

# The pdftextcmds package

Heiko Oberdiek  
<heiko.oberdiek at gmail.com>

2011/04/16 v0.15

## 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

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	General principles	3
1.2	Macros	3
1.2.1	Additional macro: <code>\pdf@isprimitive</code>	5
1.2.2	Experimental	5
<b>2</b>	<b>Implementation</b>	<b>5</b>
2.1	Reload check and package identification	5
2.2	Catcodes	7
2.3	Load packages	8
2.4	Without LuaTeX	8
2.5	<code>\pdf@primitive</code> , <code>\pdf@ifprimitive</code>	9
2.5.1	Using LuaTeX's <code>tex.enableprimitives</code>	9
2.5.2	Trying various names to find the primitives	10
2.5.3	Result	11
2.6	<code>X<sub>g</sub>TeX</code>	11
2.7	<code>\pdf@isprimitive</code>	11
2.8	<code>\pdf@draftmode</code>	12
2.9	Load Lua module	13
2.10	Lua functions	14
2.11	Lua module	18
<b>3</b>	<b>Test</b>	<b>23</b>
3.1	Catcode checks for loading	23
3.2	Test for <code>\pdf@isprimitive</code>	25
3.3	Test for <code>\pdf@shellescape</code>	26
3.4	Test for escape functions	26
<b>4</b>	<b>Installation</b>	<b>29</b>
4.1	Download	29
4.2	Bundle installation	29
4.3	Package installation	29
4.4	Refresh file name databases	30
4.5	Some details for the interested	30

<b>5</b>	<b>History</b>	<b>30</b>
	[2007/11/11 v0.1]	30
	[2007/11/12 v0.2]	30
	[2007/12/12 v0.3]	30
	[2009/04/10 v0.4]	30
	[2009/09/22 v0.5]	31
	[2009/09/23 v0.6]	31
	[2009/12/12 v0.7]	31
	[2010/03/01 v0.8]	31
	[2010/04/01 v0.9]	31
	[2010/11/04 v0.10]	31
	[2010/11/11 v0.11]	31
	[2011/01/30 v0.12]	31
	[2011/03/04 v0.13]	31
	[2011/04/10 v0.14]	31
	[2011/04/16 v0.15]	31
<b>6</b>	<b>Index</b>	<b>31</b>

## 1 Documentation

Some primitives of pdfTeX are not defined by LuaTeX. This package implements macro based solutions using Lua code for the following missing pdfTeX primitives;

- `\pdfstrcmp`
- `\pdfunescapehex`
- `\pdfescapehex`
- `\pdfescapename`
- `\pdfescapestring`
- `\pdffilesize`
- `\pdffilemoddate`
- `\pdffiledump`
- `\pdfmdfivesum`
- `\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 *general text* for the other arguments. Using token registers assignments, *general text* could be caught. However, the simulated primitives are expandable and register assignments would destroy this important property. (*general text* allows something like `\expandafter\bgroup ...}`.)
- The original primitives can be expanded using one expansion step. The new macros need two expansion steps because of the additional macro expansion. Example:

```

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

```

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

## 1.1 General principles

**Naming convention:** Usually this package defines a macro `\pdf@<cmd>` if pdfTeX provides `\pdf<cmd>`.

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

**Expandibility:** 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

`\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.

`\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⟩}`.

`\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 0.70.0 `\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 0.70.0, revision 4167 of LuaTeX.

Hints for usage:

- `\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.

### 1.2.1 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<sup>A</sup>T<sub>E</sub>X:

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

### 1.2.2 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<sup>A</sup>T<sub>E</sub>X.

```
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@pdfdoccmds.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{pdfdoccmds}{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 [2011/04/16 v0.15 Utilities of pdfTeX for LuaTeX (H0)]%

```

## 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\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{pdf texcmds}{LuaTeX not detected}%
153   \def\pdf texcmds@nopdftex{%
154     \@PackageInfoNoLine{pdf texcmds}{pdfTeX >= 1.30 not detected}%
155     \let\pdf texcmds@nopdftex\relax
156   }%
157   \def\pdf texcmds@temp#1{%
158     \begingroup\expandafter\expandafter\expandafter\endgroup
159     \expandafter\ifx\csname pdf#1\endcsname\relax
160       \pdf texcmds@nopdftex
161     \else
162       \expandafter\def\csname pdf@#1\endcsname
163       \expandafter##\expandafter{%
164         \csname pdf#1\endcsname
165       }%
166     \fi
167   }%
168   \pdf texcmds@temp{strcmp}%
169   \pdf texcmds@temp{escapehex}%
170   \let\pdf@escapehexnative\pdf@escapehex
171   \pdf texcmds@temp{unescapehex}%
172   \let\pdf@unescapehexnative\pdf@unescapehex
173   \pdf texcmds@temp{escapestring}%
174   \pdf texcmds@temp{escapename}%
175   \pdf texcmds@temp{filesize}%
176   \pdf texcmds@temp{filemoddate}%
177   \begingroup\expandafter\expandafter\expandafter\endgroup
178   \expandafter\ifx\csname pdfshellescape\endcsname\relax
179     \pdf texcmds@nopdftex
180     \ltx@ifundefined{pdf texversion}{%
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 \fi

```

## 2.5 \pdf@primitive, \pdf@ifprimitive

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

X<sub>Y</sub>TeX provides them under the name `\primitive` and `\ifprimitive`. LuaTeX knows both name variants, but they have possibly to be enabled first (`tex.enableprimitives`).

Depending on the format TeX Live uses a prefix `luatex`.

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

### 2.5.1 Using LuaTeX's `tex.enableprimitives`

```

215 \ifluatex

\pdftexcmds@directlua

216 \ifnum\luatexversion<36 %
217 \def\pdftexcmds@directlua{\directlua0 }%
218 \else
219 \let\pdftexcmds@directlua\directlua
220 \fi

221 \begingroup
222 \newlinechar=10 %
223 \endlinechar=\newlinechar
224 \pdftexcmds@directlua{%
225   if tex.enableprimitives then
226     tex.enableprimitives(
227       'pdf@',

```

```

228         {'primitive', 'ifprimitive', 'pdfdraftmode'}
229     )
230     tex.enableprimitives('', {'luaescapestring'})
231 end
232 }%
233 \endgroup %
234 \fi

```

## 2.5.2 Trying various names to find the primitives

\pdftexcmds@strip@prefix

```

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

236 \def\pdftexcmds@temp#1#2#3{%
237   \begingroup\expandafter\expandafter\expandafter\endgroup
238   \expandafter\ifx\csname pdf@#1\endcsname\relax
239     \begingroup
240       \def\x{#3}%
241       \edef\x{\expandafter\pdftexcmds@strip@prefix\meaning\x}%
242       \escapechar=-1 %
243       \edef\y{\expandafter\meaning\csname#2\endcsname}%
244       \expandafter\endgroup
245       \ifx\x\y
246         \expandafter\let\csname pdf@#1\expandafter\endcsname
247         \csname #2\endcsname
248       \fi
249     \fi
250 }

```

\pdf@primitive

```

251 \pdftexcmds@temp{primitive}{pdfprimitive}{pdfprimitive}% pdfTeX, LuaTeX
252 \pdftexcmds@temp{primitive}{primitive}{primitive}% XeTeX
253 \pdftexcmds@temp{primitive}{luatexpprimitive}{pdfprimitive}% LuaTeX
254 \pdftexcmds@temp{primitive}{luatexpdfprimitive}{pdfprimitive}% LuaTeX

```

\pdf@ifprimitive

```

255 \pdftexcmds@temp{ifprimitive}{ifpdfprimitive}{ifpdfprimitive}% pdfTeX, LuaTeX
256 \pdftexcmds@temp{ifprimitive}{ifprimitive}{ifprimitive}% XeTeX
257 \pdftexcmds@temp{ifprimitive}{luatexpifprimitive}{ifpdfprimitive}% LuaTeX
258 \pdftexcmds@temp{ifprimitive}{luatexpifpdfprimitive}{ifpdfprimitive}% LuaTeX

```

Disable broken \pdfprimitive.

