

The pdfTeXcmds package

Heiko Oberdiek

<heiko.oberdiek at gmail.com>

2016/05/10 v0.21

Abstract

LuaTeX provides most of the commands of pdfTeX 1.40. However a number of utility functions are removed. This package tries to fill the gap and implements some of the missing primitive using Lua.

Contents

1	Documentation	2
1.1	General principles	3
1.2	Macros	3
1.2.1	Strings	3
1.2.2	Files	4
1.2.3	Timekeeping	4
1.2.4	Miscellaneous	5
1.2.5	Additional macro: <code>\pdf@isprimitive</code>	6
1.2.6	Experimental	7
2	Implementation	7
2.1	Reload check and package identification	7
2.2	Catcodes	8
2.3	Load packages	9
2.4	Without LuaTeX	10
2.5	<code>\pdf@primitive</code> , <code>\pdf@ifprimitive</code>	11
2.5.1	Using LuaTeX's <code>tex.enableprimitives</code>	11
2.5.2	Trying various names to find the primitives	12
2.5.3	Result	13
2.6	X _Y TeX	13
2.7	<code>\pdf@isprimitive</code>	13
2.8	<code>\pdf@draftmode</code>	14
2.9	Load Lua module	16
2.10	Lua functions	17
2.10.1	Helper macros	17
2.10.2	Strings	18
2.10.3	Files	19
2.10.4	Timekeeping	20
2.10.5	Shell escape	21
2.11	Lua module	22
2.11.1	Strings	22
2.11.2	Files	25
2.11.3	Timekeeping	26
2.11.4	Miscellaneous	26

3	Test	28
3.1	Catcode checks for loading	28
3.2	Test for <code>\pdf@isprimitive</code>	30
3.3	Test for <code>\pdf@shellescape</code>	31
3.4	Test for escape functions	31
4	Installation	34
4.1	Download	34
4.2	Bundle installation	35
4.3	Package installation	35
4.4	Refresh file name databases	35
4.5	Some details for the interested	35
5	Catalogue	36
6	History	36
	[2007/11/11 v0.1]	36
	[2007/11/12 v0.2]	37
	[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]	38
	[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
7	Index	38

1 Documentation

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

- `\pdfstrcmp`
- `\pdfunescapehex`
- `\pdfescapehex`
- `\pdfescapename`
- `\pdfescapestring`
- `\pdffilesize`
- `\pdffilemoddate`
- `\pdffiledump`

- `\pdfmdfivesum`
- `\pdfresettimer`
- `\pdfelapsedtime`
- `\immediate\write18`

The original names of the primitives cannot be used:

- The syntax for their arguments cannot easily simulated by macros. The primitives using key words such as `file` (`\pdfmdfivesum`) or `offset` and `length` (`\pdffiledump`) and uses $\langle general\ text \rangle$ for the other arguments. Using token registers assignments, $\langle general\ text \rangle$ could be caught. However, the simulated primitives are expandable and register assignments would destroy this important property. ($\langle general\ text \rangle$ allows something like `\expandafter\bgroup ...`.)
- The original primitives can be expanded using one expansion step. The new macros need two expansion steps because of the additional macro expansion. Example:

```
\expandafter\foo\pdffilemoddate{file}
vs.
\expandafter\expandafter\expandafter
\foo\pdf@filemoddate{file}
```

LuaTeX isn't stable yet and thus the status of this package is *experimental*. Feedback is welcome.

1.1 General principles

Naming convention: Usually this package defines a macro `\pdf@ $\langle cmd \rangle$` if pdfTeX provides `\pdf $\langle cmd \rangle$` .

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

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

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

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

1.2 Macros

1.2.1 Strings [pdfTeX-manual]

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

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

`\pdf@unescapehex {⟨string⟩}`

Same as `\pdfunescapehex{⟨string⟩}`. The argument is a byte string given in hexadecimal notation. The result are character tokens from 0 until 255 with catcode 12 and the space with catcode 10.

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

Same as the primitives of pdfTeX. However pdfTeX does not know about characters with codes 256 and larger. Thus the string is treated as byte string, characters with more than eight bits are ignored.

1.2.2 Files [pdftex-manual]

`\pdf@filesize {⟨filename⟩}`

Same as `\pdffilesize{⟨filename⟩}`.

`\pdf@filemoddate {⟨filename⟩}`

Same as `\pdffilemoddate{⟨filename⟩}`.

`\pdf@filedump {⟨offset⟩} {⟨length⟩} {⟨filename⟩}`

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

`\pdf@mdfivesum {⟨string⟩}`

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

`\pdf@filemdfivesum {⟨filename⟩}`

Same as `\pdfmdfivesum file{⟨filename⟩}`.

1.2.3 Timekeeping [pdftex-manual]

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

`\pdf@resettimer`

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

`\pdf@elapsedtime`

Same as `\pdfelapsedtime`. It behaves like a read-only integer. For printing purposes it can be prefixed by `\the` or `\number`. It measures the time in scaled

seconds (seconds multiplied with 65536) since the latest call of `\pdf@resettimer` or start of program/package. The resolution, the shortest time interval that can be measured, depends on the program and system.

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

1.2.4 Miscellaneous [pdfTeX-manual]

`\pdf@draftmode`

If the TeX compiler knows `\pdfdraftmode` (pdfTeX, LuaTeX), then `\pdf@draftmode` returns, whether this mode is enabled. The result is an implicate 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 explicite 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
}%
```

```
% \pdfshellescape is available
}
```

Use `\ltx@ifundefined` in expandable contexts.

- `\pdfshellescape` 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\pdfshellescape=0 ...
```

- Print the number: `\number\pdfshellescape`

```
\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

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

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.

<pre>\pdf@pipe {\cmdline}</pre>

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{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 [2016/05/10 v0.21 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\pdf texcmds@AtEnd{%
96      \pdf texcmds@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\pdf texcmds@AtEnd{%
127   \pdf texcmds@AtEnd
128   \escapechar=\number\escapechar\relax
129   \noexpand\endinput
130 }
131 \escapechar=92 %

```

2.3 Load packages

```

132 \begingroup\expandafter\expandafter\expandafter\endgroup
133 \expandafter\ifx\csname RequirePackage\endcsname\relax
134   \def\TMP@RequirePackage#1[#2]{%
135     \begingroup\expandafter\expandafter\expandafter\endgroup
136     \expandafter\ifx\csname ver@#1.sty\endcsname\relax
137       \input #1.sty\relax
138     \fi
139   }%

