

The pdf_{tex}cmds package

Heiko Oberdiek*

2019/07/25 v0.30

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	4
1.2.1	Strings	4
1.2.2	Files	4
1.2.3	Timekeeping	4
1.2.4	Miscellaneous	5
1.2.5	Additional macro: <code>\pdf@isprimitive</code>	6
1.2.6	Experimental	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>	12
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@isprimitive</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	27
2.11.4	Miscellaneous	27

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

3	Test	29
3.1	Catcode checks for loading	29
3.2	Test for <code>\pdf@isprimitive</code>	30
3.3	Test for <code>\pdf@shellescape</code>	31
3.4	Test for escape functions	32
4	Installation	35
4.1	Download	35
4.2	Bundle installation	35
4.3	Package installation	35
4.4	Refresh file name databases	36
4.5	Some details for the interested	36
5	History	36
	[2007/11/11 v0.1]	36
	[2007/11/12 v0.2]	36
	[2007/12/12 v0.3]	37
	[2009/04/10 v0.4]	37
	[2009/09/22 v0.5]	37
	[2009/09/23 v0.6]	37
	[2009/12/12 v0.7]	37
	[2010/03/01 v0.8]	37
	[2010/04/01 v0.9]	37
	[2010/11/04 v0.10]	37
	[2010/11/11 v0.11]	37
	[2011/01/30 v0.12]	37
	[2011/03/04 v0.13]	37
	[2011/04/10 v0.14]	37
	[2011/04/16 v0.15]	38
	[2011/04/22 v0.16]	38
	[2011/06/29 v0.17]	38
	[2011/07/01 v0.18]	38
	[2011/07/28 v0.19]	38
	[2011/11/29 v0.20]	38
	[2016/05/10 v0.21]	38
	[2016/05/21 v0.22]	38
	[2016/10/02 v0.23]	38
	[2017/01/29 v0.24]	38
	[2017/03/19 v0.25]	38
	[2018/01/21 v0.26]	38
	[2018/01/30 v0.27]	38
	[2018/09/07 v0.28]	39
	[2018/09/10 v0.29]	39
	[2019/07/25 v0.30]	39

1 Documentation

Some primitives of pdfTeX [pdf`tex`-manual] are not defined by LuaTeX [lua`tex`-manual]. This package implements macro based solutions using Lua code for the following missing pdfTeX primitives;

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

- `\pdfescapename`
- `\pdfescapestring`
- `\pdffilesize`
- `\pdffilemoddate`
- `\pdffiledump`
- `\pdfmdfivesum`
- `\pdfresettimer`
- `\pdfelapsedtime`
- `\immediate\write18`

The original names of the primitives cannot be used:

- The syntax for their arguments cannot easily 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\bggroup ...}`.)
- 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 [pdfTeX-manual]

`\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 [pdfTeX-manual]

`\pdf@filesize {<filename>}`

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

`\pdf@filemoddate {<filename>}`

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

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

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

`\pdf@mdfivesum {<string>}`

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

`\pdf@filemdfivesum {<filename>}`

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

1.2.3 Timekeeping [pdfTeX-manual]

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

`\pdf@resettimer`

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

`\pdf@elapsedtime`

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

- pdfTeX with `gettimeofday`: $\geq 1/65536$ s
- pdfTeX with `ftime`: ≥ 1 ms
- pdfTeX with `time`: ≥ 1 s
- LuaTeX: ≥ 10 ms
(`os.clock()` returns a float number with two decimal digits in LuaTeX beta-0.70.1-2011061416 (rev 4277)).

1.2.4 Miscellaneous [pdfTeX-manual]

`\pdf@draftmode`

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

`\pdf@ifdraftmode { $\langle true \rangle$ } { $\langle false \rangle$ }`

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{Ops,\string@@input\space\is\not\the\space\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 [2019/07/25 v0.30 Utility functions of pdfTeX for LuaTeX (HO)]%

```

2.2 Catcodes

```

69 \begingroup\catcode61\catcode48\catcode32=10\relax%
70 \catcode13=5 % ^~M
71 \endlinechar=13 %
72 \catcode123=1 % {
73 \catcode125=2 % }
74 \catcode64=11 % @
75 \def\x{\endgroup

```



```

76 \expandafter\edef\csname pdftexcmds@AtEnd\endcsname{%
77 \endlinechar=\the\endlinechar\relax
78 \catcode13=\the\catcode13\relax
79 \catcode32=\the\catcode32\relax
80 \catcode35=\the\catcode35\relax
81 \catcode61=\the\catcode61\relax
82 \catcode64=\the\catcode64\relax
83 \catcode123=\the\catcode123\relax
84 \catcode125=\the\catcode125\relax
85 }%
86 }%
87 \x\catcode61\catcode48\catcode32=10\relax%
88 \catcode13=5 % ^~M
89 \endlinechar=13 %
90 \catcode35=6 % #
91 \catcode64=11 % @
92 \catcode123=1 % {
93 \catcode125=2 % }
94 \def\TMP@EnsureCode#1#2{%
95 \edef\pdftexcmds@AtEnd{%
96 \pdftexcmds@AtEnd
97 \catcode#1=\the\catcode#1\relax
98 }%
99 \catcode#1=#2\relax
100 }
101 \TMP@EnsureCode{0}{12}%
102 \TMP@EnsureCode{1}{12}%
103 \TMP@EnsureCode{2}{12}%
104 \TMP@EnsureCode{10}{12}% ^~J
105 \TMP@EnsureCode{33}{12}% !
106 \TMP@EnsureCode{34}{12}% "
107 \TMP@EnsureCode{38}{4}% &
108 \TMP@EnsureCode{39}{12}% '
109 \TMP@EnsureCode{40}{12}% (
110 \TMP@EnsureCode{41}{12}% )
111 \TMP@EnsureCode{42}{12}% *
112 \TMP@EnsureCode{43}{12}% +
113 \TMP@EnsureCode{44}{12}% ,
114 \TMP@EnsureCode{45}{12}% -
115 \TMP@EnsureCode{46}{12}% .
116 \TMP@EnsureCode{47}{12}% /
117 \TMP@EnsureCode{58}{12}% :
118 \TMP@EnsureCode{60}{12}% <
119 \TMP@EnsureCode{62}{12}% >
120 \TMP@EnsureCode{91}{12}% [
121 \TMP@EnsureCode{93}{12}% ]
122 \TMP@EnsureCode{94}{7}% ^ (superscript)
123 \TMP@EnsureCode{95}{12}% _ (other)
124 \TMP@EnsureCode{96}{12}% `
125 \TMP@EnsureCode{126}{12}% ~ (other)
126 \edef\pdftexcmds@AtEnd{%
127 \pdftexcmds@AtEnd
128 \escapechar=\number\escapechar\relax
129 \noexpand\endinput
130 }
131 \escapechar=92 %

```

2.3 Load packages

