

The pdf_{TeX}cmds package

Heiko Oberdiek*

2020-06-27 v0.33

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>	14
2.8	<code>\pdf@draftmode</code>	15
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	Package installation	28
3.3	Refresh file name databases	29
3.4	Some details for the interested	29
4	References	29
5	History	29
	[2007/11/11 v0.1]	29
	[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]	30
	[2010/11/11 v0.11]	30
	[2011/01/30 v0.12]	30
	[2011/03/04 v0.13]	30
	[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]	31
	[2016/10/02 v0.23]	31
	[2017/01/29 v0.24]	31
	[2017/03/19 v0.25]	31
	[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
	[2020-06-24 v0.33]	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 $\langle general\ text \rangle$ for the other arguments. Using token register 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}

```

Lua \TeX 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 pdf \TeX provides `\pdf $\langle cmd \rangle$` .

Arguments: The order of arguments in `\pdf@ $\langle cmd \rangle$` is the same as for the corresponding primitive of pdf \TeX . 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 pdf \TeX 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 Lua \TeX : The macros `\pdf@ $\langle cmd \rangle$` are mapped to the commands of pdf \TeX 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 pdf_TE_X. However pdf_TE_X 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 T_EX 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 {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@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@pdf texcmds.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 [2020-06-27 v0.33 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   \ifcsname catcodetable@string\endcsname\else\input{ltluatex}\fi
150 \else
151   \def\pdftexcmds@nopdftex{%
152     \let\pdftexcmds@nopdftex\relax
153   }%
154   \def\pdftexcmds@temp#1{%
155     \begingroup\expandafter\expandafter\expandafter\endgroup
156     \expandafter\ifx\csname
157       \expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname\relax
158     \pdftexcmds@nopdftex
159     \else
160       \expandafter\def\csname pdf@#1\expandafter\endcsname
161         \expandafter{%
162           \csname\expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname
163         }%
164     \fi
165   }%
166   \pdftexcmds@temp{strcmp}%
167   \pdftexcmds@temp{escapehex}%
168   \let\pdf@escapehexnative\pdf@escapehex
169   \pdftexcmds@temp{unescapehex}%
170   \let\pdf@unescapehexnative\pdf@unescapehex
171   \pdftexcmds@temp{escapestring}%
172   \pdftexcmds@temp{escapename}%
173   \pdftexcmds@temp{filesize}%
174   \pdftexcmds@temp{filemoddate}%
175   \begingroup\expandafter\expandafter\expandafter\endgroup
176   \expandafter\ifx\csname pdfshellescape\endcsname\relax
177     \pdftexcmds@nopdftex
178     \ltx@ifundefined{pdftexversion}{%
179       {%
180         \ifnum\pdftexversion>120 % 1.21a supports \ifeof18
181         \ifeof18 %
182           \chardef\pdf@shellescape=0 %
183         \else
184           \chardef\pdf@shellescape=1 %
185         \fi
186       }%
187     }%

```

```

188 \else
189   \def\pdf@shellescape{%
190     \pdfshellescape
191   }%
192 \fi
193 \begingroup\expandafter\expandafter\expandafter\endgroup
194 \expandafter\ifx\csname pdffiledump\endcsname\relax
195   \pdfdocmds@nopdfdoc
196 \else
197   \def\pdf@filedump#1#2#3{%
198     \pdffiledump offset#1 length#2{#3}%
199   }%
200 \fi

201 \begingroup\expandafter\expandafter\expandafter\endgroup
202 \expandafter\ifx\csname pdfmdfivesum\endcsname\relax
203   \begingroup\expandafter\expandafter\expandafter\endgroup
204   \expandafter\ifx\csname mdfivesum\endcsname\relax
205     \pdfdocmds@nopdfdoc
206   \else
207     \def\pdf@mdfivesum#\mdfivesum}%
208     \let\pdf@mdfivesumnative\pdf@mdfivesum
209     \def\pdf@filemdfivesum#\mdfivesum file}%
210   \fi
211 \else
212   \def\pdf@mdfivesum#\pdfmdfivesum}%
213   \let\pdf@mdfivesumnative\pdf@mdfivesum
214   \def\pdf@filemdfivesum#\pdfmdfivesum file}%
215 \fi

216 \def\pdf@system#{%
217   \immediate\write18%
218 }%
219 \def\pdfdocmds@temp#1{%
220   \begingroup\expandafter\expandafter\expandafter\endgroup
221   \expandafter\ifx\csname
222     \expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname\relax
223     \pdfdocmds@nopdfdoc
224   \else
225     \begingroup\expandafter\expandafter\expandafter\endgroup
226     \expandafter\let\csname pdf@#1\endcsname
227     \csname\expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname
228     \fi
229   }%
230 \pdfdocmds@temp{resettimer}%
231 \pdfdocmds@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_YTEX 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 LuaTeX's tex.enableprimitives

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

239 \begingroup
240   \newlinechar=10 %
241   \endlinechar=\newlinechar
242   \pdftexcmds@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