```

```

140 \TMP@RequirePackage{infwarerr}[2007/09/09]%
141 \TMP@RequirePackage{ifluatex}[2010/03/01]%
142 \TMP@RequirePackage{ltxcmds}[2010/12/02]%
143 \TMP@RequirePackage{ifpdf}[2010/09/13]%
144 \else
145 \RequirePackage{infwarerr}[2007/09/09]%
146 \RequirePackage{ifluatex}[2010/03/01]%
147 \RequirePackage{ltxcmds}[2010/12/02]%
148 \RequirePackage{ifpdf}[2010/09/13]%
149 \fi

```

2.4 Without LuaTeX

```

150 \ifluatex
151 \else
152 \PackageInfoNoLine{pdftexcmds}{LuaTeX not detected}%
153 \def\pdftexcmds@nopdftex{%
154 \PackageInfoNoLine{pdftexcmds}{pdfTeX >= 1.30 not detected}%
155 \let\pdftexcmds@nopdftex\relax
156 }%
157 \def\pdftexcmds@temp#1{%
158 \begingroup\expandafter\expandafter\expandafter\endgroup
159 \expandafter\ifx\csname pdf#1\endcsname\relax
160 \pdftexcmds@nopdftex
161 \else
162 \expandafter\def\csname pdf#1\endcsname\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   \pdfTEXcmds@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   \pdfTEXcmds@nopdfTEX
206 \else
207   \def\pdf@mdfivesum#{\pdfmdfivesum}%
208   \let\pdf@mdfivesumnative\pdf@mdfivesum
209   \def\pdf@filemdfivesum#{\pdfmdfivesum file}%
210 \fi
211 \def\pdf@system#{%
212   \immediate\write18%
213 }%
214 \def\pdfTEXcmds@temp#1{%
215   \begingroup\expandafter\expandafter\expandafter\endgroup
216   \expandafter\ifx\csname pdf#1\endcsname\relax
217     \pdfTEXcmds@nopdfTEX
218   \else
219     \expandafter\let\csname pdf@#1\expandafter\endcsname
220     \csname pdf#1\endcsname
221   \fi
222 }%
223 \pdfTEXcmds@temp{resettimer}%
224 \pdfTEXcmds@temp{elapsedtime}%
225 \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 Lua_{TEX}'s tex.enableprimitives

```

226 \ifluatex
\pdfTEXcmds@directlua
227 \ifnum\luatexversion<36 %
228   \def\pdfTEXcmds@directlua{\directlua0 }%
229 \else
230   \let\pdfTEXcmds@directlua\directlua
231 \fi
232 \begingroup
233   \newlinechar=10 %
234   \endlinechar=\newlinechar
235   \pdfTEXcmds@directlua{%
236     if tex.enableprimitives then
237       tex.enableprimitives(

```

```

238         'pdf@',
239         {'primitive', 'ifprimitive', 'pdfdraftmode'}
240     )
241     tex.enableprimitives('', {'luaescapestring'})
242 end
243 }%
244 \endgroup %
245 \fi

```

2.5.2 Trying various names to find the primitives

\pdftexcmds@strip@prefix

```

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

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

```

\pdf@primitive

```

262 \pdftexcmds@temp{primitive}{pdfprimitive}{pdfprimitive}% pdfTeX, LuaTeX
263 \pdftexcmds@temp{primitive}{primitive}{primitive}% XeTeX
264 \pdftexcmds@temp{primitive}{luatexpdfprimitive}{pdfprimitive}% LuaTeX
265 \pdftexcmds@temp{primitive}{luatexpdfprimitive}{pdfprimitive}% LuaTeX

```

\pdf@ifprimitive

```

266 \pdftexcmds@temp{ifprimitive}{ifpdfprimitive}{ifpdfprimitive}% pdfTeX, LuaTeX
267 \pdftexcmds@temp{ifprimitive}{ifprimitive}{ifprimitive}% XeTeX
268 \pdftexcmds@temp{ifprimitive}{luatexpdfprimitive}{ifpdfprimitive}% LuaTeX
269 \pdftexcmds@temp{ifprimitive}{luatexpdfprimitive}{ifpdfprimitive}% LuaTeX

```

Disable broken \pdfprimitive.

```

270 \begingroup
271   \expandafter\ifx\csname pdf@primitive\endcsname\relax
272   \else
273     \expandafter\ifx\csname pdftexversion\endcsname\relax
274     \else
275       \ifnum\pdftexversion=140 %
276         \expandafter\ifx\csname pdftexrevision\endcsname\relax
277         \else
278           \ifnum\pdftexrevision<4 %
279             \endgroup
280             \let\pdf@primitive@undefined
281             \@PackageInfoNoLine{pdftexcmds}{%
282               \string\pdf@primitive\space disabled, %
283               because\MessageBreak

```

```

284         \string\pdfprimitive\space is broken until pdfTeX 1.40.4%
285     }%
286     \begingroup
287     \fi
288     \fi
289     \fi
290     \fi
291     \fi
292 \endgroup

```

2.5.3 Result

```

293 \begingroup
294 \@PackageInfoNoLine{pdftexcmds}{%
295     \string\pdf@primitive\space is %
296     \expandafter\ifx\csname pdf@primitive\endcsname\relax not \fi
297     available%
298 }%
299 \@PackageInfoNoLine{pdftexcmds}{%
300     \string\pdf@ifprimitive\space is %
301     \expandafter\ifx\csname pdf@ifprimitive\endcsname\relax not \fi
302     available%
303 }%
304 \endgroup

```

2.6 X_YTEX

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

```

305 \def\pdftexcmds@temp#1{%
306     \begingroup\expandafter\expandafter\expandafter\endgroup
307     \expandafter\ifx\csname pdf@#1\endcsname\relax
308         \begingroup
309         \escapechar=-1 %
310         \edef\x{\expandafter\meaning\csname#1\endcsname}%
311         \def\y{#1}%
312         \def\z##1->{}%
313         \edef\y{\expandafter\z\meaning\y}%
314     \expandafter\endgroup
315     \ifx\x\y
316         \expandafter\def\csname pdf@#1\expandafter\endcsname
317         \expandafter{%
318             \csname#1\endcsname
319         }%
320     \fi
321 \fi
322 }%
323 \pdftexcmds@temp{shellescape}%
324 \pdftexcmds@temp{strcmp}%

```

2.7 \pdf@isprimitive