```

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

```

2.4 Without LuaTeX

```

150 \ifluatex
151 \else
152   \@PackageInfoNoLine{pdftexcmds}{LuaTeX not detected}%
153   \def\pdftexcmds@nopdftex{%
154     \@PackageInfoNoLine{pdftexcmds}{pdfTeX >= 1.30 not detected}%
155     \let\pdftexcmds@nopdftex\relax
156   }%
157   \def\pdftexcmds@temp#1{%
158     \begingroup\expandafter\expandafter\expandafter\endgroup
159     \expandafter\ifx\csname pdf#1\endcsname\relax
160       \pdftexcmds@nopdftex
161     \else
162       \expandafter\def\csname pdf@#1\expandafter\endcsname
163       \expandafter##\expandafter{%
164         \csname pdf#1\endcsname
165       }%
166     \fi
167   }%
168   \pdftexcmds@temp{strcmp}%
169   \pdftexcmds@temp{escapehex}%
170   \let\pdf@escapehexnative\pdf@escapehex
171   \pdftexcmds@temp{unescapehex}%
172   \let\pdf@unescapehexnative\pdf@unescapehex
173   \pdftexcmds@temp{escapestring}%
174   \pdftexcmds@temp{escapename}%
175   \pdftexcmds@temp{filesize}%
176   \pdftexcmds@temp{filemoddate}%
177   \begingroup\expandafter\expandafter\expandafter\endgroup
178   \expandafter\ifx\csname pdfshellescape\endcsname\relax
179     \pdftexcmds@nopdftex
180     \ltx@ifundefined{pdftexversion}{%
181       }{%
182         \ifnum\pdftexversion>120 % 1.21a supports \ifeof18
183           \ifeof18 %
184             \chardef\pdf@shellescape=0 %
185           \else
186             \chardef\pdf@shellescape=1 %
187           \fi

```

```

188     \fi
189   }%
190 \else
191   \def\pdf@shellescape{%
192     \pdfshellescape
193   }%
194 \fi
195 \begingroup\expandafter\expandafter\expandafter\endgroup
196 \expandafter\ifx\csname pdffiledump\endcsname\relax
197   \pdfcmds@nopdftex
198 \else
199   \def\pdf@filedump#1#2#3{%
200     \pdffiledump offset#1 length#2{#3}%
201   }%
202 \fi

203 \begingroup\expandafter\expandafter\expandafter\endgroup
204 \expandafter\ifx\csname pdfmdfivesum\endcsname\relax
205   \begingroup\expandafter\expandafter\expandafter\endgroup
206   \expandafter\ifx\csname mdfivesum\endcsname\relax
207     \pdfcmds@nopdftex
208   \else
209     \def\pdf@mdfivesum#\mdfivesum{%
210       \let\pdf@mdfivesumnative\pdf@mdfivesum
211       \def\pdf@filemdfivesum#\mdfivesum file{%
212         \fi
213       \else
214         \def\pdf@mdfivesum#\pdfmdfivesum{%
215           \let\pdf@mdfivesumnative\pdf@mdfivesum
216           \def\pdf@filemdfivesum#\pdfmdfivesum file{%
217             \fi
218           \def\pdf@system#{%
219             \immediate\write18%
220           }%
221           \def\pdfcmds@temp#1{%
222             \begingroup\expandafter\expandafter\expandafter\endgroup
223             \expandafter\ifx\csname pdf#1\endcsname\relax
224               \pdfcmds@nopdftex
225             \else
226               \expandafter\let\csname pdf@#1\expandafter\endcsname
227               \csname pdf#1\endcsname
228             \fi
229           }%
230           \pdfcmds@temp{resettimer}%
231           \pdfcmds@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

\pdfdocmds@directlua

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

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

```
\pdfdocmds@strip@prefix

253 \def\pdfdocmds@strip@prefix#1>{}

254 \def\pdfdocmds@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\pdfdocmds@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\expandafter\endcsname
265 \csname #2\endcsname
266 \fi
267 \fi
268 }

\pdf@primitive

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

\pdf@ifprimitive

273 \pdfdocmds@temp{ifprimitive}{ifpdfprimitive}{ifpdfprimitive}% pdfTeX, oldLuaTeX
274 \pdfdocmds@temp{ifprimitive}{ifprimitive}{ifprimitive}% XeTeX, luatex
275 \pdfdocmds@temp{ifprimitive}{luatexifprimitive}{ifpdfprimitive}% oldLuaTeX
276 \pdfdocmds@temp{ifprimitive}{luatexifpdfprimitive}{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   \fi
300 \endgroup
301 \fi

```

2.5.3 Result

```

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

```

2.6 X_YTEX

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@cclv=#1\relax
429   \edef\x{\endgroup
430     \noexpand\pdftexcmds@@setdraftmode{\the\count\ltx@cclv}}%
431   }%
432   \x
433 }

```

\pdftexcmds@@setdraftmode

```

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

```

2.9 Load Lua module

```

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

450 \ifnum\luatexversion<80
451   \begingroup\expandafter\expandafter\expandafter\endgroup
452   \expandafter\ifx\csname RequirePackage\endcsname\relax
453     \def\TMP@RequirePackage#1[#2]{%
454       \begingroup\expandafter\expandafter\expandafter\endgroup
455       \expandafter\ifx\csname ver@#1.sty\endcsname\relax
456         \input #1.sty\relax
457       \fi
458     }%
459     \TMP@RequirePackage{luatex-loader}[2009/04/10]%
460   \else
461     \RequirePackage{luatex-loader}[2009/04/10]%
462   \fi
463 \fi
464 \pdftexcmds@directlua{%
465   require("pdftexcmds")%
466 }
467 \ifnum\luatexversion>37 %
468   \ifnum0%
469     \pdftexcmds@directlua{%

```