```

259 \begingroup
260   \expandafter\ifx\csname pdf@primitive\endcsname\relax
261   \else
262     \expandafter\ifx\csname pdftexversion\endcsname\relax
263     \else
264       \ifnum\pdftexversion=140 %
265         \expandafter\ifx\csname pdftexrevision\endcsname\relax
266         \else
267           \ifnum\pdftexrevision<4 %
268           \endgroup
269           \let\pdf@primitive\@undefined
270           \@PackageInfoNoLine{pdftexcmds}{%
271             \string\pdf@primitive disabled, because\MessageBreak
272             \string\pdfprimitive\space is broken until pdfTeX 1.40.4%
273           }%
274         \begingroup
275         \fi
276       \fi
277     \fi

```

```

278 \fi
279 \fi
280 \endgroup

```

### 2.5.3 Result

```

281 \begingroup
282 \@PackageInfoNoLine{pdftexcmds}{%
283   \string\pdf@primitive\space is %
284   \expandafter\ifx\csname pdf@primitive\endcsname\relax not \fi
285   available%
286 }%
287 \@PackageInfoNoLine{pdftexcmds}{%
288   \string\pdf@ifprimitive\space is %
289   \expandafter\ifx\csname pdf@ifprimitive\endcsname\relax not \fi
290   available%
291 }%
292 \endgroup

```

## 2.6 X<sub>Y</sub>T<sub>E</sub>X

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

```

293 \def\pdftexcmds@temp#1{%
294   \begingroup\expandafter\expandafter\expandafter\endgroup
295   \expandafter\ifx\csname pdf@#1\endcsname\relax
296     \begingroup
297     \escapechar=-1 %
298     \edef\x{\expandafter\meaning\csname#1\endcsname}%
299     \def\y{#1}%
300     \def\z##1->{}%
301     \edef\y{\expandafter\z\meaning\y}%
302   \expandafter\endgroup
303   \ifx\x\y
304     \expandafter\def\csname pdf@#1\expandafter\endcsname
305     \expandafter{%
306       \csname#1\endcsname
307     }%
308   \fi
309 \fi
310 }%
311 \pdftexcmds@temp{shellescape}%
312 \pdftexcmds@temp{strcmp}%

```

## 2.7 \pdf@isprimitive

```

313 \def\pdf@isprimitive{%
314   \begingroup\expandafter\expandafter\expandafter\endgroup
315   \expandafter\ifx\csname pdf@strcmp\endcsname\relax
316     \long\def\pdf@isprimitive##1{%
317       \expandafter\pdftexcmds@isprimitive\expandafter{\meaning##1}%
318     }%
319     \long\def\pdftexcmds@isprimitive##1##2{%
320       \expandafter\pdftexcmds@@isprimitive\expandafter{\string##2}{##1}%
321     }%
322     \def\pdftexcmds@@isprimitive##1##2{%
323       \ifnum0\pdftexcmds@equal##1\delimiter##2\delimiter=1 %
324       \expandafter\ltx@firstoftwo
325     \else
326       \expandafter\ltx@secondoftwo
327     \fi
328   }%
329   \def\pdftexcmds@equal##1##2\delimiter##3##4\delimiter{%
330     \if##1##3%

```

```

331     \ifx\relax##2##4\relax
332     1%
333     \else
334     \ifx\relax##2\relax
335     \else
336     \ifx\relax##4\relax
337     \else
338     \pdftexcmds@equalcont{##2}{##4}%
339     \fi
340     \fi
341     \fi
342     \fi
343     }%
344     \def\pdftexcmds@equalcont##1{%
345     \def\pdftexcmds@equalcont####1####2##1##1##1##1{%
346     ##1##1##1##1%
347     \pdftexcmds@equal####1\delimiter####2\delimiter
348     }%
349     }%
350     \expandafter\pdftexcmds@equalcont\csname fi\endcsname
351 \else
352     \long\def\pdf@isprimitive##1##2{%
353     \ifnum\pdf@strcmp{\meaning##1}{\string##2}=0 %
354     \expandafter\ltx@firstoftwo
355     \else
356     \expandafter\ltx@secondoftwo
357     \fi
358     }%
359     \fi
360 }
361 \ifluatex
362 \else
363     \pdf@isprimitive
364 \fi

```

## 2.8 \pdf@draftmode

```

365 \let\pdftexcmds@temp\ltx@zero %
366 \ltx@ifundefined{pdfdraftmode}{%
367     \@PackageInfoNoLine{pdftexcmds}{\ltx@backslashchar pdfdraftmode not found}%
368 }{%
369     \ifpdf
370     \let\pdftexcmds@temp\ltx@one
371     \@PackageInfoNoLine{pdftexcmds}{\ltx@backslashchar pdfdraftmode found}%
372     \else
373     \@PackageInfoNoLine{pdftexcmds}{%
374     \ltx@backslashchar pdfdraftmode is ignored in DVI mode%
375     }%
376     \fi
377 }
378 \ifcase\pdftexcmds@temp

```

\pdf@draftmode

```

379     \let\pdf@draftmode\ltx@zero

```

\pdf@ifdraftmode

```

380     \let\pdf@ifdraftmode\ltx@secondoftwo

```

\pdftexcmds@setdraftmode

```

381     \def\pdftexcmds@setdraftmode#1{%
382 \else

```

```

\pdftexcmds@draftmode

383 \let\pdftexcmds@draftmode\pdfdraftmode

\pdf@ifdraftmode

384 \def\pdf@ifdraftmode{%
385 \ifnum\pdftexcmds@draftmode=\ltx@one
386 \expandafter\ltx@firstoftwo
387 \else
388 \expandafter\ltx@secondoftwo
389 \fi
390 }%

\pdf@draftmode

391 \def\pdf@draftmode{%
392 \ifnum\pdftexcmds@draftmode=\ltx@one
393 \expandafter\ltx@one
394 \else
395 \expandafter\ltx@zero
396 \fi
397 }%

\pdftexcmds@setdraftmode

398 \def\pdftexcmds@setdraftmode#1{%
399 \pdftexcmds@draftmode=#1\relax
400 }%

401 \fi

\pdf@setdraftmode

402 \def\pdf@setdraftmode#1{%
403 \begingroup
404 \count\ltx@cclv=#1\relax
405 \edef\x{\endgroup
406 \noexpand\pdftexcmds@@setdraftmode{\the\count\ltx@cclv}}%
407 }%
408 \x
409 }