```

325 \def\pdf@isprimitive{%
326     \begingroup\expandafter\expandafter\expandafter\endgroup
327     \expandafter\ifx\csname pdf@strcmp\endcsname\relax
328         \long\def\pdf@isprimitive##1{%
329             \expandafter\pdftexcmds@isprimitive\expandafter{\meaning##1}%
330         }%
331         \long\def\pdftexcmds@isprimitive##1##2{%
332             \expandafter\pdftexcmds@@isprimitive\expandafter{\string##2}{##1}%

```

```

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

```

2.8 \pdf@draftmode

```

377 \let\pdf texcmds@temp\ltx@zero %
378 \ltx@ifundefined{pdfdraftmode}{%
379   \@PackageInfoNoLine{pdf texcmds}{\ltx@backslashchar pdfdraftmode not found}%
380 }{%
381   \ifpdf
382     \let\pdf texcmds@temp\ltx@one
383     \@PackageInfoNoLine{pdf texcmds}{\ltx@backslashchar pdfdraftmode found}%
384   \else
385     \@PackageInfoNoLine{pdf texcmds}{%
386       \ltx@backslashchar pdfdraftmode is ignored in DVI mode%
387     }%
388   \fi

```

```

389 }
390 \ifcase\pdfTexcmds@temp

\pdf@draftmode
391 \let\pdf@draftmode\ltx@zero

\pdf@ifdraftmode
392 \let\pdf@ifdraftmode\ltx@secondoftwo

\pdfTexcmds@setdraftmode
393 \def\pdfTexcmds@setdraftmode#1{%
394 \else

\pdfTexcmds@draftmode
395 \let\pdfTexcmds@draftmode\pdfdraftmode

\pdf@ifdraftmode
396 \def\pdf@ifdraftmode{%
397 \ifnum\pdfTexcmds@draftmode=\ltx@one
398 \expandafter\ltx@firstoftwo
399 \else
400 \expandafter\ltx@secondoftwo
401 \fi
402 }%

\pdf@draftmode
403 \def\pdf@draftmode{%
404 \ifnum\pdfTexcmds@draftmode=\ltx@one
405 \expandafter\ltx@one
406 \else
407 \expandafter\ltx@zero
408 \fi
409 }%

\pdfTexcmds@setdraftmode
410 \def\pdfTexcmds@setdraftmode#1{%
411 \pdfTexcmds@draftmode=#1\relax
412 }%

413 \fi

\pdf@setdraftmode
414 \def\pdf@setdraftmode#1{%
415 \begingroup
416 \count\ltx@cclv=#1\relax
417 \edef\x{\endgroup
418 \noexpand\pdfTexcmds@@setdraftmode{\the\count\ltx@cclv}}%
419 }%
420 \x
421 }

\pdfTexcmds@@setdraftmode
422 \def\pdfTexcmds@@setdraftmode#1{%
423 \ifcase#1 %
424 \pdfTexcmds@setdraftmode{#1}%
425 \or
426 \pdfTexcmds@setdraftmode{#1}%

```

```

427 \else
428   \@PackageWarning{pdftexcmds}{%
429     \string\pdf@setdraftmode: Ignoring\MessageBreak
430     invalid value '#1'%
431   }%
432 \fi
433 }

```

2.9 Load Lua module

```

434 \ifluatex
435 \else
436   \expandafter\pdf@setdraftmode\AtEnd
437 \fi%

438 \begingroup\expandafter\expandafter\expandafter\endgroup
439 \expandafter\ifx\csname RequirePackage\endcsname\relax
440   \def\TMP@RequirePackage#1[#2]{%
441     \begingroup\expandafter\expandafter\expandafter\endgroup
442     \expandafter\ifx\csname ver@#1.sty\endcsname\relax
443       \input #1.sty\relax
444     \fi
445   }%
446   \TMP@RequirePackage{luatex-loader}[2009/04/10]%
447 \else
448   \RequirePackage{luatex-loader}[2009/04/10]%
449 \fi

450 \pdf@setdraftmode\directlua{%
451   require("oberdiek.pdftexcmds")%
452 }

453 \ifnum\luatexversion>37 %
454   \ifnum0%
455     \pdf@setdraftmode\directlua{%
456       if status.ini_version then %
457         tex.write("1")%
458       end%
459     }>0 %
460     \everyjob\expandafter{%
461       \the\everyjob
462       \pdf@setdraftmode\directlua{%
463         require("oberdiek.pdftexcmds")%
464       }%
465     }%
466   \fi
467 \fi

468 \begingroup
469   \def\x{2016/05/10 v0.21}%
470   \ltx@onelevel@sanitize\x
471   \edef\y{%
472     \pdf@setdraftmode\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   }%

```

\pdftexcmds@DecodeB

```

517 \def\pdftexcmds@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       \pdftexcmds@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 [pdfTeX-manual]

\pdf@strcmp

```

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

540 \pdf@isprimitive

```

\pdf@escapehex

```

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

```

\pdf@escapehexnative

```

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

```

\pdf@unescapehex

```

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

```

\pdf@unescapehexnative

```

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

```

\pdf@escapestring

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

\pdf@escapename

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

\pdf@escapenamenative

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

2.10.3 Files [pdfTeX-manual]

\pdf@filesize

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

\pdf@filemoddate

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

\pdf@filedump

```
594 \def\pdf@filedump#1#2#3{%
595   \directlua0{%
596     oberdiek.pdfcmds.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.pdfcmds.mdfivesum("\luaescapestring{#1}", "byte")%
604   }%
605 }
```

\pdf@mdfivesumnative

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

```

609 }%
610 }%

\pdf@filemdfivesum

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

```

2.10.4 Timekeeping [pdfdoc-manual]

```

\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     \pdfdoccmds@directlua0{%
653       oberdiek.pdfdoccmds.resettimer()%
654     }%
655   }%

```

`\pdf@elapsedtime`

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

663 \fi
```

2.10.5 Shell escape

`\pdf@shellescape` Caution: Catcode of digit zero might be ‘ignore’.