```

470         if status.ini_version then %
471             tex.write("1")%
472         end%
473     }>0 %
474     \everyjob\expandafter{%
475         \the\everyjob
476         \pdf texcmds@directlua{%
477             require("pdf texcmds")%
478         }%
479     }%
480 \fi
481 \fi
482 \begingroup
483 \def\x{2019/07/25 v0.30}%
484 \ltx@onelevel@sanitize\x
485 \edef\y{%
486     \pdf texcmds@directlua{%
487         if oberdiek.pdf texcmds.getversion then %
488             oberdiek.pdf texcmds.getversion()%
489         end%
490     }%
491 }%
492 \ifx\x\y
493 \else
494     \@PackageError{pdf texcmds}{%
495         Wrong version of lua module.\MessageBreak
496         Package version: \x\MessageBreak
497         Lua module: \y
498     }\@ehc
499 \fi
500 \endgroup

```

2.10 Lua functions

2.10.1 Helper macros

`\pdf texcmds@toks`

```

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

```

`\pdf texcmds@Patch`

```

507 \def\pdf texcmds@Patch{0}
508 \ifnum\lua texversion>40 %
509     \ifnum\lua texversion<66 %
510         \def\pdf texcmds@Patch{1}%
511     \fi
512 \fi

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

```

`\pdf texcmds@PatchDecode`

```

517 \def\pdf texcmds@PatchDecode#1\@nil{%

```

```

518 \pdftexcmds@DecodeA#1^^A^^A\@nil{}%
519 }%

\pdftexcmds@DecodeA

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

```

\pdftexcmds@DecodeB

```

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

543 \fi

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

```

2.10.2 Strings [pdfTeX-manual]

\pdf@strcmp

```

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

554 \pdf@isprimitive

```

\pdf@escapehex

```

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

```

\pdf@escapehexnative

```

560 \long\def\pdf@escapehexnative#1{%

```

```

561 \directlua0{%
562 oberdiek.pdf texcmds.escapehex("\luaescapestring{#1}")%
563 }%
564 }%

\pdf@unescapehex

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

\pdf@unescapehexnative

574 \def\pdf@unescapehexnative#1{%
575 & \romannumeral\expandafter\pdf texcmds@PatchDecode
576 \the\expandafter\pdf texcmds@toks
577 \directlua0{%
578 oberdiek.pdf texcmds.toks="pdf texcmds@toks"%
579 oberdiek.pdf texcmds.unescapehex("\luaescapestring{#1}", \pdf texcmds@Patch)%
580 }%
581 & \@nil
582 }%

\pdf@escapestring

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

\pdf@escapename

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

\pdf@escapenamenative

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

```

2.10.3 Files [pdf tex-manual]

```

\pdf@filesize

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

```

\pdf@filemoddate

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

\pdf@filedump

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

\pdf@mdfivesum

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

\pdf@mdfivesumnative

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

\pdf@filemdfivesum

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

2.10.4 Timekeeping [pdfTeX-manual]

\protected

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

\numexpr

```
643 \begingroup\expandafter\expandafter\expandafter\endgroup
644 \expandafter\ifx\csname numexpr\endcsname\relax
```

```

645 \pdftexcmds@directlua{%
646   if tex.enableprimitives then %
647     tex.enableprimitives(' ', {'numexpr'})%
648   end%
649 }%
650 \fi
651 \begingroup\expandafter\expandafter\expandafter\endgroup
652 \expandafter\ifx\csname numexpr\endcsname\relax
653   \let\pdftexcmds@temp=N%
654 \fi

655 \ifx\pdftexcmds@temp N%
656   \@PackageWarningNoLine{pdftexcmds}{%
657     Definitions of \ltx@backslashchar pdf@resettimer and%
658     \MessageBreak
659     \ltx@backslashchar pdf@elapsedtime are skipped, because%
660     \MessageBreak
661     e-TeX's \ltx@backslashchar protected or %
662     \ltx@backslashchar numexpr are missing%
663   }%
664 \else

\pdf@resettimer

665 \protected\def\pdf@resettimer{%
666   \pdftexcmds@directlua{%
667     oberdiek.pdftexcmds.resettimer()%
668   }%
669 }%

\pdf@elapsedtime

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

677 \fi

```

2.10.5 Shell escape

```

\pdf@shellescape

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

\pdf@system

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

```

```

\pdf@lastsystemstatus
689 \def\pdf@lastsystemstatus{%
690   \directlua0{%
691     oberdiek.pdfdoccmds.lastsystemstatus()%
692   }%
693 }

\pdf@lastsystemexit
694 \def\pdf@lastsystemexit{%
695   \directlua0{%
696     oberdiek.pdfdoccmds.lastsystemexit()%
697   }%
698 }

699 \catcode`\0=12 %

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

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

```

2.11 Lua module

```

719 (*lua)
720 oberdiek = oberdiek or {}
721 local pdfdoccmds = oberdiek.pdfdoccmds or {}
722 oberdiek.pdfdoccmds = pdfdoccmds
723 local systemexitstatus
724 function pdfdoccmds.getversion()
725   tex.write("2019/07/25 v0.30")
726 end

```

2.11.1 Strings [pdfdoccmds-manual]

```

727 function pdfdoccmds.strcmp(A, B)
728   if A == B then
729     tex.write("0")
730   elseif A < B then
731     tex.write("-1")
732   else
733     tex.write("1")
734   end

```

```

735 end
736 local function utf8_to_byte(str)
737   local i = 0
738   local n = string.len(str)
739   local t = {}
740   while i < n do
741     i = i + 1
742     local a = string.byte(str, i)
743     if a < 128 then
744       table.insert(t, string.char(a))
745     else
746       if a >= 192 and i < n then
747         i = i + 1
748         local b = string.byte(str, i)
749         if b < 128 or b >= 192 then
750           i = i - 1
751         elseif a == 194 then
752           table.insert(t, string.char(b))
753         elseif a == 195 then
754           table.insert(t, string.char(b + 64))
755         end
756       end
757     end
758   end
759   return table.concat(t)
760 end
761 function pdftexcmds.escapex(str, mode)
762   if mode == "byte" then
763     str = utf8_to_byte(str)
764   end
765   tex.write((string.gsub(str, ".",
766     function (ch)
767       return string.format("%02X", string.byte(ch))
768     end
769   )))
770 end

```

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