\pdftexcmds@@setdraftmode

410 \def\pdftexcmds@@setdraftmode#1{%
411 \ifcase#1 %
412 \pdftexcmds@setdraftmode{#1}%
413 \or
414 \pdftexcmds@setdraftmode{#1}%
415 \else
416 \@PackageWarning{pdftexcmds}{%
417 \string\pdf@setdraftmode: Ignoring\MessageBreak
418 invalid value ‘#1’%
419 }%
420 \fi
421 }

```

## 2.9 Load Lua module

```

422 \ifluatex
423 \else
424 \expandafter\pdftexcmds@AtEnd
425 \fi%

426 \begingroup\expandafter\expandafter\expandafter\endgroup
427 \expandafter\ifx\csname RequirePackage\endcsname\relax
428 \def\TMP@RequirePackage#1[#2]{%

```

```

429 \begingroup\expandafter\expandafter\expandafter\endgroup
430 \expandafter\ifx\csname ver@#1.sty\endcsname\relax
431 \input #1.sty\relax
432 \fi
433 }%
434 \TMP@RequirePackage{luatex-loader}[2009/04/10]%
435 \else
436 \RequirePackage{luatex-loader}[2009/04/10]%
437 \fi
438 \pdfdocmds@directlua{%
439 require("oberdiek.pdfdocmds")%
440 }
441 \begingroup
442 \def\x{2011/04/16 v0.15}%
443 \ltx@onelevel@sanitize\x
444 \edef\y{%
445 \pdfdocmds@directlua{%
446 if oberdiek.pdfdocmds.getversion then %
447 oberdiek.pdfdocmds.getversion()%
448 end%
449 }%
450 }%
451 \ifx\x\y
452 \else
453 \@PackageError{pdfdocmds}{%
454 Wrong version of lua module.\MessageBreak
455 Package version: \x\MessageBreak
456 Lua module: \y
457 }\@ehc
458 \fi
459 \endgroup

```

## 2.10 Lua functions

\pdfdocmds@toks

```

460 \begingroup\expandafter\expandafter\expandafter\endgroup
461 \expandafter\ifx\csname newtoks\endcsname\relax
462 \toksdef\pdfdocmds@toks=0 %
463 \else
464 \csname newtoks\endcsname\pdfdocmds@toks
465 \fi

```

\pdfdocmds@Patch

```

466 \def\pdfdocmds@Patch{0}
467 \ifnum\luatexversion>40 %
468 \ifnum\luatexversion<66 %
469 \def\pdfdocmds@Patch{1}%
470 \fi
471 \fi

472 \ifcase\pdfdocmds@Patch
473 \catcode'\&=14 %
474 \else
475 \catcode'\&=9 %

```

\pdfdocmds@PatchDecode

```

476 \def\pdfdocmds@PatchDecode#1\@nil{%
477 \pdfdocmds@DecodeA#1^^A^^A\@nil}%
478 }%

```

\pdfdocmds@DecodeA

```

479 \def\pdfdocmds@DecodeA#1^^A^^A#2\@nil#3{%

```

```

480 \ifx\relax#2\relax
481 \ltx@ReturnAfterElseFi{%
482 \pdftexcmds@DecodeB#3#1^^A^^B\@nil{}}%
483 }%
484 \else
485 \ltx@ReturnAfterFi{%
486 \pdftexcmds@DecodeA#2\@nil{#3#1^^@}%
487 }%
488 \fi
489 }%

```

\pdftexcmds@DecodeB

```

490 \def\pdftexcmds@DecodeB#1^^A^^B#2\@nil#3{%
491 \ifx\relax#2\relax%
492 \ltx@ReturnAfterElseFi{%
493 \ltx@zero
494 #3#1%
495 }%
496 \else
497 \ltx@ReturnAfterFi{%
498 \pdftexcmds@DecodeB#2\@nil{#3#1^^A}%
499 }%
500 \fi
501 }%

```

```

502 \fi
503 \ifnum\luatexversion<36 %
504 \else
505 \catcode'\0=9 %
506 \fi

```

\pdf@strcmp

```

507 \long\def\pdf@strcmp#1#2{%
508 \directlua0{%
509 oberdiek.pdf texcmds.strptime("\luaescapestring{#1}",%
510 "\luaescapestring{#2}")%
511 }%
512 }%

```

513 \pdf@isprimitive

\pdf@escapehex

```

514 \long\def\pdf@escapehex#1{%
515 \directlua0{%
516 oberdiek.pdf texcmds.escapehex("\luaescapestring{#1}", "byte")%
517 }%
518 }%

```

\pdf@escapehexnative

```

519 \long\def\pdf@escapehexnative#1{%
520 \directlua0{%
521 oberdiek.pdf texcmds.escapehex("\luaescapestring{#1}")%
522 }%
523 }%

```

\pdf@unescapehex

```

524 \def\pdf@unescapehex#1{%
525 & \romannumeral\expandafter\pdftexcmds@PatchDecode
526 \the\expandafter\pdftexcmds@toks
527 \directlua0{%
528 oberdiek.pdf texcmds.toks="pdf texcmds@toks"%

```

```

529 oberdiek.pdfTeXcmds.unescapehex("\luaescapestring{#1}", "byte", \pdfTeXcmds@Patch)%
530 }%
531 & \@nil
532 }%

\pdf@unescapehexnative

533 \def\pdf@unescapehexnative#1{%
534 & \romannumeral\expandafter\pdfTeXcmds@PatchDecode
535 \the\expandafter\pdfTeXcmds@toks
536 \directlua0{%
537 oberdiek.pdfTeXcmds.toks="pdfTeXcmds@toks"%
538 oberdiek.pdfTeXcmds.unescapehex("\luaescapestring{#1}", \pdfTeXcmds@Patch)%
539 }%
540 & \@nil
541 }%

\pdf@escapestring

542 \long\def\pdf@escapestring#1{%
543 \directlua0{%
544 oberdiek.pdfTeXcmds.escapestring("\luaescapestring{#1}", "byte")%
545 }%
546 }

\pdf@escapename

547 \long\def\pdf@escapename#1{%
548 \directlua0{%
549 oberdiek.pdfTeXcmds.escapename("\luaescapestring{#1}", "byte")%
550 }%
551 }

\pdf@escapenamenative

552 \long\def\pdf@escapenamenative#1{%
553 \directlua0{%
554 oberdiek.pdfTeXcmds.escapename("\luaescapestring{#1}")%
555 }%
556 }

\pdf@filesize

557 \def\pdf@filesize#1{%
558 \directlua0{%
559 oberdiek.pdfTeXcmds.filesize("\luaescapestring{#1}")%
560 }%
561 }

\pdf@filemoddate

562 \def\pdf@filemoddate#1{%
563 \directlua0{%
564 oberdiek.pdfTeXcmds.filemoddate("\luaescapestring{#1}")%
565 }%
566 }

\pdf@filedump

567 \def\pdf@filedump#1#2#3{%
568 \directlua0{%
569 oberdiek.pdfTeXcmds.filedump("\luaescapestring{\number#1}",%
570 "\luaescapestring{\number#2}",%
571 "\luaescapestring{#3}")%
572 }%
573 }%

```