```
664 \ifnum\luatexversion<68 %
665 \else
666 \def\pdf@shellescape{%
667   \directlua{%
668     oberdiek.pdftexcmds.shellescape()}%
669   }%
670 }%
671 \fi
```

`\pdf@system`

```
672 \def\pdf@system#1{%
673   \directlua{%
674     oberdiek.pdftexcmds.system("\luaescapestring{#1}")%
675   }%
676 }
```

`\pdf@lastsystemstatus`

```
677 \def\pdf@lastsystemstatus{%
678   \directlua{%
679     oberdiek.pdftexcmds.lastsystemstatus()}%
680   }%
681 }
```

`\pdf@lastsystemexit`

```
682 \def\pdf@lastsystemexit{%
683   \directlua{%
684     oberdiek.pdftexcmds.lastsystemexit()}%
685   }%
686 }

687 \catcode'\0=12 %
```

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

```
688 \ifnum0%
689   \pdftexcmds@directlua{%
690     if io.popen then %
691       tex.write("1")%
692     end%
693   }%
694   =1 %
695 \def\pdf@pipe#1{%
696 & \romannumeral\expandafter\pdftexcmds@PatchDecode
697   \the\expandafter\pdftexcmds@toks
698   \pdftexcmds@directlua{%
```

```

699     oberdiek.pdfdoccmds.toks="pdfdoccmds@toks"%
700     oberdiek.pdfdoccmds.pipe("\luaescapestring{#1}", \pdfdoccmds@Patch)%
701   }%
702 &   \@nil
703 }%
704 \fi

705 \pdfdoccmds@AtEnd%
706 \endpackage

```

2.11 Lua module

```

707 \lua

708 module("oberdiek.pdfdoccmds", package.seeall)
709 local systemexitstatus
710 function getversion()
711   tex.write("2016/05/10 v0.21")
712 end

```

2.11.1 Strings [pdfdoc-manual]

```

713 function 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 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 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(toks, string.char(unpack(result)))
822 end

```

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

```

823 function 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 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 [pdftex-manual]

```
861 function 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 pdf_T_EX.

```
870 function 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%s",
903                 d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
904         end
905     end
906 end
907 function 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, "r")
917         if filehandle then
918             if offset > 0 then
919                 filehandle:seek("set", offset)
920             end
921             local dump = filehandle:read(length)
922             escapehex(dump)
923         end
924     end
925 end
926 end
927 function md5sum(str, mode)
928     if mode == "byte" then
929         str = utf8_to_byte(str)
930     end
931     escapehex(md5.sum(str))
932 end
933 function filemd5sum(filename)
934     local foundfile = kpse.find_file(filename, "tex", true)
935     if foundfile then
936         local filehandle = io.open(foundfile, "r")
937         if filehandle then
938             local contents = filehandle:read("*a")
939             escapehex(md5.sum(contents))
940         end
941     end
942 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`.

```

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

```

2.11.4 Miscellaneous [pdftex-manual]

```

954 function shellescape()
955   if os.execute then
956     if status
957       and status.luatex_version
958       and status.luatex_version >= 68 then
959       tex.write(os.execute())
960     else
961       local result = os.execute()
962       if result == 0 then
963         tex.write("0")
964       else
965         if result == nil then
966           tex.write("0")
967         else
968           tex.write("1")
969         end
970       end
971     end
972   else
973     tex.write("0")
974   end
975 end
976 function system(cmdline)
977   systemexitstatus = nil
978   texio.write_nl("log", "system(" .. cmdline .. ") ")
979   if os.execute then
980     texio.write("log", "executed.")
981     systemexitstatus = os.execute(cmdline)
982   else
983     texio.write("log", "disabled.")
984   end
985 end
986 function lastsystemstatus()
987   local result = tonumber(systemexitstatus)
988   if result then
989     local x = math.floor(result / 256)
990     tex.write(result - 256 * math.floor(result / 256))
991   end
992 end
993 function lastsystemexit()
994   local result = tonumber(systemexitstatus)
995   if result then
996     tex.write(math.floor(result / 256))
997   end
998 end
999 function pipe(cmdline, patch)
1000   local result
1001   systemexitstatus = nil
1002   texio.write_nl("log", "pipe(" .. cmdline .. ") ")
1003   if io.popen then
1004     texio.write("log", "executed.")
1005     local handle = io.popen(cmdline, "r")
1006     if handle then
1007       result = handle:read("*a")
1008       handle:close()
1009     end
1010   else
1011     texio.write("log", "disabled.")

```

```

1012 end
1013 if result then
1014   if patch == 1 then
1015     local temp = {}
1016     for i, a in ipairs(result) do
1017       if a == 0 then
1018         table.insert(temp, 1)
1019         table.insert(temp, 1)
1020       else
1021         if a == 1 then
1022           table.insert(temp, 1)
1023           table.insert(temp, 2)
1024         else
1025           table.insert(temp, a)
1026         end
1027       end
1028     end
1029     result = temp
1030   end
1031   tex.settoks(toks, result)
1032 else
1033   tex.settoks(toks, "")
1034 end
1035 end
1036  $\langle$ /lua $\rangle$ 

```

3 Test

3.1 Catcode checks for loading

```

1037  $\langle$ *test1 $\rangle$ 
1038 \catcode'\{=1 %
1039 \catcode'\}=2 %
1040 \catcode'\#=6 %
1041 \catcode'\@=11 %
1042 \expandafter\ifx\csname count@\endcsname\relax
1043   \countdef\count@=255 %
1044 \fi
1045 \expandafter\ifx\csname @gobble\endcsname\relax
1046   \long\def\@gobble#1{}%
1047 \fi
1048 \expandafter\ifx\csname @firstofone\endcsname\relax
1049   \long\def\@firstofone#1{#1}%
1050 \fi
1051 \expandafter\ifx\csname loop\endcsname\relax
1052   \expandafter\@firstofone
1053 \else
1054   \expandafter\@gobble
1055 \fi
1056 {%
1057   \def\loop#1\repeat{%
1058     \def\body{#1}%
1059     \iterate
1060   }%
1061   \def\iterate{%
1062     \body
1063     \let\next\iterate
1064   \else