```

771 function pdftexcmds.unescapehex(str, mode, patch)
772   local a = 0
773   local first = true
774   local result = {}
775   for i = 1, string.len(str), 1 do
776     local ch = string.byte(str, i)
777     if ch >= 48 and ch <= 57 then
778       ch = ch - 48
779     elseif ch >= 65 and ch <= 70 then
780       ch = ch - 55
781     elseif ch >= 97 and ch <= 102 then
782       ch = ch - 87
783     else
784       ch = nil
785     end
786     if ch then
787       if first then
788         a = ch * 16
789         first = false
790       else

```

```

791         table.insert(result, a + ch)
792         first = true
793     end
794 end
795 end
796 if not first then
797     table.insert(result, a)
798 end
799 if patch == 1 then
800     local temp = {}
801     for i, a in ipairs(result) do
802         if a == 0 then
803             table.insert(temp, 1)
804             table.insert(temp, 1)
805         else
806             if a == 1 then
807                 table.insert(temp, 1)
808                 table.insert(temp, 2)
809             else
810                 table.insert(temp, a)
811             end
812         end
813     end
814     result = temp
815 end
816 if mode == "byte" then
817     local utf8 = {}
818     for i, a in ipairs(result) do
819         if a < 128 then
820             table.insert(utf8, a)
821         else
822             if a < 192 then
823                 table.insert(utf8, 194)
824                 a = a - 128
825             else
826                 table.insert(utf8, 195)
827                 a = a - 192
828             end
829             table.insert(utf8, a + 128)
830         end
831     end
832     result = utf8
833 end

```

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

```

834 local unpack = _G["unpack"] or table.unpack
835 tex.settoks(pdftecmds.toks, string.char(unpack(result)))
836 end

```

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

```

837 function pdftecmds.escapestring(str, mode)
838     if mode == "byte" then
839         str = utf8_to_byte(str)
840     end
841     tex.write((string.gsub(str, ".",
842         function (ch)
843             local b = string.byte(ch)
844             if b < 33 or b > 126 then

```



```

845         return string.format("\\%.3o", b)
846     end
847     if b == 40 or b == 41 or b == 92 then
848         return "\\" .. ch
849     end

```

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

```

850         return nil
851     end
852 )))
853 end

```

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

```

854 function pdftexcmds.escapename(str, mode)
855     if mode == "byte" then
856         str = utf8_to_byte(str)
857     end
858     tex.write((string.gsub(str, ".",
859         function (ch)
860             local b = string.byte(ch)
861             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.

```

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

```

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

```

870         return nil
871     end
872 end
873 )))
874 end

```

2.11.2 Files [pdftex-manual]

```

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

```

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

```

884 function pdftexcmds.filemoddate(filename)
885     local foundfile = kpse.find_file(filename, "tex", true)
886     if foundfile then
887         local date = lfs.attributes(foundfile, "modification")
888         if date then
889             local d = os.date("*t", date)
890             if d.sec >= 60 then
891                 d.sec = 59
892             end

```

```

893     local u = os.date("!*t", date)
894     local off = 60 * (d.hour - u.hour) + d.min - u.min
895     if d.year ~= u.year then
896         if d.year > u.year then
897             off = off + 1440
898         else
899             off = off - 1440
900         end
901     elseif d.yday ~= u.yday then
902         if d.yday > u.yday then
903             off = off + 1440
904         else
905             off = off - 1440
906         end
907     end
908     local timezone
909     if off == 0 then
910         timezone = "Z"
911     else
912         local hours = math.floor(off / 60)
913         local mins = math.abs(off - hours * 60)
914         timezone = string.format("%+03d'%02d'", hours, mins)
915     end
916     tex.write(string.format("D:%04d%02d%02d%02d%02d%s",
917         d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
918     end
919 end
920 end
921 function pdftexcmds.filedump(offset, length, filename)
922     length = tonumber(length)
923     if length and length > 0 then
924         local foundfile = kpse.find_file(filename, "tex", true)
925         if foundfile then
926             offset = tonumber(offset)
927             if not offset then
928                 offset = 0
929             end
930             local filehandle = io.open(foundfile, "rb")
931             if filehandle then
932                 if offset > 0 then
933                     filehandle:seek("set", offset)
934                 end
935                 local dump = filehandle:read(length)
936                 pdftexcmds.escapehex(dump)
937                 filehandle:close()
938             end
939         end
940     end
941 end
942 function pdftexcmds.md5sum(str, mode)
943     if mode == "byte" then
944         str = utf8_to_byte(str)
945     end
946     pdftexcmds.escapehex(md5.sum(str))
947 end
948 function pdftexcmds.filemd5sum(filename)
949     local foundfile = kpse.find_file(filename, "tex", true)
950     if foundfile then

```

```

951     local filehandle = io.open(foundfile, "rb")
952     if filehandle then
953         local contents = filehandle:read("*a")
954         pdftexcmds.escapehex(md5.sum(contents))
955         filehandle:close()
956     end
957 end
958 end

```

2.11.3 Timekeeping [pdftex-manual]

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

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

```

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