```
\pdftexcmds@strip@prefix
253 \def\pdftexcmds@strip@prefix#1>{}

254 \def\pdftexcmds@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\pdftexcmds@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\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}{luatexpprimitive}{pdfprimitive}% oldLuaTeX
272 \pdftexcmds@temp{primitive}{luatexpdfprimitive}{pdfprimitive}% oldLuaTeX

\pdf@ifprimitive
273 \pdftexcmds@temp{ifprimitive}{ifpdfprimitive}{ifpdfprimitive}% pdfTeX, oldLuaTeX
274 \pdftexcmds@temp{ifprimitive}{ifprimitive}{ifprimitive}% XeTeX, luatex
275 \pdftexcmds@temp{ifprimitive}{luatexpifprimitive}{ifpdfprimitive}% oldLuaTeX
276 \pdftexcmds@temp{ifprimitive}{luatexpifpdfprimitive}{ifpdfprimitive}% oldLuaTeX
```

Disable broken `\pdfprimitive`.

```
277 \ifluatex\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 \endgroup
300 \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_TTEX

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

```
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       \ifx##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{%
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 \ifluatex
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@ccclv=#1\relax  
429   \edef\x{\endgroup  
430     \noexpand\pdftexcmds@@setdraftmode{\the\count\ltx@ccclv}}%  
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 \pdftexcmds@directlua{%  
451   require("pdftexcmds")%  
452 }%  
453 \ifnum\luatexversion>37 %  
454   \ifnum0%  
455     \pdftexcmds@directlua{%  
456       if status.ini_version then %  
457         tex.write("1")%  
458       end%  
459     }>0 %  
460   \everyjob\expandafter{%  
461     \the\everyjob  
462     \pdftexcmds@directlua{%  
463       require("pdftexcmds")%  
464     }%  
465   }%  
466 \fi  
467 \fi  
468 \begingroup  
469 \def\x{2020-06-27 v0.33}%
```



```

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

```

2.10 Lua functions

2.10.1 Helper macros

`\pdftexcmds@toks`

```

487 \begingroup\expandafter\expandafter\expandafter\endgroup
488 \expandafter\ifx\csname newtoks\endcsname\relax
489   \toksdef\pdftexcmds@toks=0 %
490 \else
491   \csname newtoks\endcsname\pdftexcmds@toks
492 \fi

```

`\pdftexcmds@Patch`

```

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

499 \ifcase\pdftexcmds@Patch
500   \catcode'\&=14 %
501 \else
502   \catcode'\&=9 %

```

`\pdftexcmds@PatchDecode`

```

503 \def\pdftexcmds@PatchDecode#1\@nil{%
504   \pdftexcmds@DecodeA#1^^A^^A\@nil{%
505 }%

```

`\pdftexcmds@DecodeA`

```

506 \def\pdftexcmds@DecodeA#1^^A^^A#2\@nil#3{%
507   \ifx\relax#2\relax
508     \ltx@ReturnAfterElseFi{%
509       \pdftexcmds@DecodeB#3#1^^A^^B\@nil{%
510     }%
511   \else
512     \ltx@ReturnAfterFi{%
513       \pdftexcmds@DecodeA#2\@nil{#3#1^^@}%
514     }%
515   \fi
516 }%

```

`\pdf texcmds@DecodeB`

```
517 \def\pdf texcmds@DecodeB#1^^A^^B#2\@nil#3{%
518   \ifx\relax#2\relax%
519     \ltx@ReturnAfterElseFi{%
520       \ltx@zero
521       #3#1%
522     }%
523   \else
524     \ltx@ReturnAfterFi{%
525       \pdf texcmds@DecodeB#2\@nil{#3#1^^A}%
526     }%
527   \fi
528 }%

529 \fi

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

2.10.2 Strings [1, “7.15 Strings”]

`\pdf@strcmp`