```

```

1065     \let\next\relax
1066     \fi
1067     \next
1068   }%
1069   \let\repeat=\fi
1070 }%
1071 \def\RestoreCatcodes{}
1072 \count@=0 %
1073 \loop
1074   \edef\RestoreCatcodes{%
1075     \RestoreCatcodes
1076     \catcode\the\count@=\the\catcode\count@\relax
1077   }%
1078 \ifnum\count@<255 %
1079   \advance\count@ 1 %
1080 \repeat
1081
1082 \def\RangeCatcodeInvalid#1#2{%
1083   \count@=#1\relax
1084   \loop
1085     \catcode\count@=15 %
1086   \ifnum\count@<#2\relax
1087     \advance\count@ 1 %
1088   \repeat
1089 }
1090 \def\RangeCatcodeCheck#1#2#3{%
1091   \count@=#1\relax
1092   \loop
1093     \ifnum#3=\catcode\count@
1094   \else
1095     \errmessage{%
1096       Character \the\count@\space
1097       with wrong catcode \the\catcode\count@\space
1098       instead of \number#3%
1099     }%
1100   \fi
1101   \ifnum\count@<#2\relax
1102     \advance\count@ 1 %
1103   \repeat
1104 }
1105 \def\space{ }
1106 \expandafter\ifx\csname LoadCommand\endcsname\relax
1107   \def\LoadCommand{\input pdftexcmds.sty\relax}%
1108 \fi
1109 \def\Test{%
1110   \RangeCatcodeInvalid{0}{47}%
1111   \RangeCatcodeInvalid{58}{64}%
1112   \RangeCatcodeInvalid{91}{96}%
1113   \RangeCatcodeInvalid{123}{255}%
1114   \catcode'\@=12 %
1115   \catcode'\=0 %
1116   \catcode'\%=14 %
1117   \LoadCommand
1118   \RangeCatcodeCheck{0}{36}{15}%
1119   \RangeCatcodeCheck{37}{37}{14}%
1120   \RangeCatcodeCheck{38}{47}{15}%
1121   \RangeCatcodeCheck{48}{57}{12}%
1122   \RangeCatcodeCheck{58}{63}{15}%

```

```

1123 \RangeCatcodeCheck{64}{64}{12}%
1124 \RangeCatcodeCheck{65}{90}{11}%
1125 \RangeCatcodeCheck{91}{91}{15}%
1126 \RangeCatcodeCheck{92}{92}{0}%
1127 \RangeCatcodeCheck{93}{96}{15}%
1128 \RangeCatcodeCheck{97}{122}{11}%
1129 \RangeCatcodeCheck{123}{255}{15}%
1130 \RestoreCatcodes
1131 }
1132 \Test
1133 \csname @@end\endcsname
1134 \end
1135 \</test1>

```

3.2 Test for \pdf@isprimitive

```

1136 \<test2>
1137 \catcode'\{=1 %
1138 \catcode'\}=2 %
1139 \catcode'\#=6 %
1140 \catcode'\@=11 %
1141 \input pdftexcmds.sty\relax
1142 \def\msg#1{%
1143   \begingroup
1144     \escapechar=92 %
1145     \immediate\write16{#1}%
1146   \endgroup
1147 }
1148 \long\def\test#1#2#3#4{%
1149   \begingroup
1150     #4%
1151     \def\str{%
1152       Test \string\pdf@isprimitive
1153       {\string #1}{\string #2}{...}: %
1154     }%
1155     \pdf@isprimitive{#1}{#2}{%
1156       \ifx#3Y%
1157         \msg{\str true ==> OK.}%
1158       \else
1159         \errmessage{\str false ==> FAILED}%
1160       \fi
1161     }{%
1162       \ifx#3Y%
1163         \errmessage{\str true ==> FAILED}%
1164       \else
1165         \msg{\str false ==> OK.}%
1166       \fi
1167     }%
1168   \endgroup
1169 }
1170 \test\relax\relax Y{}
1171 \test\foobar\relax Y{\let\foobar\relax}
1172 \test\foobar\relax N{}
1173 \test\hbox\hbox Y{}
1174 \test\foobar@hbox\hbox Y{\let\foobar@hbox\hbox}
1175 \test\if\if Y{}
1176 \test\if\ifx N{}
1177 \test\ifx\if N{}
1178 \test\par\par Y{}

```

```

1179 \test\hbox\par N{}
1180 \test\par\hbox N{}
1181 \csname @@end\endcsname\end
1182 /test2)

```

3.3 Test for \pdf@shellescape

```

1183 (*test-shell)
1184 \catcode'\{=1 %
1185 \catcode'\}=2 %
1186 \catcode'\#=6 %
1187 \catcode'\@=11 %
1188 \input pdftexcmds.sty\relax
1189 \def\msg#{\immediate\write16}
1190 \def\MaybeEnd{}
1191 \ifx\luatexversion\UnDeFiNeD
1192 \else
1193   \ifnum\luatexversion<68 %
1194     \ifx\pdf@shellescape\undefined
1195       \msg{SHELL=U}%
1196       \msg{OK (LuaTeX < 0.68)}%
1197     \else
1198       \msg{SHELL=defined}%
1199       \errmessage{Failed (LuaTeX < 0.68)}%
1200     \fi
1201     \def\MaybeEnd{\csname @@end\endcsname\end}%
1202   \fi
1203 \fi
1204 \MaybeEnd
1205 \ifx\pdf@shellescape\undefined
1206   \msg{SHELL=U}%
1207 \else
1208   \msg{SHELL=\number\pdf@shellescape}%
1209 \fi
1210 \ifx\expected\undefined
1211 \else
1212   \ifx\expected\relax
1213     \msg{EXPECTED=U}%
1214     \ifx\pdf@shellescape\undefined
1215       \msg{OK}%
1216     \else
1217       \errmessage{Failed}%
1218     \fi
1219   \else
1220     \msg{EXPECTED=\number\expected}%
1221     \ifnum\pdf@shellescape=\expected\relax
1222       \msg{OK}%
1223     \else
1224       \errmessage{Failed}%
1225     \fi
1226   \fi
1227 \fi
1228 \csname @@end\endcsname\end
1229 /test-shell)

