function pdftexcmds.filedump(offset, length, filename)
906     length = tonumber(length)
907     if length and length > 0 then
908         local foundfile = kpse.find_file(filename, "tex", true)
909         if foundfile then
910             offset = tonumber(offset)
911             if not offset then
912                 offset = 0
913             end
914             local filehandle = io.open(foundfile, "rb")
915             if filehandle then
916                 if offset > 0 then
917                     filehandle:seek("set", offset)
918                 end
919                 local dump = filehandle:read(length)
920                 pdftexcmds.escapehex(dump)
921                 filehandle:close()
922             end
923         end
924     end
925 end
926 function pdftexcmds.md5sum(str, mode)
927     if mode == "byte" then
928         str = utf8_to_byte(str)
929     end
930     pdftexcmds.escapehex(md5.sum(str))
931 end
932 function pdftexcmds.filemd5sum(filename)
933     local foundfile = kpse.find_file(filename, "tex", true)
934     if foundfile then
935         local filehandle = io.open(foundfile, "rb")
936         if filehandle then
937             local contents = filehandle:read("*a")
938             pdftexcmds.escapehex(md5.sum(contents))
939             filehandle:close()
940         end
941     end
942 end

```

2.11.3 Timekeeping [1, “7.17 Timekeeping”]

The functions for timekeeping are based on Andy Thomas’ work [3]. 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`.

```

943 local basetime = 0
944 function pdftexcmds.resettimer()
945   basetime = os.clock()
946 end
947 function pdftexcmds.elapsedtime()
948   local val = (os.clock() - basetime) * 65536 + .5
949   if val > 2147483647 then
950     val = 2147483647
951   end
952   tex.write(string.format("%d", math.floor(val)))
953 end

```

2.11.4 Miscellaneous [1, “7.21 Miscellaneous”]

```

954 function pdftexcmds.shellescape()
955   if os.execute then
956     if status
957       and status.luatex_version
958       and status.luatex_version >= 68 then
959       tex.write(os.execute())
960     else
961       local result = os.execute()
962       if result == 0 then
963         tex.write("0")
964       else
965         if result == nil then
966           tex.write("0")
967         else
968           tex.write("1")
969         end
970       end
971     end
972   else
973     tex.write("0")
974   end
975 end
976 function pdftexcmds.system(cmdline)
977   systemexitstatus = nil
978   texio.write_nl("log", "system(" .. cmdline .. ") ")
979   if os.execute then
980     texio.write("log", "executed.")
981     systemexitstatus = os.execute(cmdline)
982   else
983     texio.write("log", "disabled.")
984   end
985 end
986 function pdftexcmds.lastsystemstatus()
987   local result = tonumber(systemexitstatus)
988   if result then
989     local x = math.floor(result / 256)
990     tex.write(result - 256 * math.floor(result / 256))
991   end
992 end
993 function pdftexcmds.lastsystemexit()
994   local result = tonumber(systemexitstatus)
995   if result then
996     tex.write(math.floor(result / 256))
997   end

```