```
534 \long\def\pdf@strcmp#1#2{%
535   \directlua0{%
536     oberdiek.pdf texcmds.strcmp("\luaescapestring{#1}",%
537       "\luaescapestring{#2}")%
538   }%
539 }%

540 \pdf@isprimitive
```

`\pdf@escapehex`

```
541 \long\def\pdf@escapehex#1{%
542   \directlua0{%
543     oberdiek.pdf texcmds.escapehex("\luaescapestring{#1}", "byte")%
544   }%
545 }%
```

`\pdf@escapehexnative`

```
546 \long\def\pdf@escapehexnative#1{%
547   \directlua0{%
548     oberdiek.pdf texcmds.escapehex("\luaescapestring{#1}")%
549   }%
550 }%
```

`\pdf@unescapehex`

```
551 \def\pdf@unescapehex#1{%
552 & \romannumeral\expandafter\pdf texcmds@PatchDecode
553 \the\expandafter\pdf texcmds@toks
554 \directlua0{%
555   oberdiek.pdf texcmds.toks="pdf texcmds@toks"%
556   oberdiek.pdf texcmds.unescapehex("\luaescapestring{#1}", "byte", \pdf texcmds@Patch)%
557 }%
558 & \@nil
559 }%
```

`\pdf@unescapehexnative`

```
560 \def\pdf@unescapehexnative#1{%
561 & \romannumeral\expandafter\pdf texcmds@PatchDecode
562 \the\expandafter\pdf texcmds@toks
563 \directlua0{%
564 oberdiek.pdf texcmds.toks="pdf texcmds@toks"%
565 oberdiek.pdf texcmds.unescapehex("\luaescapestring{#1}", \pdf texcmds@Patch)%
566 }%
567 & \@nil
568 }%
```

`\pdf@escapestring`

```
569 \long\def\pdf@escapestring#1{%
570 \directlua0{%
571 oberdiek.pdf texcmds.escapestring("\luaescapestring{#1}")%
572 }%
573 }
```

`\pdf@escapename`

```
574 \long\def\pdf@escapename#1{%
575 \directlua0{%
576 oberdiek.pdf texcmds.escapename("\luaescapestring{#1}", "byte")%
577 }%
578 }
```

`\pdf@escapenamename`

```
579 \long\def\pdf@escapenamename#1{%
580 \directlua0{%
581 oberdiek.pdf texcmds.escapename("\luaescapestring{#1}")%
582 }%
583 }
```

2.10.3 Files [1, “7.18 Files”]

`\pdf@filesize`

```
584 \def\pdf@filesize#1{%
585 \directlua0{%
586 oberdiek.pdf texcmds.filesize("\luaescapestring{#1}")%
587 }%
588 }
```

`\pdf@filemoddate`

```
589 \def\pdf@filemoddate#1{%
590 \directlua0{%
591 oberdiek.pdf texcmds.filemoddate("\luaescapestring{#1}")%
592 }%
593 }
```

`\pdf@filedump`

```
594 \def\pdf@filedump#1#2#3{%
595 \directlua0{%
596 oberdiek.pdf texcmds.filedump("\luaescapestring{\number#1}",%
597 "\luaescapestring{\number#2}",%
598 "\luaescapestring{\number#3}")%
599 }%
600 }%
```

`\pdf@mdfivesum`

```
601 \long\def\pdf@mdfivesum#1{%
602   \directlua0{%
603     oberdiek.pdfdoccmds.mdfivesum("\luaescapestring{#1}", "byte")%
604   }%
605 }%
```

`\pdf@mdfivesumnative`

```
606 \long\def\pdf@mdfivesumnative#1{%
607   \directlua0{%
608     oberdiek.pdfdoccmds.mdfivesum("\luaescapestring{#1}")%
609   }%
610 }%
```

`\pdf@filemdfivesum`