```

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

\pdf@mdfivesumnative
579 \long\def\pdf@mdfivesumnative#1{%
580   \directlua0{%
581     oberdiek.pdf_texcmds.mdfivesum("\luaescapestring{#1}")%
582   }%
583 }%

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

\pdf@shellescape
589 \ifnum\luatexversion<70 %
590 \else
591   \def\pdf@shellescape{%
592     \directlua0{%
593       oberdiek.pdf_texcmds.shellescape()%
594     }%
595   }%
596 \fi

\pdf@system
597 \def\pdf@system#1{%
598   \directlua0{%
599     oberdiek.pdf_texcmds.system("\luaescapestring{#1}")%
600   }%
601 }

\pdf@lastsystemstatus
602 \def\pdf@lastsystemstatus{%
603   \directlua0{%
604     oberdiek.pdf_texcmds.lastsystemstatus()%
605   }%
606 }

\pdf@lastsystemexit
607 \def\pdf@lastsystemexit{%
608   \directlua0{%
609     oberdiek.pdf_texcmds.lastsystemexit()%
610   }%
611 }

612 \catcode'\0=12 %

\pdf@pipe Check availability of io.popen first.
613 \ifnum0%
614   \pdf_texcmds@directlua{%
615     if io.popen then %
616       tex.write("1")%
617     end%
618   }%

```

```

619     =1 %
620     \def\pdf@pipe#1{%
621 &     \romannumeral\expandafter\pdf texcmds@PatchDecode
622     \the\expandafter\pdf texcmds@toks
623     \pdf texcmds@directlua{%
624         oberdiek.pdf texcmds.toks="pdf texcmds@toks"%
625         oberdiek.pdf texcmds.pipe("\luaescapestring{#1}", \pdf texcmds@Patch)%
626     }%
627 &     \@nil
628     }%
629 \fi

630 \pdf texcmds@AtEnd%
631 \</package>

```

## 2.11 Lua module

```

632 < *lua >

633 module("oberdiek.pdf texcmds", package.seeall)
634 local systemexitstatus
635 function getversion()
636     tex.write("2011/04/16 v0.15")
637 end
638 function strcmp(A, B)
639     if A == B then
640         tex.write("0")
641     elseif A < B then
642         tex.write("-1")
643     else
644         tex.write("1")
645     end
646 end
647 local function utf8_to_byte(str)
648     local i = 0
649     local n = string.len(str)
650     local t = {}
651     while i < n do
652         i = i + 1
653         local a = string.byte(str, i)
654         if a < 128 then
655             table.insert(t, string.char(a))
656         else
657             if a >= 192 and i < n then
658                 i = i + 1
659                 local b = string.byte(str, i)
660                 if b < 128 or b >= 192 then
661                     i = i - 1
662                 elseif a == 194 then
663                     table.insert(t, string.char(b))
664                 elseif a == 195 then
665                     table.insert(t, string.char(b + 64))
666                 end
667             end
668         end
669     end
670     return table.concat(t)
671 end
672 function escapehex(str, mode)
673     if mode == "byte" then
674         str = utf8_to_byte(str)
675     end
676     tex.write((string.gsub(str, ".",

```

```

677     function (ch)
678         return string.format("%02X", string.byte(ch))
679     end
680 )))
681 end

```

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

```

682 function unescapehex(str, mode, patch)
683     local a = 0
684     local first = true
685     local result = {}
686     for i = 1, string.len(str), 1 do
687         local ch = string.byte(str, i)
688         if ch >= 48 and ch <= 57 then
689             ch = ch - 48
690         elseif ch >= 65 and ch <= 70 then
691             ch = ch - 55
692         elseif ch >= 97 and ch <= 102 then
693             ch = ch - 87
694         else
695             ch = nil
696         end
697         if ch then
698             if first then
699                 a = ch * 16
700                 first = false
701             else
702                 table.insert(result, a + ch)
703                 first = true
704             end
705         end
706     end
707     if not first then
708         table.insert(result, a)
709     end
710     if patch == 1 then
711         local temp = {}
712         for i, a in ipairs(result) do
713             if a == 0 then
714                 table.insert(temp, 1)
715                 table.insert(temp, 1)
716             else
717                 if a == 1 then
718                     table.insert(temp, 1)
719                     table.insert(temp, 2)
720                 else
721                     table.insert(temp, a)
722                 end
723             end
724         end
725         result = temp
726     end
727     if mode == "byte" then
728         local utf8 = {}
729         for i, a in ipairs(result) do
730             if a < 128 then
731                 table.insert(utf8, a)
732             else
733                 if a < 192 then
734                     table.insert(utf8, 194)
735                     a = a - 128
736                 else

```

```

737         table.insert(utf8, 195)
738         a = a - 192
739     end
740     table.insert(utf8, a + 128)
741 end
742 end
743 result = utf8
744 end
745 tex.settoks(toks, string.char(unpack(result)))
746 end

```

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

```

747 function escapestring(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       local b = string.byte(ch)
754       if b < 33 or b > 126 then
755         return string.format("\\%.3o", b)
756       end
757       if b == 40 or b == 41 or b == 92 then
758         return "\\\" .. ch
759       end

```

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

```

760       return nil
761     end
762   )))
763 end

```

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

```

764 function escapename(str, mode)
765   if mode == "byte" then
766     str = utf8_to_byte(str)
767   end
768   tex.write((string.gsub(str, ".",
769     function (ch)
770       local b = string.byte(ch)
771       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.

```

772       return ""
773     end
774     if b <= 32 or b >= 127
775       or b == 35 or b == 37 or b == 40 or b == 41
776       or b == 47 or b == 60 or b == 62 or b == 91
777       or b == 93 or b == 123 or b == 125 then
778       return string.format("#%.2X", b)
779     else

```

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

```

780       return nil
781     end
782   end
783   )))
784 end
785 function filesize(filename)
786   local foundfile = kpse.find_file(filename, "tex", true)
787   if foundfile then
788     local size = lfs.attributes(foundfile, "size")
789     if size then
790       tex.write(size)
791     end

```

```

792 end
793 end

```

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

```

794 function filemoddate(filename)
795   local foundfile = kpse.find_file(filename, "tex", true)
796   if foundfile then
797     local date = lfs.attributes(foundfile, "modification")
798     if date then
799       local d = os.date("!*t", date)
800       if d.sec >= 60 then
801         d.sec = 59
802       end
803       local u = os.date("!*t", date)
804       local off = 60 * (d.hour - u.hour) + d.min - u.min
805       if d.year ~= u.year then
806         if d.year > u.year then
807           off = off + 1440
808         else
809           off = off - 1440
810         end
811       elseif d.yday ~= u.yday then
812         if d.yday > u.yday then
813           off = off + 1440
814         else
815           off = off - 1440
816         end
817       end
818       local timezone
819       if off == 0 then
820         timezone = "Z"
821       else
822         local hours = math.floor(off / 60)
823         local mins = math.abs(off - hours * 60)
824         timezone = string.format("%+03d'%02d'", hours, mins)
825       end
826       tex.write(string.format("D:%04d%02d%02d%02d%02d%s",
827         d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
828     end
829   end
830 end
831 function filedump(offset, length, filename)
832   length = tonumber(length)
833   if length and length > 0 then
834     local foundfile = kpse.find_file(filename, "tex", true)
835     if foundfile then
836       offset = tonumber(offset)
837       if not offset then
838         offset = 0
839       end
840       local filehandle = io.open(foundfile, "r")
841       if filehandle then
842         if offset > 0 then
843           filehandle:seek("set", offset)
844         end
845         local dump = filehandle:read(length)
846         escapehex(dump)
847       end
848     end
849   end
850 end
851 function mdffivesum(str, mode)
852   if mode == "byte" then