```

998 end
999 function pdftexcmds.pipe(cmdline, patch)
1000   local result
1001   systemexitstatus = nil
1002   texio.write_nl("log", "pipe(" .. cmdline .. ") ")
1003   if io.popen then
1004     texio.write("log", "executed.")
1005     local handle = io.popen(cmdline, "r")
1006     if handle then
1007       result = handle:read("*a")
1008       handle:close()
1009     end
1010   else
1011     texio.write("log", "disabled.")
1012   end
1013   if result then
1014     if patch == 1 then
1015       local temp = {}
1016       for i, a in ipairs(result) do
1017         if a == 0 then
1018           table.insert(temp, 1)
1019           table.insert(temp, 1)
1020         else
1021           if a == 1 then
1022             table.insert(temp, 1)
1023             table.insert(temp, 2)
1024           else
1025             table.insert(temp, a)
1026           end
1027         end
1028       end
1029       result = temp
1030     end
1031     tex.settoks(pdftexcmds.toks, result)
1032   else
1033     tex.settoks(pdftexcmds.toks, "")
1034   end
1035 end
1036  $\langle$ /lua $\rangle$ 

```

3 Installation

3.1 Download

Package. This package is available on CTAN¹:

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

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

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

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

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

¹[CTAN:pkg/pdftexcmds](#)

3.2 Bundle installation

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

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

Script installation. Check the directory `TDS:scripts/pdftexcmds/` for scripts that need further installation steps.

3.3 Package installation

Unpacking. The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain `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/pdftexcmds/pdftexcmds.sty
pdftexcmds.lua → scripts/pdftexcmds/pdftexcmds.lua
pdftexcmds.pdf → doc/latex/pdftexcmds/pdftexcmds.pdf
pdftexcmds.dtx → source/latex/pdftexcmds/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`.

3.4 Refresh file name databases

If your `TeX` distribution (`TeX Live`, `MiKTeX`, ...) relies on file name databases, you must refresh these. For example, `TeX Live` users run `texhash` or `mktextlsr`.

3.5 Some details for the interested

Unpacking with \LaTeX . The `.dtx` chooses its action depending on the format:

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

\LaTeX : Generate the documentation.

If you insist on using \LaTeX for `docstrip` (really, `docstrip` does not need \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 \LaTeX` :

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

4 References

- [1] Hàn Thê Thành et al. *The pdfTeX user manual*. Version 655 (1.40.11). 2010-11-23. URL: <http://mirror.ctan.org/systems/pdftex/manual/pdftex-a.pdf> (visited on 2011-11-29).
- [2] LuaTeX development team. *LuaTeX Reference*. Version beta 0.71.0. 2011-10-11. URL: <http://www.luatex.org/svn/trunk/manual/luatex.pdf> (visited on 2011-11-29).
- [3] Andy Thomas. *Analog of \pdfelapsedtime for LuaTeX and XeTeX*. URL: <http://tex.stackexchange.com/a/32531> (visited on 2011-11-29).

5 History

[2007/11/11 v0.1]

- First version.

[2007/11/12 v0.2]

- Short description fixed.

[2007/12/12 v0.3]

- Organization of Lua code as module.

[2009/04/10 v0.4]

- Adaptation for syntax change of `\directlua` in LuaTeX 0.36.

[2009/09/22 v0.5]

- `\pdf@primitive`, `\pdf@ifprimitive` added.
- XeTeX's variants are detected for `\pdf@shellescape`, `\pdf@strcmp`, `\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 `XYTeX`

[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.

[2019/07/25 v0.30]

- Remove uses of module function, see PR70

[2019/11/24 v0.31]

- Use `iftex` directly rather than `ifluatex` and `ifpdf` wrappers.
- detect `\filmoddate` and other `XYTeX` commands.
- Adjust `\pdf@escapestring` in `LuaTeX` to produce the same as in `pdfTeX` in the 8bit range and not drop all non ascii characters.

[2020-06-04 v0.32]

- Updated `pdftexcmds.elapsedtime` to lua 5.3 (issue 4).

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

Symbols	
<code>\&</code>	498, 500
<code>\@PackageError</code>	478
<code>\@PackageInfoNoLine</code>	287, 301, 306, 389, 393, 395
<code>\@PackageWarning</code>	438
<code>\@PackageWarningNoLine</code>	640
<code>\@ehc</code>	482
<code>\@nil</code>	501, 502, 504, 507, 511, 515, 523, 556, 565, 698
<code>\@undefined</code>	58, 286, 381
<code>\@</code>	829, 832
Numbers	
<code>\0</code>	530, 683
A	
<code>\aftergroup</code>	29
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, 498, 500, 530, 683
<code>\catcodetable@string</code>	666