```
611 \def\pdf@filemdfivesum#1{%
612   \directlua0{%
613     oberdiek.pdfdoccmds.filemdfivesum("\luaescapestring{#1}")%
614   }%
615 }%
```

2.10.4 Timekeeping [1, “7.17 Timekeeping”]

`\protected`

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

`\numexpr`

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

641 \ifx\pdfdoccmds@temp N%
642   \@PackageWarningNoLine{pdfdoccmds}{%
643     Definitions of \ltx@backslashchar pdf@resettimer and%
644     \MessageBreak
645     \ltx@backslashchar pdf@elapsedtime are skipped, because%
646     \MessageBreak
```

```

647 e-TeX's \ltx@backslashchar protected or %
648 \ltx@backslashchar numexpr are missing%
649 }%
650 \else

```

`\pdf@resettimer`

```

651 \protected\def\pdf@resettimer{%
652 \pdfptexc@directlua0{%
653 oberdiek.pdfptexc@resettimer()}%
654 }%
655 }%

```

`\pdf@elapsedtime`

```

656 \protected\def\pdf@elapsedtime{%
657 \numexpr
658 \pdfptexc@directlua0{%
659 oberdiek.pdfptexc@elapsedtime()}%
660 }%
661 \relax
662 }%

663 \fi

```

2.10.5 Shell escape

`\pdf@shellescape`

```

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

```

`\pdf@system`

```

670 \def\pdf@system#1{%
671 \directlua0{%
672 oberdiek.pdfptexc@system("\luaescapestring{#1}")%
673 }%
674 }

```

`\pdf@lastsystemstatus`

```

675 \def\pdf@lastsystemstatus{%
676 \directlua0{%
677 oberdiek.pdfptexc@lastsystemstatus()}%
678 }%
679 }

```

`\pdf@lastsystemexit`

```

680 \def\pdf@lastsystemexit{%
681 \directlua0{%
682 oberdiek.pdfptexc@lastsystemexit()}%
683 }%
684 }

```

```

685 \catcode'\0=12 %

```

```

\pdf@pipe Check availability of io.popen first.
686 \ifnum0%
687   \pdftexcmds@directlua{%
688     if io.popen then %
689       tex.write("1")%
690     end%
691   }%
692   =1 %
693 \def\pdf@pipe#1{%
694 &   \romannumeral\expandafter\pdftexcmds@PatchDecode
695 \the\expandafter\pdftexcmds@toks
696   \pdftexcmds@directlua{%
697     oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
698     oberdiek.pdftexcmds.pipe("\luaescapestring{#1}", \pdftexcmds@Patch)%
699   }%
700 &   \@nil
701 }%
702 \fi

703 \pdftexcmds@AtEnd%
704 \endpackage

```

2.11 Lua module

```

705 (*lua)
706 oberdiek = oberdiek or {}
707 local pdftexcmds = oberdiek.pdftexcmds or {}
708 oberdiek.pdftexcmds = pdftexcmds
709 local systemexitstatus
710 function pdftexcmds.getversion()
711   tex.write("2020-06-27 v0.33")
712 end

```

2.11.1 Strings [1, “7.15 Strings”]

```

713 function pdftexcmds.strcmp(A, B)
714   if A == B then
715     tex.write("0")
716   elseif A < B then
717     tex.write("-1")
718   else
719     tex.write("1")
720   end
721 end
722 local function utf8_to_byte(str)
723   local i = 0
724   local n = string.len(str)
725   local t = {}
726   while i < n do
727     i = i + 1
728     local a = string.byte(str, i)
729     if a < 128 then
730       table.insert(t, string.char(a))
731     else
732       if a >= 192 and i < n then
733         i = i + 1
734         local b = string.byte(str, i)
735         if b < 128 or b >= 192 then
736           i = i - 1

```

```

737     elseif a == 194 then
738         table.insert(t, string.char(b))
739     elseif a == 195 then
740         table.insert(t, string.char(b + 64))
741     end
742 end
743 end
744 end
745 return table.concat(t)
746 end
747 function pdftexcmds.escapehex(str, mode)
748     if mode == "byte" then
749         str = utf8_to_byte(str)
750     end
751     tex.write((string.gsub(str, ".",
752         function (ch)
753             return string.format("%02X", string.byte(ch))
754         end
755     )))
756 end

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

836         return nil
837     end
838 )))
839 end

```

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

```

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

```



```

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

```

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

```

856             return nil
857         end
858     end
859 )))
860 end

```

2.11.2 Files [1, “7.18 Files”]

```

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

```

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

```

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