```

2.11.4 Miscellaneous [pdftex-manual]

```

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

```

```

995  if os.execute then
996      texio.write("log", "executed.")
997      systemexitstatus = os.execute(cmdline)
998  else
999      texio.write("log", "disabled.")
1000  end
1001 end
1002 function pdftexcmds.lastsystemstatus()
1003     local result = tonumber(systemexitstatus)
1004     if result then
1005         local x = math.floor(result / 256)
1006         tex.write(result - 256 * math.floor(result / 256))
1007     end
1008 end
1009 function pdftexcmds.lastsystemexit()
1010     local result = tonumber(systemexitstatus)
1011     if result then
1012         tex.write(math.floor(result / 256))
1013     end
1014 end
1015 function pdftexcmds.pipe(cmdline, patch)
1016     local result
1017     systemexitstatus = nil
1018     texio.write_nl("log", "pipe(" .. cmdline .. ") ")
1019     if io.popen then
1020         texio.write("log", "executed.")
1021         local handle = io.popen(cmdline, "r")
1022         if handle then
1023             result = handle:read("*a")
1024             handle:close()
1025         end
1026     else
1027         texio.write("log", "disabled.")
1028     end
1029     if result then
1030         if patch == 1 then
1031             local temp = {}
1032             for i, a in ipairs(result) do
1033                 if a == 0 then
1034                     table.insert(temp, 1)
1035                     table.insert(temp, 1)
1036                 else
1037                     if a == 1 then
1038                         table.insert(temp, 1)
1039                         table.insert(temp, 2)
1040                     else
1041                         table.insert(temp, a)
1042                     end
1043                 end
1044             end
1045             result = temp
1046         end
1047         tex.settoks(pdftexcmds.toks, result)
1048     else
1049         tex.settoks(pdftexcmds.toks, "")
1050     end
1051 end
1052 

```

3 Test

3.1 Catcode checks for loading

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

```

1107 \count@=#1\relax
1108 \loop
1109 \ifnum#3=\catcode\count@
1110 \else
1111 \errmessage{%
1112 Character \the\count@\space
1113 with wrong catcode \the\catcode\count@\space
1114 instead of \number#3%
1115 }%
1116 \fi
1117 \ifnum\count@<#2\relax
1118 \advance\count@ 1 %
1119 \repeat
1120 }
1121 \def\space{ }
1122 \expandafter\ifx\csname LoadCommand\endcsname\relax
1123 \def\LoadCommand{\input pdftexcmds.sty\relax}%
1124 \fi
1125 \def\Test{%
1126 \RangeCatcodeInvalid{0}{47}%
1127 \RangeCatcodeInvalid{58}{64}%
1128 \RangeCatcodeInvalid{91}{96}%
1129 \RangeCatcodeInvalid{123}{255}%
1130 \catcode`\@=12 %
1131 \catcode`\=0 %
1132 \catcode`\%=14 %
1133 \LoadCommand
1134 \RangeCatcodeCheck{0}{36}{15}%
1135 \RangeCatcodeCheck{37}{37}{14}%
1136 \RangeCatcodeCheck{38}{47}{15}%
1137 \RangeCatcodeCheck{48}{57}{12}%
1138 \RangeCatcodeCheck{58}{63}{15}%
1139 \RangeCatcodeCheck{64}{64}{12}%
1140 \RangeCatcodeCheck{65}{90}{11}%
1141 \RangeCatcodeCheck{91}{91}{15}%
1142 \RangeCatcodeCheck{92}{92}{0}%
1143 \RangeCatcodeCheck{93}{96}{15}%
1144 \RangeCatcodeCheck{97}{122}{11}%
1145 \RangeCatcodeCheck{123}{255}{15}%
1146 \RestoreCatcodes
1147 }
1148 \Test
1149 \csname @@end\endcsname
1150 \end
1151 </test1>

```

3.2 Test for \pdf@isprimitive

```

1152 <*test2>
1153 \catcode`\{=1 %
1154 \catcode`\}=2 %
1155 \catcode`\#=6 %
1156 \catcode`\@=11 %
1157 \input pdftexcmds.sty\relax
1158 \def\msg#1{%
1159 \begingroup
1160 \escapechar=92 %
1161 \immediate\write16{#1}%
1162 \endgroup

```

```

1163 }
1164 \long\def\test#1#2#3#4{%
1165   \begingroup
1166     #4%
1167     \def\str{%
1168       Test \string\pdf@isprimitive
1169       {\string #1}{\string #2}{...}: %
1170     }%
1171     \pdf@isprimitive{#1}{#2}{%
1172       \ifx#3Y%
1173         \msg{\str true ==> OK.}%
1174       \else
1175         \errmessage{\str false ==> FAILED}%
1176       \fi
1177     }{%
1178       \ifx#3Y%
1179         \errmessage{\str true ==> FAILED}%
1180       \else
1181         \msg{\str false ==> OK.}%
1182       \fi
1183     }%
1184   \endgroup
1185 }
1186 \test\relax\relax Y{}
1187 \test\foobar\relax Y{\let\foobar\relax}
1188 \test\foobar\relax N{}
1189 \test\hbox\hbox Y{}
1190 \test\foobar@hbox\hbox Y{\let\foobar@hbox\hbox}
1191 \test\if\if Y{}
1192 \test\if\ifx N{}
1193 \test\ifx\if N{}
1194 \test\par\par Y{}
1195 \test\hbox\par N{}
1196 \test\par\hbox N{}
1197 \csname @@end\endcsname\end
1198 /test2}

```

3.3 Test for \pdf@shellescape

```

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

```

```

1219 \fi
1220 \MaybeEnd
1221 \ifx\pdf@shellescape\@undefined
1222   \msg{SHELL=U}%
1223 \else
1224   \msg{SHELL=\number\pdf@shellescape}%
1225 \fi
1226 \ifx\expected\@undefined
1227 \else
1228   \ifx\expected\relax
1229     \msg{EXPECTED=U}%
1230     \ifx\pdf@shellescape\@undefined
1231       \msg{OK}%
1232     \else
1233       \errmessage{Failed}%
1234     \fi
1235   \else
1236     \msg{EXPECTED=\number\expected}%
1237     \ifnum\pdf@shellescape=\expected\relax
1238       \msg{OK}%
1239     \else
1240       \errmessage{Failed}%
1241     \fi
1242   \fi
1243 \fi
1244 \csname @@end\endcsname\end
1245 \test-shell

```

3.4 Test for escape functions

```

1246 \test-escape
1247 \catcode`\{=1 %
1248 \catcode`\}=2 %
1249 \catcode`\#=6 %
1250 \catcode`\~=7 %
1251 \catcode`\@=11 %
1252 \errorcontextlines=1000 %
1253 \input pdftexcmds.sty\relax
1254 \def\msg#1{%
1255   \begingroup
1256     \escapechar=92 %
1257     \immediate\write16{#1}%
1258   \endgroup
1259 }
1260 \begingroup
1261   \catcode`\@=11 %
1262   \countdef\count@=255 %
1263   \def\space{ }%
1264   \long\def\@whilenum#1\do #2{%
1265     \ifnum #1\relax
1266       #2\relax
1267     \@iwhilenum{#1\relax#2\relax}%
1268   \fi
1269 }%
1270 \long\def\@iwhilenum#1{%
1271   \ifnum #1%
1272     \expandafter\@iwhilenum
1273   \else
1274     \expandafter\ltx@gobble