```

```

853     str = utf8_to_byte(str)
854 end
855 escapehex(md5.sum(str))
856 end
857 function filemdfivesum(filename)
858     local foundfile = kpse.find_file(filename, "tex", true)
859     if foundfile then
860         local filehandle = io.open(foundfile, "r")
861         if filehandle then
862             local contents = filehandle:read("*a")
863             escapehex(md5.sum(contents))
864         end
865     end
866 end
867 function shellescape()
868     if os.execute then
869         if status
870             and status.luatex_version
871             and status.luatex_version >= 70 then
872             tex.write(os.execute())
873         else
874             local result = os.execute()
875             if result == 0 then
876                 tex.write("0")
877             else
878                 if result == nil then
879                     tex.write("0")
880                 else
881                     tex.write("1")
882                 end
883             end
884         end
885     else
886         tex.write("0")
887     end
888 end
889 function system(cmdline)
890     systemexitstatus = nil
891     texio.write_nl("log", "system(" .. cmdline .. ") ")
892     if os.execute then
893         texio.write("log", "executed.")
894         systemexitstatus = os.execute(cmdline)
895     else
896         texio.write("log", "disabled.")
897     end
898 end
899 function lastsystemstatus()
900     local result = tonumber(systemexitstatus)
901     if result then
902         local x = math.floor(result / 256)
903         tex.write(result - 256 * math.floor(result / 256))
904     end
905 end
906 function lastsystemexit()
907     local result = tonumber(systemexitstatus)
908     if result then
909         tex.write(math.floor(result / 256))
910     end
911 end
912 function pipe(cmdline, patch)
913     local result
914     systemexitstatus = nil

```

```

915 texio.write_nl("log", "pipe(" .. cmdline ..") ")
916 if io.popen then
917     texio.write("log", "executed.")
918     local handle = io.popen(cmdline, "r")
919     if handle then
920         result = handle:read("*a")
921         handle:close()
922     end
923 else
924     texio.write("log", "disabled.")
925 end
926 if result then
927     if patch == 1 then
928         local temp = {}
929         for i, a in ipairs(result) do
930             if a == 0 then
931                 table.insert(temp, 1)
932                 table.insert(temp, 1)
933             else
934                 if a == 1 then
935                     table.insert(temp, 1)
936                     table.insert(temp, 2)
937                 else
938                     table.insert(temp, a)
939                 end
940             end
941         end
942         result = temp
943     end
944     tex.settoks(toks, result)
945 else
946     tex.settoks(toks, "")
947 end
948 end
949  $\langle$ /lua $\rangle$ 

```

## 3 Test

### 3.1 Catcode checks for loading

```

950  $\langle$ *test1 $\rangle$ 
951 \catcode'\{=1 %
952 \catcode'\}=2 %
953 \catcode'\#=6 %
954 \catcode'\@=11 %
955 \expandafter\ifx\csname count@\endcsname\relax
956   \countdef\count@=255 %
957 \fi
958 \expandafter\ifx\csname @gobble\endcsname\relax
959   \long\def\@gobble#1{}%
960 \fi
961 \expandafter\ifx\csname @firstofone\endcsname\relax
962   \long\def\@firstofone#1{#1}%
963 \fi
964 \expandafter\ifx\csname loop\endcsname\relax
965   \expandafter\@firstofone
966 \else
967   \expandafter\@gobble
968 \fi
969 {%
970   \def\loop#1\repeat{%
971     \def\body{#1}%

```

```

972 \iterate
973 }%
974 \def\iterate{%
975 \body
976 \let\next\iterate
977 \else
978 \let\next\relax
979 \fi
980 \next
981 }%
982 \let\repeat=\fi
983 }%
984 \def\RestoreCatcodes{}
985 \count@=0 %
986 \loop
987 \edef\RestoreCatcodes{%
988 \RestoreCatcodes
989 \catcode\the\count@=\the\catcode\count@\relax
990 }%
991 \ifnum\count@<255 %
992 \advance\count@ 1 %
993 \repeat
994
995 \def\RangeCatcodeInvalid#1#2{%
996 \count@=#1\relax
997 \loop
998 \catcode\count@=15 %
999 \ifnum\count@<#2\relax
1000 \advance\count@ 1 %
1001 \repeat
1002 }
1003 \def\RangeCatcodeCheck#1#2#3{%
1004 \count@=#1\relax
1005 \loop
1006 \ifnum#3=\catcode\count@
1007 \else
1008 \errmessage{%
1009 Character \the\count@\space
1010 with wrong catcode \the\catcode\count@\space
1011 instead of \number#3%
1012 }%
1013 \fi
1014 \ifnum\count@<#2\relax
1015 \advance\count@ 1 %
1016 \repeat
1017 }
1018 \def\space{ }
1019 \expandafter\ifx\csname LoadCommand\endcsname\relax
1020 \def\LoadCommand{\input pdftexcmds.sty\relax}%
1021 \fi
1022 \def\Test{%
1023 \RangeCatcodeInvalid{0}{47}%
1024 \RangeCatcodeInvalid{58}{64}%
1025 \RangeCatcodeInvalid{91}{96}%
1026 \RangeCatcodeInvalid{123}{255}%
1027 \catcode'\@=12 %
1028 \catcode'\=0 %
1029 \catcode'\%=14 %
1030 \LoadCommand
1031 \RangeCatcodeCheck{0}{36}{15}%
1032 \RangeCatcodeCheck{37}{37}{14}%
1033 \RangeCatcodeCheck{38}{47}{15}%

```



```

1034 \RangeCatcodeCheck{48}{57}{12}%
1035 \RangeCatcodeCheck{58}{63}{15}%
1036 \RangeCatcodeCheck{64}{64}{12}%
1037 \RangeCatcodeCheck{65}{90}{11}%
1038 \RangeCatcodeCheck{91}{91}{15}%
1039 \RangeCatcodeCheck{92}{92}{0}%
1040 \RangeCatcodeCheck{93}{96}{15}%
1041 \RangeCatcodeCheck{97}{122}{11}%
1042 \RangeCatcodeCheck{123}{255}{15}%
1043 \RestoreCatcodes
1044 }
1045 \Test
1046 \csname @@end\endcsname
1047 \end
1048 </test1>