```

```

896     timezone = "Z"
897 else
898     local hours = math.floor(off / 60)
899     local mins = math.abs(off - hours * 60)
900     timezone = string.format("%+03d'%02d'", hours, mins)
901 end
902 tex.write(string.format("D:%04d%02d%02d%02d%02d%02d%s",
903     d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
904 end
905 end
906 end
907 function pdftexcmds.filedump(offset, length, filename)
908     length = tonumber(length)
909     if length and length > 0 then
910         local foundfile = kpse.find_file(filename, "tex", true)
911         if foundfile then
912             offset = tonumber(offset)
913             if not offset then
914                 offset = 0
915             end
916             local filehandle = io.open(foundfile, "rb")
917             if filehandle then
918                 if offset > 0 then
919                     filehandle:seek("set", offset)
920                 end
921                 local dump = filehandle:read(length)
922                 pdftexcmds.escapehex(dump)
923                 filehandle:close()
924             end
925         end
926     end
927 end
928 function pdftexcmds.md5sum(str, mode)
929     if mode == "byte" then
930         str = utf8_to_byte(str)
931     end
932     pdftexcmds.escapehex(md5.sum(str))
933 end
934 function pdftexcmds.filemd5sum(filename)
935     local foundfile = kpse.find_file(filename, "tex", true)
936     if foundfile then
937         local filehandle = io.open(foundfile, "rb")
938         if filehandle then
939             local contents = filehandle:read("*a")
940             pdftexcmds.escapehex(md5.sum(contents))
941             filehandle:close()
942         end
943     end
944 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`.

```

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

```

2.11.4 Miscellaneous [1, “7.21 Miscellaneous”]

```

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

```

```

1000 end
1001 function pdftexcmds.pipe(cmdline, patch)
1002   local result
1003   systemexitstatus = nil
1004   texio.write_nl("log", "pipe(" .. cmdline .. ") ")
1005   if io.popen then
1006     texio.write("log", "executed.")
1007     local handle = io.popen(cmdline, "r")
1008     if handle then
1009       result = handle:read("*a")
1010       handle:close()
1011     end
1012   else
1013     texio.write("log", "disabled.")
1014   end
1015   if result then
1016     if patch == 1 then
1017       local temp = {}
1018       for i, a in ipairs(result) do
1019         if a == 0 then
1020           table.insert(temp, 1)
1021           table.insert(temp, 1)
1022         else
1023           if a == 1 then
1024             table.insert(temp, 1)
1025             table.insert(temp, 2)
1026           else
1027             table.insert(temp, a)
1028           end
1029         end
1030       end
1031       result = temp
1032     end
1033     tex.settoks(pdftexcmds.toks, result)
1034   else
1035     tex.settoks(pdftexcmds.toks, "")
1036   end
1037 end
1038  $\langle$ /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.