```



```

1275     \fi
1276     {#1}%
1277 }%
1278 \gdef\AllBytes{%
1279 \count@=0 %
1280 \catcode0=12 %
1281 \@whilenum\count@<256 \do{%
1282     \lccode0=\count@
1283     \ifnum\count@=32 %
1284         \xdef\AllBytes{\AllBytes\space}%
1285     \else
1286         \lowercase{%
1287             \xdef\AllBytes{\AllBytes^^@}%
1288         }%
1289     \fi
1290     \advance\count@ by 1 %
1291 }%
1292 \endgroup

1293 \def\AllBytesHex{%
1294 000102030405060708090A0B0C0D0E0F%
1295 101112131415161718191A1B1C1D1E1F%
1296 202122232425262728292A2B2C2D2E2F%
1297 303132333435363738393A3B3C3D3E3F%
1298 404142434445464748494A4B4C4D4E4F%
1299 505152535455565758595A5B5C5D5E5F%
1300 606162636465666768696A6B6C6D6E6F%
1301 707172737475767778797A7B7C7D7E7F%
1302 808182838485868788898A8B8C8D8E8F%
1303 909192939495969798999A9B9C9D9E9F%
1304 A0A1A2A3A4A5A6A7A8A9AABACADAFAF%
1305 B0B1B2B3B4B5B6B7B8B9BABBBBCDBEBF%
1306 C0C1C2C3C4C5C6C7C8C9CACBCCDCECF%
1307 D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
1308 E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
1309 F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF%
1310 }
1311 \ltx@onelevel@sanitize\AllBytesHex
1312 \expandafter\lowercase\expandafter{%
1313 \expandafter\def\expandafter\AllBytesHexLC
1314     \expandafter{\AllBytesHex}%
1315 }
1316 \begingroup
1317 \catcode`\#=12 %
1318 \xdef\AllBytesName{%
1319     #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
1320     #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
1321     #20!"#23$#25&'#28#29*+,-.#2F%
1322     0123456789:;#3C=#3E?%
1323     @ABCDEFGHIJKLMNO%
1324     PQRSTUVWXYZ#5B\ltx@backslashchar#5D^_%
1325     `abcdefghijklmnopqrstuvwxyz%
1326     pqrstuvwxyz#7B|#7D\string~#7F%
1327     #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
1328     #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
1329     #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
1330     #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
1331     #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
1332     #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%

```

```

1333   #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
1334   #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
1335   }%
1336 \endgroup
1337 \ltx@onelevel@sanitize\AllBytesName
1338 \edef\AllBytesFromName{\expandafter\ltx@gobble\AllBytes}
1339 \begingroup
1340   \def\{|}%
1341   \edef%\{\ltx@percentchar}%
1342   \catcode`\|=0 %
1343   \catcode`\#=12 %
1344   \catcode`\~=12 %
1345   \catcode`\=12 %
1346   \xdef\AllBytesString{%
1347     \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
1348     \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
1349     \040!"#$%&'(\)*+,-./%
1350     0123456789:;<=>?%
1351     @ABCDEFGHIJKLMNO%
1352     PQRSTUVWXYZ[\]^_%
1353     `abcdefghijklmnopqrstuvwxyz%
1354     pqrstuvwxyz{|}~\177%
1355     \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
1356     \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
1357     \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
1358     \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
1359     \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
1360     \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
1361     \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
1362     \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
1363   }%
1364 \endgroup
1365 \ltx@onelevel@sanitize\AllBytesString
1366 \def\Test#1#2#3{%
1367   \begingroup
1368     \expandafter\expandafter\expandafter\def
1369     \expandafter\expandafter\expandafter\TestResult
1370     \expandafter\expandafter\expandafter{%
1371       #1#2}%
1372   }%
1373   \ifx\TestResult#3%
1374   \else
1375     \newlinechar=10 %
1376     \msg{Expect:^^J#3}%
1377     \msg{Result:^^J\TestResult}%
1378     \errmessage{\string#2 -\string#1-> \string#3}%
1379   \fi
1380 \endgroup
1381 }
1382 \def\test#1#2#3{%
1383   \edef\TestFrom{#2}%
1384   \edef\TestExpect{#3}%
1385   \ltx@onelevel@sanitize\TestExpect
1386   \Test#1\TestFrom\TestExpect
1387 }
1388 \test\pdf@unescapehex{74657374}{test}
1389 \begingroup
1390   \catcode0=12 %

```

```

1391 \catcode1=12 %
1392 \test\pdf@unescapehex{740074017400740174}{t^^@t^^At^^@t^^At}%
1393 \endgroup
1394 \Test\pdf@escapehex\AllBytes\AllBytesHex
1395 \Test\pdf@unescapehex\AllBytesHex\AllBytes
1396 \Test\pdf@escapename\AllBytes\AllBytesName
1397 \Test\pdf@escapestring\AllBytes\AllBytesString
1398 \csname @@end\endcsname\end
1399 </test-escape>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

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

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

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

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

4.2 Bundle installation

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

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

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

```

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

```

4.3 Package installation

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

```
tex pdftexcmds.dtx
```

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):

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

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

4.4 Refresh file name databases

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

4.5 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_p\pdftexcmds.dtx
bibtex_p\pdftexcmds.aux
makeindex_p-s_gind.ist_p\pdftexcmds.idx
pdflatex_p\pdftexcmds.dtx
makeindex_p-s_gind.ist_p\pdftexcmds.idx
pdflatex_p\pdftexcmds.dtx
```

5 History

[2007/11/11 v0.1]

- First version.

[2007/11/12 v0.2]

- Short description fixed.

[2007/12/12 v0.3]

- Organization of Lua code as module.