```

### 3.2 Test for \pdf@isprimitive

```

1049 <*test2>
1050 \catcode'\{=1 %
1051 \catcode'\}=2 %
1052 \catcode'\#=6 %
1053 \catcode'\@=11 %
1054 \input pdftexcmds.sty\relax
1055 \def\msg#1{%
1056   \begingroup
1057     \escapechar=92 %
1058     \immediate\write16{#1}%
1059   \endgroup
1060 }
1061 \long\def\test#1#2#3#4{%
1062   \begingroup
1063     #4%
1064     \def\str{%
1065       Test \string\pdf@isprimitive
1066       {\string #1}{\string #2}{...}: %
1067     }%
1068     \pdf@isprimitive{#1}{#2}{%
1069       \ifx#3Y%
1070         \msg{\str true ==> OK.}%
1071       \else
1072         \errmessage{\str false ==> FAILED}%
1073       \fi
1074     }{%
1075       \ifx#3Y%
1076         \errmessage{\str true ==> FAILED}%
1077       \else
1078         \msg{\str false ==> OK.}%
1079       \fi
1080     }%
1081   \endgroup
1082 }
1083 \test\relax\relax Y{}
1084 \test\foobar\relax Y{\let\foobar\relax}
1085 \test\foobar\relax N{}
1086 \test\hbox\hbox Y{}
1087 \test\foobar@hbox\hbox Y{\let\foobar@hbox\hbox}
1088 \test\if\if Y{}
1089 \test\if\ifx N{}
1090 \test\ifx\if N{}
1091 \test\par\par Y{}
1092 \test\hbox\par N{}
1093 \test\par\hbox N{}

```

```

1094 \csname @@end\endcsname\end
1095 </test2>

```

### 3.3 Test for \pdf@shellescape

```

1096 <*test-shell>
1097 \catcode'\{=1 %
1098 \catcode'\}=2 %
1099 \catcode'\#=6 %
1100 \catcode'\@=11 %
1101 \input pdftexcmds.sty\relax
1102 \def\msg#{\immediate\write16{
1103 \ifx\pdf@shellescape\@undefined
1104   \msg{SHELL=U}%
1105 \else
1106   \msg{SHELL=\number\pdf@shellescape}%
1107 \fi
1108 \ifx\expected\@undefined
1109 \else
1110   \ifx\expected\relax
1111     \msg{EXPECTED=U}%
1112     \ifx\pdf@shellescape\@undefined
1113       \msg{OK}%
1114     \else
1115       \errmessage{Failed}%
1116     \fi
1117   \else
1118     \msg{EXPECTED=\number\expected}%
1119     \ifnum\pdf@shellescape=\expected\relax
1120       \msg{OK}%
1121     \else
1122       \errmessage{Failed}%
1123     \fi
1124   \fi
1125 \fi
1126 \csname @@end\endcsname\end
1127 </test-shell>

```

### 3.4 Test for escape functions

```

1128 <*test-escape>
1129 \catcode'\{=1 %
1130 \catcode'\}=2 %
1131 \catcode'\#=6 %
1132 \catcode'\^=7 %
1133 \catcode'\@=11 %
1134 \errorcontextlines=1000 %
1135 \input pdftexcmds.sty\relax
1136 \def\msg#1{%
1137   \begingroup
1138     \escapechar=92 %
1139     \immediate\write16{#1}%
1140   \endgroup
1141 }
1142 \begingroup
1143   \catcode'\@=11 %
1144   \countdef\count@=255 %
1145   \def\space{ }%
1146   \long\def\@whilenum#1\do #2{%
1147     \ifnum #1\relax
1148       #2\relax
1149     \@iwhilenum{#1\relax#2\relax}%
1150   \fi
1151 }%

```

```

1152 \long\def\@iwhilenum#1{%
1153     \ifnum #1%
1154         \expandafter\@iwhilenum
1155     \else
1156         \expandafter\ltx@gobble
1157     \fi
1158     {#1}%
1159 }%
1160 \gdef\AllBytes{%
1161     \count@=0 %
1162     \catcode0=12 %
1163     \@whilenum\count@<256 \do{%
1164         \lccode0=\count@
1165         \ifnum\count@=32 %
1166             \xdef\AllBytes{\AllBytes\space}%
1167         \else
1168             \lowercase{%
1169                 \xdef\AllBytes{\AllBytes^^@}%
1170             }%
1171         \fi
1172         \advance\count@ by 1 %
1173     }%
1174 \endgroup

1175 \def\AllBytesHex{%
1176     000102030405060708090A0B0C0D0E0F%
1177     101112131415161718191A1B1C1D1E1F%
1178     202122232425262728292A2B2C2D2E2F%
1179     303132333435363738393A3B3C3D3E3F%
1180     404142434445464748494A4B4C4D4E4F%
1181     505152535455565758595A5B5C5D5E5F%
1182     606162636465666768696A6B6C6D6E6F%
1183     707172737475767778797A7B7C7D7E7F%
1184     808182838485868788898A8B8C8D8E8F%
1185     909192939495969798999A9B9C9D9E9F%
1186     A0A1A2A3A4A5A6A7A8A9AABACADA EAF%
1187     B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF%
1188     C0C1C2C3C4C5C6C7C8C9CACBCCDCECF%
1189     D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
1190     E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
1191     F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF%
1192 }
1193 \ltx@onelevel@sanitize\AllBytesHex
1194 \expandafter\lowercase\expandafter{%
1195     \expandafter\def\expandafter\AllBytesHexLC
1196         \expandafter{\AllBytesHex}%
1197 }
1198 \begingroup
1199     \catcode'\#=12 %
1200     \xdef\AllBytesName{%
1201         #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
1202         #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
1203         #20!"#23$#25&'#28#29*+,-.#2F%
1204         0123456789:;#3C=#3E?%
1205         @ABCDEFGHIJKLMNO%
1206         PQRSTUVWXYZ#5B\ltx@backslashchar#5D^_%
1207         'abcdefghijklmno%
1208         pqrstuvwxyz7B|7D\string~#7F%
1209         #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
1210         #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
1211         #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
1212         #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
1213         #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%

```

```

1214     #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
1215     #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
1216     #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
1217 }%
1218 \endgroup
1219 \ltx@onelevel@sanitize\AllBytesName
1220 \edef\AllBytesFromName{\expandafter\ltx@gobble\AllBytes}
1221 \begingroup
1222   \def\|{|}%
1223   \edef\%{\ltx@percentchar}%
1224   \catcode'\|=0 %
1225   \catcode'\#=12 %
1226   \catcode'\~=12 %
1227   \catcode'\|=12 %
1228   \xdef\AllBytesString{%
1229     \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
1230     \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
1231     \040!"#$%&'(\)*+,-./%
1232     0123456789:;<=>?%
1233     @ABCDEFGHIJKLMNO%
1234     PQRSTUVWXYZ[\]^_%
1235     'abcdefghijklmno%
1236     pqrstuvwxyz{|}~\177%
1237     \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
1238     \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
1239     \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
1240     \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
1241     \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
1242     \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
1243     \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
1244     \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
1245 }%
1246 \endgroup
1247 \ltx@onelevel@sanitize\AllBytesString
1248 \def\Test#1#2#3{%
1249   \begingroup
1250     \expandafter\expandafter\expandafter\def
1251     \expandafter\expandafter\expandafter\TestResult
1252     \expandafter\expandafter\expandafter{%
1253       #1{#2}%
1254     }%
1255     \ifx\TestResult#3%
1256     \else
1257       \newlinechar=10 %
1258       \msg{Expect:^^J#3}%
1259       \msg{Result:^^J\TestResult}%
1260       \errmessage{\string#2 -\string#1-> \string#3}%
1261     \fi
1262   \endgroup
1263 }
1264 \def\test#1#2#3{%
1265   \edef\TestFrom{#2}%
1266   \edef\TestExpect{#3}%
1267   \ltx@onelevel@sanitize\TestExpect
1268   \Test#1\TestFrom\TestExpect
1269 }
1270 \test\pdf@unescapehex{74657374}{test}
1271 \begingroup
1272   \catcode0=12 %
1273   \catcode1=12 %
1274   \test\pdf@unescapehex{740074017400740174}{t^^@t^^@t^^@t^^@}%
1275 \endgroup