3.2 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
```

¹[CTAN:pkg/pdftexcmds](#)

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 → tex/generic/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`.

3.3 Refresh file name databases

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

3.4 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 pdf \TeX 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] Lua \TeX development team. *Lua \TeX 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 Lua \TeX and X_{\LaTeX}* . 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: `\pdfshellescape` 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 `\pdfshellescape`.

[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]

- `\pdfresettimer` and `\pdfelapsedtime` 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 Lua \TeX , 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 X \TeX

[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 `\filmmoddate` and other X \TeX commands.
- Adjust `\pdf@escapestring` in Lua \TeX to produce the same as in pdf \TeX 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).

[2020-06-24 v0.33]

- avoid that `\pdfelapsedtime` and `\pdfresettimer` are set to `\relax` when using `xelatex` (issue 5).
- load `l \TeX uatex` when using `plain` so that the catcode tables are available.

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>\@PackageWarningNoLine</code>	642
<code>\&</code>	500, 502	<code>\@ehc</code>	484
<code>\@PackageError</code>	480	<code>\@nil</code>	503, 504, 506,
<code>\@PackageInfoNoLine</code>			509, 513, 517, 525, 558, 567, 700
	289, 303, 308, 391, 395, 397	<code>\undefined</code>	58, 288, 383
<code>\@PackageWarning</code>	440	<code>\</code>	831, 834

	Numbers	
<code>\0</code>	532, 685	305, 310, 316, 324, 336, 351, 352, 355, 357, 383, 478, 488, 507, 518, 618, 626, 630, 638, 641
	A	
<code>\aftergroup</code>	29	<code>\immediate</code> 23, 52, 217 <code>\input</code> 137, 149
	C	L
<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, 500, 502, 532, 685	<code>\ltx@backslashchar</code>
<code>\catcodetable@string</code>	668	. 391, 395, 398, 643, 645, 647, 648
<code>\chardef</code>	182, 184	<code>\ltx@ccclv</code> 428, 430
<code>\count</code>	428, 430	<code>\ltx@firstoftwo</code> 345, 375, 410
<code>\csname</code>	14, 21, 50, 66, 76, 133, 136, 156, 157, 160, 162, 176, 194, 202, 204, 221, 222, 226, 227, 256, 261, 264, 265, 279, 281, 284, 305, 310, 316, 319, 325, 327, 336, 371, 488, 491, 618, 626, 630, 638	<code>\ltx@ifundefined</code> 178, 390
		<code>\ltx@one</code> 394, 409, 416, 417
		<code>\ltx@onelevel@sanitize</code> 470
		<code>\ltx@ReturnAfterElseFi</code> 508, 519
		<code>\ltx@ReturnAfterFi</code> 512, 524
		<code>\ltx@secondoftwo</code> .. 347, 377, 404, 412
		<code>\ltx@zero</code> 389, 403, 419, 520
		<code>\luaescapestring</code>
		. 536, 537, 543, 548, 556, 565, 571, 576, 581, 586, 591, 596, 597, 598, 603, 608, 613, 672, 698
		<code>\luatexversion</code>
	 234, 453, 494, 495, 530, 664
	D	M
<code>\delimiter</code>	344, 350, 368	<code>\mdfivesum</code> 207, 209
<code>\directlua</code>	235, 237, 535, 542, 547, 554, 563, 570, 575, 580, 585, 590, 595, 602, 607, 612, 667, 671, 676, 681	<code>\meaning</code> .. 259, 261, 319, 322, 338, 374
<code>\draftmode</code>	384	<code>\MessageBreak</code>
	 291, 441, 481, 482, 644, 646
	E	N
<code>\empty</code>	17, 18	<code>\newlinechar</code> 240, 241
<code>\endcsname</code>	14, 21, 50, 66, 76, 133, 136, 149, 157, 160, 162, 176, 194, 202, 204, 222, 226, 227, 256, 261, 264, 265, 279, 281, 284, 305, 310, 316, 319, 325, 327, 336, 371, 488, 491, 618, 626, 630, 638	<code>\number</code> 128, 596, 597, 668
<code>\endinput</code>	29, 129	<code>\numexpr</code> <u>629</u> , 657, 667
<code>\endlinechar</code>	4, 35, 71, 77, 89, 241	
<code>\escapechar</code>	128, 131, 260, 318	
<code>\everyjob</code>	460, 461	
	I	P