[2009/04/10 v0.4]

- Adaptation for syntax change of `\directlua` in 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 XeTeX

[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

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			
\#	.. 1056, 1155, 1202, 1249, 1317, 1343	\AllBytesFromName 1338
\% 1132, 1341	\AllBytesHex
\& 514, 516 1293, 1311, 1314, 1394, 1395	
\(..... 1349	\AllBytesHexLC 1313
\) 1349	\AllBytesName 1318, 1337, 1396
\@	.. 1057, 1130, 1156, 1203, 1251, 1261	\AllBytesString 1365, 1397
\@PackageError 494	B	
\@PackageInfoNoLine 152,	\body 1074, 1078
..... 154, 289, 303, 308, 391, 395, 397		C	
\@PackageWarning 440	\catcode 2,
\@PackageWarningNoLine 656 3, 5, 6, 7, 8, 9, 10, 11, 12, 13,	
\@ehc 498 33, 34, 36, 37, 38, 39, 40, 41, 42,	
\@firstofone 1065, 1068 43, 44, 45, 46, 47, 48, 49, 69, 70,	
\@gobble 1062, 1070 72, 73, 74, 78, 79, 80, 81, 82, 83,	
\@iwhilenum 1267, 1270, 1272 84, 87, 88, 90, 91, 92, 93, 97, 99,	
\@nil 517, 518, 520, 514, 516, 546, 699, 1054, 1055,	
..... 523, 527, 531, 539, 572, 581, 714	 1056, 1057, 1092, 1101, 1109,	
\@undefined 58, 1113, 1130, 1131, 1132, 1153,	
..... 288, 383, 1210, 1221, 1226, 1230	 1154, 1155, 1156, 1200, 1201,	
\@whilenum 1264, 1281 1202, 1203, 1247, 1248, 1249,	
\\ 845, 848, 1131, 1345, 1352 1250, 1251, 1261, 1280, 1317,	
\{ 1054, 1153, 1200, 1247 1342, 1343, 1344, 1345, 1390, 1391	
\} 1055, 1154, 1201, 1248	\catcodetable@string 682
\^ 1250	\chardef 184, 186
\ 1340, 1342	\count 428, 430
\~ 1344	\count@	1059, 1088, 1092, 1094, 1095,
Numbers	 1099, 1101, 1102, 1103, 1107,	
\0 546, 699, 1347, 1348, 1349 1109, 1112, 1113, 1117, 1118,	
\1 1354 1262, 1279, 1281, 1282, 1283, 1290	
\2 1355, 1356, 1357, 1358	\countdef 1059, 1262
\3 1359, 1360, 1361, 1362	\csname 14, 21,
A	 50, 66, 76, 133, 136, 159, 162,	
\advance 1095, 1103, 1118, 1290 164, 178, 196, 204, 206, 223,	
\aftergroup 29 226, 227, 256, 261, 264, 265,	
\AllBytes 1278, 1284, 279, 281, 284, 305, 310, 316,	
..... 1287, 1338, 1394, 1395, 1396, 1397	 319, 325, 327, 336, 371, 452,	
	 455, 502, 505, 632, 640, 644,	

652, 1058, 1061, 1064, 1067,
1122, 1149, 1197, 1217, 1244, 1398

D

`\delimiter` 344, 350, 368
`\directlua` 235,
237, 549, 556, 561, 568, 577,
584, 589, 594, 599, 604, 609,
616, 621, 626, 681, 685, 690, 695
`\do` 1264, 1281
`\draftmode` 384

E

`\empty` 17, 18
`\end` 1150, 1197, 1217, 1244, 1398
`\endcsname` 14, 21,
50, 66, 76, 133, 136, 159, 162,
164, 178, 196, 204, 206, 223,
226, 227, 256, 261, 264, 265,
279, 281, 284, 305, 310, 316,
319, 325, 327, 336, 371, 452,
455, 502, 505, 632, 640, 644,
652, 1058, 1061, 1064, 1067,
1122, 1149, 1197, 1217, 1244, 1398
`\endinput` 29, 129
`\endlinechar` ... 4, 35, 71, 77, 89, 241
`\errmessage` 1111,
1175, 1179, 1215, 1233, 1240, 1378
`\errorcontextlines` 1252
`\escapechar`
... 128, 131, 260, 318, 1160, 1256
`\everyjob` 474, 475
`\expected` 1226, 1228, 1236, 1237

F

`\foobar` 1187, 1188
`\foobar@hbox` 1190

G

`\gdef` 1278

H

`\hbox` 1189, 1190, 1195, 1196

I

`\if` 1191, 1192, 1193
`\ifcase` 402, 435, 513
`\ifeof` 182, 183
`\ifluatex` 150, 233, 277, 382, 446
`\ifnum` 182,
234, 283, 286, 344, 374, 409,
416, 450, 467, 468, 508, 509,
544, 678, 700, 1094, 1102, 1109,
1117, 1209, 1237, 1265, 1271, 1283
`\ifpdf` 393
`\ifx` 15, 18, 21,
50, 58, 61, 133, 136, 159, 178,
196, 204, 206, 223, 256, 263,
279, 281, 284, 305, 310, 316,
324, 336, 351, 352, 355, 357,
383, 452, 455, 492, 502, 521,
532, 632, 640, 644, 652, 655,
1058, 1061, 1064, 1067, 1122,
1172, 1178, 1192, 1193, 1207,
1210, 1221, 1226, 1228, 1230, 1373
`\immediate` 23, 52, 219, 1161, 1205, 1257
`\input` . 137, 456, 1123, 1157, 1204, 1253
`\iterate` 1075, 1077, 1079

L

`\lccode` 1282
`\LoadCommand` 1123, 1133
`\loop` 1073, 1089, 1100, 1108
`\lowercase` 1286, 1312
`\ltx@backslashchar` 391,
395, 398, 657, 659, 661, 662, 1324
`\ltx@ccclv` 428, 430
`\ltx@firstoftwo` 345, 375, 410
`\ltx@gobble` 1274, 1338
`\ltx@ifUndefined` 180, 390
`\ltx@one` 394, 409, 416, 417
`\ltx@onelevel@sanitize`
..... 484, 1311, 1337, 1365, 1385
`\ltx@percentchar` 1341
`\ltx@ReturnAfterElseFi` 522, 533
`\ltx@ReturnAfterFi` 526, 538
`\ltx@secondoftwo` .. 347, 377, 404, 412
`\ltx@zero` 389, 403, 419, 534
`\luaescapestring`
. 550, 551, 557, 562, 570, 579,
585, 590, 595, 600, 605, 610,
611, 612, 617, 622, 627, 686, 712
`\luatexversion` 234, 450,
467, 508, 509, 544, 678, 1207, 1209