```

3.4 Test for escape functions

```

1230 (*test-escape)
1231 \catcode'\{=1 %
1232 \catcode'\}=2 %

```

```

1233 \catcode'\#=6 %
1234 \catcode'\^=7 %
1235 \catcode'\@=11 %
1236 \errorcontextlines=1000 %
1237 \input pdftexcmds.sty\relax
1238 \def\msg#1{%
1239   \begingroup
1240     \escapechar=92 %
1241     \immediate\write16{#1}%
1242   \endgroup
1243 }
1244 \begingroup
1245   \catcode'\@=11 %
1246   \countdef\count@=255 %
1247   \def\space{ }%
1248   \long\def\@whilenum#1\do #2{%
1249     \ifnum #1\relax
1250       #2\relax
1251     \@iwhilenum{#1\relax#2\relax}%
1252   \fi
1253 }%
1254 \long\def\@iwhilenum#1{%
1255   \ifnum #1%
1256     \expandafter\@iwhilenum
1257   \else
1258     \expandafter\ltx@gobble
1259   \fi
1260   {#1}%
1261 }%
1262 \gdef\AllBytes{%
1263   \count@=0 %
1264   \catcode0=12 %
1265   \@whilenum\count@<256 \do{%
1266     \lccode0=\count@
1267     \ifnum\count@=32 %
1268       \xdef\AllBytes{\AllBytes\space}%
1269     \else
1270       \lowercase{%
1271         \xdef\AllBytes{\AllBytes^^@}%
1272       }%
1273     \fi
1274     \advance\count@ by 1 %
1275   }%
1276 \endgroup
1277 \def\AllBytesHex{%
1278   000102030405060708090A0B0C0D0E0F%
1279   101112131415161718191A1B1C1D1E1F%
1280   202122232425262728292A2B2C2D2E2F%
1281   303132333435363738393A3B3C3D3E3F%
1282   404142434445464748494A4B4C4D4E4F%
1283   505152535455565758595A5B5C5D5E5F%
1284   606162636465666768696A6B6C6D6E6F%
1285   707172737475767778797A7B7C7D7E7F%
1286   808182838485868788898A8B8C8D8E8F%
1287   909192939495969798999A9B9C9D9E9F%
1288   A0A1A2A3A4A5A6A7A8A9AAABACADAFAF%
1289   B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF%
1290   C0C1C2C3C4C5C6C7C8C9CACBCCDCECF%

```



```

1291 D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
1292 E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
1293 F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF%
1294 }
1295 \ltx@onelevel@sanitize\AllBytesHex
1296 \expandafter\lowercase\expandafter{%
1297 \expandafter\def\expandafter\AllBytesHexLC
1298 \expandafter{\AllBytesHex}%
1299 }
1300 \begingroup
1301 \catcode'\#=12 %
1302 \xdef\AllBytesName{%
1303 #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
1304 #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
1305 #20!"#23$#25&'#28#29*+,-.#2F%
1306 0123456789:;#3C=#3E?%
1307 @ABCDEFGHJKLMNOP%
1308 PQRSTUVWXYZ#5B\ltx@backslashchar#5D^_%
1309 'abcdefghijklmno%
1310 pqrstuvwxyz#7B|#7D|string~#7F%
1311 #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
1312 #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
1313 #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
1314 #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
1315 #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
1316 #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
1317 #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
1318 #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
1319 }%
1320 \endgroup
1321 \ltx@onelevel@sanitize\AllBytesName
1322 \edef\AllBytesFromName{\expandafter\ltx@gobble\AllBytes}
1323 \begingroup
1324 \def\|{|}%
1325 \edef%\{\ltx@percentchar}%
1326 \catcode'\|=0 %
1327 \catcode'\#=12 %
1328 \catcode'\~=12 %
1329 \catcode'\|=12 %
1330 |xdef\AllBytesString{%
1331 \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
1332 \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
1333 \040!"#$%&'(\)*+,-./%
1334 0123456789:;<=>?%
1335 @ABCDEFGHJKLMNOP%
1336 PQRSTUVWXYZ[\]^_%
1337 'abcdefghijklmno%
1338 pqrstuvwxyz{|}~\177%
1339 \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
1340 \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
1341 \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
1342 \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
1343 \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
1344 \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
1345 \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
1346 \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
1347 }%
1348 \endgroup

```

```

1349 \ltx@onelevel@sanitize\AllBytesString
1350 \def\Test#1#2#3{%
1351   \begingroup
1352     \expandafter\expandafter\expandafter\def
1353       \expandafter\expandafter\expandafter\TestResult
1354         \expandafter\expandafter\expandafter{%
1355           #1{#2}%
1356         }%
1357     \ifx\TestResult#3%
1358     \else
1359       \newlinechar=10 %
1360       \msg{Expect:^^J#3}%
1361       \msg{Result:^^J\TestResult}%
1362       \errmessage{\string#2 -\string#1-> \string#3}%
1363     \fi
1364   \endgroup
1365 }
1366 \def\test#1#2#3{%
1367   \edef\TestFrom{#2}%
1368   \edef\TestExpect{#3}%
1369   \ltx@onelevel@sanitize\TestExpect
1370   \Test#1\TestFrom\TestExpect
1371 }
1372 \test\pdf@unescapehex{74657374}{test}
1373 \begingroup
1374   \catcode0=12 %
1375   \catcode1=12 %
1376   \test\pdf@unescapehex{740074017400740174}{t^^@t^^At^^@t^^At}%
1377 \endgroup
1378 \Test\pdf@escapehex\AllBytes\AllBytesHex
1379 \Test\pdf@unescapehex\AllBytesHex\AllBytes
1380 \Test\pdf@escapename\AllBytes\AllBytesName
1381 \Test\pdf@escapestring\AllBytes\AllBytesString
1382 \csname @@end\endcsname\end
1383 </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.

¹[ftp://ftp.ctan.org/tex-archive/](http://ftp.ctan.org/tex-archive/)

4.2 Bundle installation

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

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

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

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

4.3 Package installation

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

```
tex pdftexcmds.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as texmf tree):

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

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

4.4 Refresh file name databases

If your T_EX distribution (teT_EX, miK_TE_X, ...) relies on file name databases, you must refresh these. For example, teT_EX users run texhash or mktexlsr.

4.5 Some details for the interested

Attached source. The PDF documentation on CTAN also includes the .dtx source file. It can be extracted by AcrobatReader 6 or higher. Another option is pdftk, e.g. unpack the file into the current directory:

```
pdftk pdftexcmds.pdf unpack_files output .
```

Unpacking with L^AT_EX. The .dtx chooses its action depending on the format:
plain T_EX: Run docstrip and extract the files.

L^AT_EX: Generate the documentation.