<code>\chardef</code>	181, 183
<code>\count</code>	426, 428
<code>\csname</code>	14, 21, 50, 66, 76, 133, 136, 155, 156, 159, 161, 175, 193, 201, 203, 220, 221, 224, 225, 254, 259, 262, 263, 277, 279, 282, 303, 308, 314, 317, 323, 325, 334, 369, 486, 489, 616, 624, 628, 636
D	
<code>\delimiter</code>	342, 348, 366
<code>\directlua</code>	233, 235, 533, 540, 545, 552, 561, 568, 573, 578, 583, 588, 593, 600, 605, 610, 665, 669, 674, 679
<code>\draftmode</code>	382
E	
<code>\empty</code>	17, 18
<code>\endcsname</code>	14, 21, 50, 66, 76, 133, 136, 156, 159, 161, 175, 193, 201, 203, 221, 224, 225, 254, 259, 262, 263, 277, 279, 282, 303, 308, 314, 317, 323, 325, 334, 369, 486, 489, 616, 624, 628, 636
<code>\endinput</code>	29, 129
<code>\endlinechar</code>	4, 35, 71, 77, 89, 239
<code>\escapechar</code>	128, 131, 258, 316
<code>\everyjob</code>	458, 459
I	
<code>\ifcase</code>	400, 433, 497
<code>\ifeof</code>	179, 180
<code>\ifluatex</code>	148, 231, 275, 380, 444
<code>\ifnum</code>	179, 232, 281, 284, 342, 372, 407, 414, 451, 452, 492, 493, 528, 662, 684
<code>\ifpdf</code>	391
<code>\ifx</code>	15, 18, 21, 50, 58, 61, 133, 136, 155, 156, 161, 175, 193, 201, 203, 220, 221, 225, 254, 261, 277, 279, 282, 303, 308, 314, 322, 334, 349, 350, 353, 355, 381, 476, 486, 505, 516, 616, 624, 628, 636, 639
<code>\immediate</code>	23, 52, 216
<code>\input</code>	137
L	
<code>\ltx@backslashchar</code>	389, 393, 396, 641, 643, 645, 646
<code>\ltx@ccclv</code>	426, 428
<code>\ltx@firstoftwo</code>	343, 373, 408
<code>\ltx@ifUndefined</code>	177, 388
<code>\ltx@one</code>	392, 407, 414, 415
<code>\ltx@onelevel@sanitize</code>	468
<code>\ltx@ReturnAfterElseFi</code>	506, 517
<code>\ltx@ReturnAfterFi</code>	510, 522
<code>\ltx@secondoftwo</code> ..	345, 375, 402, 410
<code>\ltx@zero</code>	387, 401, 417, 518
<code>\luaescapestring</code>	534, 535, 541, 546, 554, 563, 569, 574, 579, 584, 589, 594, 595, 596, 601, 606, 611, 670, 696
<code>\luatexversion</code>	232, 451, 492, 493, 528, 662
M	
<code>\mdfivesum</code>	206, 208
<code>\meaning</code> ..	257, 259, 317, 320, 336, 372
<code>\MessageBreak</code>	289, 439, 479, 480, 642, 644
N	
<code>\newlinechar</code>	238, 239
<code>\number</code>	128, 594, 595, 666
<code>\numexpr</code>	<u>627</u> , 655, 665

P

`\PackageInfo` 26
`\pdf@draftmode` 5, 401, 413
`\pdf@elapsedtime` 4, 654
`\pdf@escapehex` 4, 167, 539
`\pdf@escapehexnative` 167, 544
`\pdf@escapename` 572
`\pdf@escapenamenative` 577
`\pdf@escapestring` 567
`\pdf@filedump` 4, 196, 592
`\pdf@filemdfivesum` ... 4, 208, 213, 609
`\pdf@filemoddate` 4, 587
`\pdf@filesize` 4, 582
`\pdf@ifdraftmode` 5, 402, 406
`\pdf@ifprimitive` 6, 271, 307
`\pdf@isprimitive`
..... 6, 332, 335, 371, 385, 538
`\pdf@lastsystemexit` 678
`\pdf@lastsystemstatus` 673
`\pdf@mdfivesum` 4, 206, 207, 211, 212, 599
`\pdf@mdfivesumnative` .. 207, 212, 604
`\pdf@pipe` 7, 684
`\pdf@primitive` .. 6, 267, 286, 288, 302
`\pdf@resettimer` 4, 649
`\pdf@setdraftmode` 5, 424, 439
`\pdf@shellescape` 5, 181, 183, 188, 662
`\pdf@strcmp` 3, 372, 532
`\pdf@system` 6, 215, 668
`\pdf@unescapehex` 4, 169, 549
`\pdf@unescapehexnative` .. 7, 169, 558
`\pdf@draftmode` 381, 382, 405
`\pdf@filedump` 197
`\pdf@mdfivesum` 211, 213
`\pdf@primitive` 290
`\pdf@shellescape` 189
`\pdf@texcmds@isprimitive` ... 339, 341
`\pdf@texcmds@setdraftmode` .. 428, 432
`\pdf@texcmds@AtEnd`
..... 95, 96, 126, 127, 446, 701
`\pdf@texcmds@DecodeA` 502, 504
`\pdf@texcmds@DecodeB` 507, 515
`\pdf@texcmds@directlua`
..... 232, 240, 448, 453, 460,
470, 617, 629, 650, 656, 685, 694
`\pdf@texcmds@draftmode`
..... 405, 407, 414, 421
`\pdf@texcmds@equal` 342, 348, 366
`\pdf@texcmds@equalcont`
..... 357, 363, 364, 369
`\pdf@texcmds@isprimitive` ... 336, 338

`\pdf@texcmds@nopdfTeX`
. 150, 151, 157, 176, 194, 204, 222
`\pdf@texcmds@Patch`
..... 491, 497, 554, 563, 696
`\pdf@texcmds@PatchDecode`
..... 501, 550, 559, 692
`\pdf@texcmds@setdraftmode`
..... 403, 420, 434, 436
`\pdf@texcmds@strip@prefix` ... 251, 257
`\pdf@texcmds@temp`
..... 153, 165, 166, 168, 170,
171, 172, 173, 218, 228, 229,
252, 267, 268, 269, 270, 271,
272, 273, 274, 312, 330, 331,
387, 392, 400, 614, 625, 637, 639
`\pdf@texcmds@toks` .. 485, 551, 560, 693
`\pdf@tex@revision` 284
`\pdf@tex@version` 179, 281
`\protected` 614, 649, 654, 664
`\ProvidesPackage` 19, 67

R

`\RequirePackage` 144, 145, 146
`\romannumeral` 550, 559, 692

S

`\space` 288, 290, 302, 307

T

`\the` 77, 78, 79, 80, 81, 82,
83, 84, 97, 428, 459, 551, 560, 693
`\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
`\toksdef` 487

W

`\write` 23, 52, 216

X

`\x` ... 14, 15, 18, 22, 26, 28, 51, 56,
66, 75, 87, 256, 257, 261, 317,
322, 427, 430, 467, 468, 476, 480

Y

`\y` 259, 261, 318, 320, 322, 469, 476, 481

Z

`\z` 319, 320