```

```

1276 \Test\pdf@escapehex\AllBytes\AllBytesHex
1277 \Test\pdf@unescapehex\AllBytesHex\AllBytes
1278 \Test\pdf@escapename\AllBytes\AllBytesName
1279 \Test\pdf@escapestring\AllBytes\AllBytesString
1280 \csname @@end\endcsname\end
1281 </test-escape>

```

## 4 Installation

### 4.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

[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<sub>E</sub>X 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<sub>E</sub>X:

```
tex pdftexcmds.dtx
```

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

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

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

<sup>1</sup><http://ftp.ctan.org/tex-archive/>

## 4.4 Refresh file name databases

If your  $\text{T}_{\text{E}}\text{X}$  distribution (te $\text{T}_{\text{E}}\text{X}$ , mik $\text{T}_{\text{E}}\text{X}$ , ...) relies on file name databases, you must refresh these. For example, te $\text{T}_{\text{E}}\text{X}$  users run `texhash` or `mktextlsr`.

## 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  $\text{\LaTeX}$ .** The `.dtx` chooses its action depending on the format:

**plain  $\text{T}_{\text{E}}\text{X}$ :** Run `docstrip` and extract the files.

**$\text{\LaTeX}$ :** Generate the documentation.

If you insist on using  $\text{\LaTeX}$  for `docstrip` (really, `docstrip` does not need  $\text{\LaTeX}$ ), 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 $\text{\LaTeX}$ :

```
pdflatex pdftexcmds.dtx
makeindex -s gind.ist pdftexcmds.idx
pdflatex pdftexcmds.dtx
makeindex -s gind.ist pdftexcmds.idx
pdflatex 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 Lua $\text{T}_{\text{E}}\text{X}$  0.36.

[2009/09/22 v0.5]

- `\pdf@primitive`, `\pdf@ifprimitive` added.
- `XgTeX`'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.

## 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

`\#` . . . 953, 1052, 1099, 1131, 1199, 1225<sub>31</sub>

<code>\%</code> .....	1029, 1223	1006, 1009, 1010, 1014, 1015,	
<code>\&amp;</code> .....	473, 475	1144, 1161, 1163, 1164, 1165, 1172	
<code>\(</code> .....	1231	<code>\countdef</code> .....	956, 1144
<code>\)</code> .....	1231	<code>\csname</code> .....	14, 21, 50, 66,
<code>\@</code> ...	954, 1027, 1053, 1100, 1133, 1143		76, 133, 136, 159, 162, 164, 178,
<code>\@PackageError</code> .....	453		196, 204, 238, 243, 246, 247,
<code>\@PackageInfoNoLine</code> .....	152,		260, 262, 265, 284, 289, 295,
	154, 270, 282, 287, 367, 371, 373		298, 304, 306, 315, 350, 427,
<code>\@PackageWarning</code> .....	416		430, 461, 464, 955, 958, 961,
<code>\@ehc</code> .....	457		964, 1019, 1046, 1094, 1126, 1280
<code>\@firstofone</code> .....	962, 965	<b>D</b>	
<code>\@gobble</code> .....	959, 967	<code>\delimiter</code> .....	323, 329, 347
<code>\@iwhilenum</code> .....	1149, 1152, 1154	<code>\directlua</code> .....	217,
<code>\@nil</code> .....	476, 477, 479,		219, 508, 515, 520, 527, 536,
	482, 486, 490, 498, 531, 540, 627		543, 548, 553, 558, 563, 568,
<code>\@undefined</code> ..	58, 269, 1103, 1108, 1112		575, 580, 585, 592, 598, 603, 608
<code>\@whilenum</code> .....	1146, 1163	<code>\do</code> .....	1146, 1163
<code>\ </code> .....	755, 758, 1028, 1227, 1234	<b>E</b>	
<code>\{</code> .....	951, 1050, 1097, 1129	<code>\empty</code> .....	17, 18
<code>\}</code> .....	952, 1051, 1098, 1130	<code>\end</code> .....	1047, 1094, 1126, 1280
<code>\^</code> .....	1132	<code>\endcsname</code> .....	14, 21, 50, 66,
<code>\ </code> .....	1222, 1224		76, 133, 136, 159, 162, 164, 178,
<code>\~</code> .....	1226		196, 204, 238, 243, 246, 247,
<b>Numbers</b>			260, 262, 265, 284, 289, 295,
<code>\0</code> .....	505, 612, 1229, 1230, 1231		298, 304, 306, 315, 350, 427,
<code>\1</code> .....	1236		430, 461, 464, 955, 958, 961,
<code>\2</code> .....	1237, 1238, 1239, 1240		964, 1019, 1046, 1094, 1126, 1280
<code>\3</code> .....	1241, 1242, 1243, 1244	<code>\endinput</code> .....	29, 129
<b>A</b>		<code>\endlinechar</code> ....	4, 35, 71, 77, 89, 223
<code>\advance</code> .....	992, 1000, 1015, 1172	<code>\errmessage</code> .....	
<code>\aftergroup</code> .....	29		1008, 1072, 1076, 1115, 1122, 1260
<code>\AllBytes</code> .....	1160, 1166,	<code>\errorcontextlines</code> .....	1134
	1169, 1220, 1276, 1277, 1278, 1279	<code>\escapechar</code> .....	
<code>\AllBytesFromName</code> .....	1220		128, 131, 242, 297, 1057, 1138
<code>\AllBytesHex</code> .....		<code>\expected</code> .....	1108, 1110, 1118, 1119
	1175, 1193, 1196, 1276, 1277	<b>F</b>	
<code>\AllBytesHexLC</code> .....	1195	<code>\foobar</code> .....	1084, 1085
<code>\AllBytesName</code> .....	1200, 1219, 1278	<code>\foobar@hbox</code> .....	1087
<code>\AllBytesString</code> .....	1247, 1279	<b>G</b>	
<b>B</b>		<code>\gdef</code> .....	1160
<code>\body</code> .....	971, 975	<b>H</b>	
<b>C</b>		<code>\hbox</code> .....	1086, 1087, 1092, 1093
<code>\catcode</code> ..	2, 3, 5, 6, 7, 8, 9, 10, 11,	<b>I</b>	
	12, 13, 33, 34, 36, 37, 38, 39, 40,	<code>\if</code> .....	1088, 1089, 1090