If you insist on using L^AT_EX for docstrip (really, docstrip does not need L^AT_EX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{pdftexcmds.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfL^AT_EX:

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

5 Catalogue

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

```
1384 (*catalogue)
1385 <?xml version='1.0' encoding='us-ascii'?>
1386 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
1387 <entry datestamp='$Date$' modifier='$Author$' id='pdftexcmds'>
1388   <name>pdftexcmds</name>
1389   <caption>LuaTeX support for pdfTeX utility functions.</caption>
1390   <authorref id='auth:oberdiek'/>
1391   <copyright owner='Heiko Oberdiek' year='2007,2009-2011'/>
1392   <license type='lppl1.3'/>
1393   <version number='0.20'/>
1394   <description>
1395     LuaTeX provides most of the commands of
1396     <xref refid='pdftex'>pdfTeX</xref> 1.40. However, a number of
1397     utility functions are not available. This package tries to fill
1398     the gap and implements some of the missing primitives using Lua.
1399     <p/>
1400     The package is part of the <xref refid='oberdiek'>oberdiek</xref>
1401     bundle.
1402   </description>
1403   <documentation details='Package documentation'
1404     href='ctan:/macros/latex/contrib/oberdiek/pdftexcmds.pdf'/>
1405   <ctan file='true' path='/macros/latex/contrib/oberdiek/pdftexcmds.dtx'/>
1406   <miktex location='oberdiek'/>
1407   <texlive location='oberdiek'/>
1408   <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip'/>
1409 </entry>
1410 </catalogue>
```

6 History

[2007/11/11 v0.1]

- First version.

[2007/11/12 v0.2]

- Short description fixed.

[2007/12/12 v0.3]

- Organization of Lua code as module.

[2009/04/10 v0.4]

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

[2009/09/22 v0.5]

- `\pdf@primitive`, `\pdf@ifprimitive` added.
- 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).

7 Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.

Symbols	
<code>\#</code> .. 1040, 1139, 1186, 1233, 1301, 1327	<code>\@PackageWarning</code> 428
<code>\%</code> 1116, 1325	<code>\@PackageWarningNoLine</code> 642
<code>\&</code> 500, 502	<code>\@ehc</code> 484
<code>\(</code> 1333	<code>\@firstofone</code> 1049, 1052
<code>\)</code> 1333	<code>\@gobble</code> 1046, 1054
<code>\@</code> .. 1041, 1114, 1140, 1187, 1235, 1245	<code>\@iwhilenum</code> 1251, 1254, 1256
<code>\@PackageError</code> 480	<code>\@nil</code> 503, 504, 506,
<code>\@PackageInfoNoLine</code> 152,	509, 513, 517, 525, 558, 567, 702
154, 281, 294, 299, 379, 383, 385	<code>\@undefined</code> 58, 280, 1194, 1205, 1210, 1214

\@whilenum	1248, 1265		
\\	831, 834, 1115, 1329, 1336		
\{	1038, 1137, 1184, 1231		
\}	1039, 1138, 1185, 1232		
\^	1234		
\	1324, 1326		
\~	1328		
Numbers			
\0	532, 687, 1331, 1332, 1333		
\1	1338		
\2	1339, 1340, 1341, 1342		
\3	1343, 1344, 1345, 1346		
A			
\advance	1079, 1087, 1102, 1274		
\aftergroup	29		
\AllBytes	1262, 1268, 1271, 1322, 1378, 1379, 1380, 1381		
\AllBytesFromName	1322		
\AllBytesHex	1277, 1295, 1298, 1378, 1379		
\AllBytesHexLC	1297		
\AllBytesName	1302, 1321, 1380		
\AllBytesString	1349, 1381		
B			
\body	1058, 1062		
C			
\catcode	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, 687, 1038, 1039, 1040, 1041, 1076, 1085, 1093, 1097, 1114, 1115, 1116, 1137, 1138, 1139, 1140, 1184, 1185, 1186, 1187, 1231, 1232, 1233, 1234, 1235, 1245, 1264, 1301, 1326, 1327, 1328, 1329, 1374, 1375		
\chardef	184, 186		
\count	416, 418		
\count@	1043, 1072, 1076, 1078, 1079, 1083, 1085, 1086, 1087, 1091, 1093, 1096, 1097, 1101, 1102, 1246, 1263, 1265, 1266, 1267, 1274		
\countdef	1043, 1246		
\csname	14, 21, 50, 66, 76, 133, 136, 159, 162, 164, 178, 196, 204, 216, 219, 220, 249, 254, 257, 258, 271, 273, 276, 296, 301, 307, 310, 316, 318, 327, 362, 439, 442, 488, 491, 618, 626, 630, 638, 1042, 1045, 1048, 1051, 1106, 1133, 1181, 1201, 1228, 1382		
D			
\delimiter	335, 341, 359		
\directlua	228, 230, 535, 542, 547, 554, 563, 570, 575, 580, 585, 590, 595, 602, 607, 612, 667, 673, 678, 683		
\do	1248, 1265		
E			
\empty	17, 18		
\end	1134, 1181, 1201, 1228, 1382		
\endcsname	14, 21, 50, 66, 76, 133, 136, 159, 162, 164, 178, 196, 204, 216, 219, 220, 249, 254, 257, 258, 271, 273, 276, 296, 301, 307, 310, 316, 318, 327, 362, 439, 442, 488, 491, 618, 626, 630, 638, 1042, 1045, 1048, 1051, 1106, 1133, 1181, 1201, 1228, 1382		
\endinput	29, 129		
\endlinechar	4, 35, 71, 77, 89, 234		
\errmessage	1095, 1159, 1163, 1199, 1217, 1224, 1362		
\errorcontextlines	1236		
\escapechar	128, 131, 253, 309, 1144, 1240		
\everyjob	460, 461		
\expected	1210, 1212, 1220, 1221		
F			
\foobar	1171, 1172		
\foobar@hbox	1174		
G			
\gdef	1262		
H			
\hbox	1173, 1174, 1179, 1180		
I			
\if	1175, 1176, 1177		
\ifcase	390, 423, 499		
\ifeof	182, 183		
\ifluatex	150, 226, 373, 434		
\ifnum	182, 227, 275, 278, 335, 365, 397, 404, 453, 454, 494, 495, 530, 664, 688, 1078, 1086, 1093, 1101, 1193, 1221, 1249, 1255, 1267		
\ifpdf	381		
\ifx	15, 18, 21, 50, 58, 61, 133, 136, 159, 178, 196, 204, 216, 249, 256, 271, 273, 276, 296, 301, 307, 315, 327, 342, 343, 346, 348, 439, 442, 478, 488, 507, 518, 618, 626, 630, 638, 641, 1042, 1045, 1048, 1051, 1106, 1156, 1162, 1176, 1177, 1191, 1194, 1205, 1210, 1212, 1214, 1357		
\immediate	23, 52, 212, 1145, 1189, 1241		