<code>\ifcase</code>	402, 435, 499	<code>\PackageInfo</code> 26
<code>\ifcsname</code>	149	<code>\pdf@draftmode</code> 5, <u>403</u> , <u>415</u>
<code>\ifeof</code>	180, 181	<code>\pdf@elapsedtime</code> 4, <u>656</u>
<code>\ifluatex</code>	148, 233, 277, 382, 446	<code>\pdf@escapehex</code> 4, 168, <u>541</u>
<code>\ifnum</code>	180, 234, 283, 286, 344, 374, 409, 416, 453, 454, 494, 495, 530, 664, 686	<code>\pdf@escapehexnative</code> 168, <u>546</u>
<code>\ifpdf</code>	393	<code>\pdf@escapename</code> <u>574</u>
<code>\ifx</code>	15, 18, 21, 50, 58, 61, 133, 136, 156, 157, 162, 176, 194, 202, 204, 221, 222, 227, 256, 263, 279, 281, 284,	<code>\pdf@escapenamenative</code> <u>579</u>
		<code>\pdf@escapestring</code> <u>569</u>
		<code>\pdf@filedump</code> 4, 197, <u>594</u>
		<code>\pdf@filemdfivesum</code> ... 4, 209, 214, <u>611</u>
		<code>\pdf@filemoddate</code> 4, <u>589</u>
		<code>\pdf@filesize</code> 4, <u>584</u>
		<code>\pdf@ifdraftmode</code> 5, <u>404</u> , <u>408</u>
		<code>\pdf@ifprimitive</code> 6, <u>273</u> , 309
		<code>\pdf@isprimitive</code>
	 6, 334, 337, 373, 387, 540
		<code>\pdf@lastsystemexit</code> <u>680</u>
		<code>\pdf@lastsystemstatus</code> <u>675</u>
		<code>\pdf@mdfivesum</code> 4, 207, 208, 212, 213, <u>601</u>
		<code>\pdf@mdfivesumnative</code> .. 208, 213, <u>606</u>
		<code>\pdf@pipe</code> 7, <u>686</u>
		<code>\pdf@primitive</code> .. 6, <u>269</u> , 288, 290, 304

<code>\pdf@resettimer</code>	4, 651	254, 269, 270, 271, 272, 273,
<code>\pdf@setdraftmode</code>	5, 426 , 441	274, 275, 276, 314, 332, 333,
<code>\pdf@shellescape</code>	5, 182 , 184 , 189 , 664	389, 394, 402, 616, 627, 639, 641
<code>\pdf@strcmp</code>	3, 374 , 534	<code>\pdftexcmds@toks</code> .. 487 , 553 , 562 , 695
<code>\pdf@system</code>	6, 216 , 670	<code>\pdftexrevision</code> .. 286
<code>\pdf@unescapehex</code>	4, 170 , 551	<code>\pdftexversion</code> .. 180 , 283
<code>\pdf@unescapehexnative</code>	7, 170 , 560	<code>\protected</code> .. 616 , 651 , 656 , 666
<code>\pdfdraftmode</code>	383 , 384 , 407	<code>\ProvidesPackage</code> .. 19 , 67
<code>\pdffiledump</code>	198	
<code>\pdfmdfivesum</code>	212 , 214	
<code>\pdfprimitive</code>	292	
<code>\pdfshellescape</code>	190	
<code>\pdftexcmds@isprimitive</code>	.. 341 , 343	
<code>\pdftexcmds@setdraftmode</code>	.. 430 , 434	
<code>\pdftexcmds@AtEnd</code> 95 , 96 , 126 , 127 , 448 , 703	
<code>\pdftexcmds@DecodeA</code> 504 , 506	
<code>\pdftexcmds@DecodeB</code> 509 , 517	
<code>\pdftexcmds@directlua</code> 234 , 242 , 450 , 455 , 462 , 472 , 619 , 631 , 652 , 658 , 687 , 696	
<code>\pdftexcmds@draftmode</code> 407 , 409 , 416 , 423	
<code>\pdftexcmds@equal</code> 344 , 350 , 368	
<code>\pdftexcmds@equalcont</code> 359 , 365 , 366 , 371	
<code>\pdftexcmds@isprimitive</code> 338 , 340	
<code>\pdftexcmds@nopdftex</code> 151 , 152 , 158 , 177 , 195 , 205 , 223	
<code>\pdftexcmds@Patch</code> 493 , 499 , 556 , 565 , 698	
<code>\pdftexcmds@PatchDecode</code> 503 , 552 , 561 , 694	
<code>\pdftexcmds@setdraftmode</code> 405 , 422 , 436 , 438	
<code>\pdftexcmds@strip@prefix</code> 253 , 259	
<code>\pdftexcmds@temp</code> 154 , 166 , 167 , 169 , 171 , 172 , 173 , 174 , 219 , 230 , 231 ,	
		R
		<code>\RequirePackage</code> .. 144 , 145 , 146
		<code>\romannumeral</code> .. 552 , 561 , 694
		S
		<code>\space</code> .. 290 , 292 , 304 , 309
		T
		<code>\the</code> .. 77 , 78 , 79 , 80 , 81 , 82 , 83 , 84 , 97 , 430 , 461 , 553 , 562 , 695
		<code>\TMP@EnsureCode</code> .. 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
		<code>\TMP@RequirePackage</code> .. 134 , 140 , 141 , 142
		<code>\toksdef</code> .. 489
		W
		<code>\write</code> .. 23 , 52 , 217
		X
		<code>\x</code> .. 14 , 15 , 18 , 22 , 26 , 28 , 51 , 56 , 66 , 75 , 87 , 258 , 259 , 263 , 319 , 324 , 429 , 432 , 469 , 470 , 478 , 482
		Y
		<code>\y</code> .. 261 , 263 , 320 , 322 , 324 , 471 , 478 , 483
		Z
		<code>\z</code> .. 321 , 322