	41, 42, 43, 44, 45, 46, 47, 48, 49,	<code>\ifcase</code> .....	378, 411, 472
	69, 70, 72, 73, 74, 78, 79, 80, 81,	<code>\ifeof</code> .....	182, 183
	82, 83, 84, 87, 88, 90, 91, 92, 93,	<code>\ifluatex</code> .....	150, 215, 361, 422
	97, 99, 473, 475, 505, 612, 951,	<code>\ifnum</code> .....	182, 216, 264,
	952, 953, 954, 989, 998, 1006,		267, 323, 353, 385, 392, 467,
	1010, 1027, 1028, 1029, 1050,		468, 503, 589, 613, 991, 999,
	1051, 1052, 1053, 1097, 1098,		1006, 1014, 1119, 1147, 1153, 1165
	1099, 1100, 1129, 1130, 1131,	<code>\ifpdf</code> .....	369
	1132, 1133, 1143, 1162, 1199,	<code>\ifx</code> .....	15, 18, 21, 50,
	1224, 1225, 1226, 1227, 1272, 1273		58, 61, 133, 136, 159, 178, 196,
<code>\chardef</code> .....	184, 186		204, 238, 245, 260, 262, 265,
<code>\count</code> .....	404, 406		284, 289, 295, 303, 315, 330,
<code>\count@</code> .....	956, 985, 989, 991,		331, 334, 336, 427, 430, 451,
	992, 996, 998, 999, 1000, 1004,		



461, 480, 491, 955, 958, 961, 964, 1019, 1069, 1075, 1089, 1090, 1103, 1108, 1110, 1112, 1255	
\immediate	23, 52, 212, 1058, 1102, 1139
\input	137, 431, 1020, 1054, 1101, 1135
\iterate	972, 974, 976
<b>L</b>	
\lccode	1164
\LoadCommand	1020, 1030
\loop	970, 986, 997, 1005
\lowercase	1168, 1194
\ltx@backslashchar	367, 371, 374, 1206
\ltx@ccclv	404, 406
\ltx@firstoftwo	324, 354, 386
\ltx@gobble	1156, 1220
\ltx@ifUndefined	180, 366
\ltx@one	370, 385, 392, 393
\ltx@onelevel@sanitize	443, 1193, 1219, 1247, 1267
\ltx@percentchar	1223
\ltx@ReturnAfterElseFi	481, 492
\ltx@ReturnAfterFi	485, 497
\ltx@secondoftwo	326, 356, 380, 388
\ltx@zero	365, 379, 395, 493
\luaescapestring	509, 510, 516, 521, 529, 538, 544, 549, 554, 559, 564, 569, 570, 571, 576, 581, 586, 599, 625
\luatexversion	216, 467, 468, 503, 589
<b>M</b>	
\meaning	241, 243, 298, 301, 317, 353
\MessageBreak	271, 417, 454, 455
\msg	1055, 1070, 1078, 1102, 1104, 1106, 1111, 1113, 1118, 1120, 1136, 1258, 1259
<b>N</b>	
\newlinechar	222, 223, 1257
\next	976, 978, 980
\number	128, 569, 570, 1011, 1106, 1118
<b>P</b>	
\PackageInfo	26
\par	1091, 1092, 1093
\pdf@draftmode	4, 379, 391
\pdf@escapehex	3, 170, 514, 1276
\pdf@escapehexnative	170, 519
\pdf@escapename	547, 1278
\pdf@escapenamenative	552
\pdf@escapestring	542, 1279
\pdf@filedump	3, 199, 567
\pdf@filemdfivesum	4, 209, 584
\pdf@filemoddate	3, 562
\pdf@filesize	3, 557
\pdf@ifdraftmode	4, 380, 384
\pdf@ifprimitive	5, 255, 288
\pdf@isprimitive	5, 313, 316, 352, 363, 513, 1065, 1068
\pdf@lastsystemexit	607
\pdf@lastsystemstatus	602
\pdf@mdfivesum	4, 207, 208, 574
\pdf@mdfivesumnative	208, 579
\pdf@pipe	5, 613
\pdf@primitive	5, 251, 269, 271, 283
\pdf@setdraftmode	4, 402, 417
\pdf@shellescape	4, 184, 186, 191, 589, 1103, 1106, 1112, 1119
\pdf@strcmp	3, 353, 507
\pdf@system	4, 211, 597
\pdf@unescapehex	3, 172, 524, 1270, 1274, 1277
\pdf@unescapehexnative	5, 172, 533
\pdf@draftmode	383
\pdf@filedump	200
\pdf@mdfivesum	207, 209
\pdf@primitive	272
\pdf@shellescape	192
\pdf@texcmds@isprimitive	320, 322
\pdf@texcmds@setdraftmode	406, 410
\pdf@texcmds@AtEnd	95, 96, 126, 127, 424, 630
\pdf@texcmds@DecodeA	477, 479
\pdf@texcmds@DecodeB	482, 490
\pdf@texcmds@directlua	216, 224, 438, 445, 614, 623
\pdf@texcmds@draftmode	383, 385, 392, 399
\pdf@texcmds@equal	323, 329, 347
\pdf@texcmds@equalcont	338, 344, 345, 350
\pdf@texcmds@isprimitive	317, 319
\pdf@texcmds@nopdftex	153, 155, 160, 179, 197, 205
\pdf@texcmds@Patch	466, 472, 529, 538, 625
\pdf@texcmds@PatchDecode	476, 525, 534, 621
\pdf@texcmds@setdraftmode	381, 398, 412, 414
\pdf@texcmds@strip@prefix	235, 241
\pdf@texcmds@temp	157, 168, 169, 171, 173, 174, 175, 176, 236, 251, 252, 253, 254, 255, 256, 257, 258, 293, 311, 312, 365, 370, 378
\pdf@texcmds@toks	460, 526, 535, 622
\pdf@texrevision	267
\pdf@texversion	182, 264
\ProvidesPackage	19, 67
<b>R</b>	
\RangeCatcodeCheck	1003, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042
\RangeCatcodeInvalid	995, 1023, 1024, 1025, 1026
\repeat	970, 982, 993, 1001, 1016
\RequirePackage	145, 146, 147, 148, 436
\RestoreCatcodes	984, 987, 988, 1043
\romannumeral	525, 534, 621
<b>S</b>	
\space	272, 283, 288, 1009, 1010, 1018, 1145, 1166
\str	1064, 1070, 1072, 1076, 1078

<b>T</b>		\TMP@RequirePackage .....	
\Test .....	1022, 1045,	.	134, 140, 141, 142, 143, 428, 434
	1248, 1268, 1276, 1277, 1278, 1279	\toksdef .....	462
\test .....	1061, 1083, 1084, 1085,	<b>W</b>	
	1086, 1087, 1088, 1089, 1090,	\write ...	23, 52, 212, 1058, 1102, 1139
	1091, 1092, 1093, 1264, 1270, 1274	<b>X</b>	
\TestExpect .....	1266, 1267, 1268	\x ...	14, 15, 18, 22, 26, 28, 51, 56,
\TestFrom .....	1265, 1268		66, 75, 87, 240, 241, 245, 298,
\TestResult .....	1251, 1255, 1259		303, 405, 408, 442, 443, 451, 455
\the 77, 78, 79, 80, 81, 82, 83, 84, 97,		<b>Y</b>	
406, 526, 535, 622, 989, 1009, 1010		\y	243, 245, 299, 301, 303, 444, 451, 456
\TMP@EnsureCode .....	94,	<b>Z</b>	
101, 102, 103, 104, 105, 106,		\z .....	300, 301
107, 108, 109, 110, 111, 112,			
113, 114, 115, 116, 117, 118,			
119, 120, 121, 122, 123, 124, 125			