\iterate 1059, 1061, 1063

L

\lccode 1266
\LoadCommand 1107, 1117
\loop 1057, 1073, 1084, 1092
\lowercase 1270, 1296
\ltx@backslashchar 379,
383, 386, 643, 645, 647, 648, 1308
\ltx@cclv 416, 418
\ltx@firstoftwo 336, 366, 398
\ltx@gobble 1258, 1322
\ltx@ifUndefined 180, 378
\ltx@one 382, 397, 404, 405
\ltx@onelevel@sanitize
. 470, 1295, 1321, 1349, 1369
\ltx@percentchar 1325
\ltx@ReturnAfterElseFi 508, 519
\ltx@ReturnAfterFi 512, 524
\ltx@secondoftwo 338, 368, 392, 400
\ltx@zero 377, 391, 407, 520
\luaescapestring
. 536, 537, 543, 548, 556, 565,
571, 576, 581, 586, 591, 596,
597, 598, 603, 608, 613, 674, 700
\luatexversion 227,
453, 494, 495, 530, 664, 1191, 1193

M

\MaybeEnd 1190, 1201, 1204
\meaning 252, 254, 310, 313, 329, 365
\MessageBreak
. 283, 429, 481, 482, 644, 646
\msg 1142, 1157, 1165, 1189, 1195,
1196, 1198, 1206, 1208, 1213,
1215, 1220, 1222, 1238, 1360, 1361

N

\newlinechar 233, 234, 1359
\next 1063, 1065, 1067
\number 128, 596, 597, 1098, 1208, 1220
\numexpr 629, 657

P

\PackageInfo 26
\par 1178, 1179, 1180
\pdf@draftmode 5, 391, 403
\pdf@elapseddtime 4, 656
\pdf@escapehex 4, 170, 541, 1378
\pdf@escapehexnative 170, 546
\pdf@escapename 574, 1380
\pdf@escapenamenative 579
\pdf@escapestring 569, 1381
\pdf@filedump 4, 199, 594
\pdf@filemdfivesum 4, 209, 611
\pdf@filemoddate 4, 589
\pdf@filesize 4, 584
\pdf@ifdraftmode 5, 392, 396
\pdf@ifprimitive 6, 266, 300
\pdf@isprimitive 6,
325, 328, 364, 375, 540, 1152, 1155
\pdf@lastsystemexit 682
\pdf@lastsystemstatus 677
\pdf@mdfivesum 4, 207, 208, 601
\pdf@mdfivesumnative 208, 606
\pdf@pipe 7, 688
\pdf@primitive 6, 262, 280, 282, 295
\pdf@resettimer 4, 651
\pdf@setdraftmode 5, 414, 429
\pdf@shellescape 5, 184, 186, 191,
664, 1194, 1205, 1208, 1214, 1221
\pdf@strcmp 3, 365, 534
\pdf@system 6, 211, 672
\pdf@unescapehex
. 4, 172, 551, 1372, 1376, 1379
\pdf@unescapehexnative 7, 172, 560
\pdf@draftmode 395
\pdf@filedump 200
\pdf@mdfivesum 207, 209
\pdf@primitive 284
\pdf@shellescape 192
\pdf@texcmds@isprimitive 332, 334
\pdf@texcmds@setdraftmode 418, 422
\pdf@texcmds@AtEnd
. 95, 96, 126, 127, 436, 705
\pdf@texcmds@DecodeA 504, 506
\pdf@texcmds@DecodeB 509, 517
\pdf@texcmds@directlua
. 227, 235, 450, 455, 462,
472, 619, 631, 652, 658, 689, 698
\pdf@texcmds@draftmode
. 395, 397, 404, 411
\pdf@texcmds@equal 335, 341, 359
\pdf@texcmds@equalcont
. 350, 356, 357, 362
\pdf@texcmds@isprimitive 329, 331
\pdf@texcmds@nopdftex
. 153, 155, 160, 179, 197, 205, 217
\pdf@texcmds@Patch
. 493, 499, 556, 565, 700
\pdf@texcmds@PatchDecode
. 503, 552, 561, 696
\pdf@texcmds@setdraftmode
. 393, 410, 424, 426
\pdf@texcmds@strip@prefix 246, 252
\pdf@texcmds@temp
. 157, 168, 169, 171, 173,
174, 175, 176, 214, 223, 224,
247, 262, 263, 264, 265, 266,
267, 268, 269, 305, 323, 324,
377, 382, 390, 616, 627, 639, 641
\pdf@texcmds@toks 487, 553, 562, 697
\pdf@texrevision 278
\pdf@texversion 182, 275
\protected 616, 651, 656
\ProvidesPackage 19, 67

R		461, 553, 562, 697, 1076, 1096, 1097
\RangeCatcodeCheck . . .	1090, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129	\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
\RangeCatcodeInvalid	1082, 1110, 1111, 1112, 1113	\TMP@RequirePackage
\repeat . .	1057, 1069, 1080, 1088, 1103	. 134, 140, 141, 142, 143, 440, 446
\RequirePackage	145, 146, 147, 148, 448	\toksdef 489
\RestoreCatcodes	1071, 1074, 1075, 1130	
\romannumeral	552, 561, 696	
S		U
\space	282, 284, 295, 300, 1096, 1097, 1105, 1247, 1268	\UnDeFiNeD 1191
\str	1151, 1157, 1159, 1163, 1165	
T		W
\Test	1109, 1132, 1350, 1370, 1378, 1379, 1380, 1381	\write . . . 23, 52, 212, 1145, 1189, 1241
\test	1148, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1366, 1372, 1376	
\TestExpect	1368, 1369, 1370	X
\TestFrom	1367, 1370	\x . . . 14, 15, 18, 22, 26, 28, 51, 56, 66, 75, 87, 251, 252, 256, 310, 315, 417, 420, 469, 470, 478, 482
\TestResult	1353, 1357, 1361	
\the	77, 78, 79, 80, 81, 82, 83, 84, 97, 418,	Y
		\y 254, 256, 311, 313, 315, 471, 478, 483
		Z
		\z 312, 313