M

`\MaybeEnd` 1206, 1217, 1220
`\mdfivesum` 209, 211
`\meaning` .. 259, 261, 319, 322, 338, 374
`\MessageBreak`
..... 291, 441, 495, 496, 658, 660
`\msg` . 1158, 1173, 1181, 1205, 1211,
1212, 1214, 1222, 1224, 1229,
1231, 1236, 1238, 1254, 1376, 1377

N

`\newlinechar` 240, 241, 1375
`\next` 1079, 1081, 1083
`\number`
128, 610, 611, 682, 1114, 1224, 1236
`\numexpr` 643, 671, 681

P

`\PackageInfo` 26
`\par` 1194, 1195, 1196
`\pdf@draftmode` 5, 403, 415
`\pdf@elapseddtime` 5, 670
`\pdf@escapehex` 4, 170, 555, 1394

`\pdf@escapehexnative` 170, 560
`\pdf@escapename` 588, 1396
`\pdf@escapenamenative` 593
`\pdf@escapestring` 583, 1397
`\pdf@filedump` 4, 199, 608
`\pdf@filemdfivesum` .. 4, 211, 216, 625
`\pdf@filemoddate` 4, 603
`\pdf@filesize` 4, 598
`\pdf@ifdraftmode` 5, 404, 408
`\pdf@ifprimitive` 6, 273, 309
`\pdf@isprimitive` 6,
334, 337, 373, 387, 554, 1168, 1171
`\pdf@lastsystemexit` 694
`\pdf@lastsystemstatus` 689
`\pdf@mdfivesum` 4, 209, 210, 214, 215, 615
`\pdf@mdfivesumnative` .. 210, 215, 620
`\pdf@pipe` 7, 700
`\pdf@primitive` ... 6, 269, 288, 290, 304
`\pdf@resettimer` 5, 665
`\pdf@setdraftmode` 5, 426, 441
`\pdf@shellescape` . 5, 184, 186, 191,
678, 1210, 1221, 1224, 1230, 1237
`\pdf@strcmp` 4, 374, 548
`\pdf@system` 6, 218, 684
`\pdf@unescapehex`
.... 4, 172, 565, 1388, 1392, 1395
`\pdf@unescapehexnative` ... 7, 172, 574
`\pdf@draftmode` 383, 384, 407
`\pdf@filedump` 200
`\pdf@mdfivesum` 214, 216
`\pdf@primitive` 292
`\pdf@shellescape` 192
`\pdf@texcmds@isprimitive` ... 341, 343
`\pdf@texcmds@setdraftmode` .. 430, 434
`\pdf@texcmds@AtEnd`
..... 95, 96, 126, 127, 448, 717
`\pdf@texcmds@DecodeA` 518, 520
`\pdf@texcmds@DecodeB` 523, 531
`\pdf@texcmds@directlua`
..... 234, 242, 464, 469, 476,
486, 633, 645, 666, 672, 701, 710
`\pdf@texcmds@draftmode`
..... 407, 409, 416, 423
`\pdf@texcmds@equal` 344, 350, 368
`\pdf@texcmds@equalcont`
..... 359, 365, 366, 371
`\pdf@texcmds@isprimitive` ... 338, 340
`\pdf@texcmds@nopdfTeX`
. 153, 155, 160, 179, 197, 207, 224
`\pdf@texcmds@Patch`
..... 507, 513, 570, 579, 712
`\pdf@texcmds@PatchDecode`
..... 517, 566, 575, 708
`\pdf@texcmds@setdraftmode`
..... 405, 422, 436, 438
`\pdf@texcmds@strip@prefix` ... 253, 259
`\pdf@texcmds@temp`
..... 157, 168, 169, 171, 173,
174, 175, 176, 221, 230, 231,
254, 269, 270, 271, 272, 273,
274, 275, 276, 314, 332, 333,
389, 394, 402, 630, 641, 653, 655
`\pdf@texcmds@toks` .. 501, 567, 576, 709
`\pdf@texrevision` 286
`\pdf@texversion` 182, 283
`\protected` 630, 665, 670, 680
`\ProvidesPackage` 19, 67

R

`\RangeCatcodeCheck` ... 1106, 1134,
1135, 1136, 1137, 1138, 1139,
1140, 1141, 1142, 1143, 1144, 1145
`\RangeCatcodeInvalid`
.... 1098, 1126, 1127, 1128, 1129
`\repeat` ... 1073, 1085, 1096, 1104, 1119
`\RequirePackage` 145, 146, 147, 148, 461
`\RestoreCatcodes` 1087, 1090, 1091, 1146
`\romannumeral` 566, 575, 708

S

`\space` 290, 292, 304,
309, 1112, 1113, 1121, 1263, 1284
`\str` 1167, 1173, 1175, 1179, 1181

T

`\Test` 1125, 1148,
1366, 1386, 1394, 1395, 1396, 1397
`\test` 1164, 1186, 1187, 1188,
1189, 1190, 1191, 1192, 1193,
1194, 1195, 1196, 1382, 1388, 1392
`\TestExpect` 1384, 1385, 1386
`\TestFrom` 1383, 1386
`\TestResult` 1369, 1373, 1377
`\the` 77, 78,
79, 80, 81, 82, 83, 84, 97, 430,
475, 567, 576, 709, 1092, 1112, 1113
`\TMP@EnsureCode` 94,
101, 102, 103, 104, 105, 106,
107, 108, 109, 110, 111, 112,
113, 114, 115, 116, 117, 118,
119, 120, 121, 122, 123, 124, 125
`\TMP@RequirePackage`
. 134, 140, 141, 142, 143, 453, 459
`\toksdef` 503

U

`\UnDeFiNeD` 1207

W

`\write` ... 23, 52, 219, 1161, 1205, 1257

X

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

Y

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

Z

\z 321, 322