

# 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  $\text{\TeX}$  compiler knows `\pdfdraftmode` ( $\text{\pdfTeX}$ ,  $\text{\LuaTeX}$ ), 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  $\text{\pdfTeX}$  external commands must be enabled first by command line option or configuration option. In  $\text{\LuaTeX}$  option `--safer` disables the execution of external commands.

In  $\text{\LuaTeX}$  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  $\text{\LuaTeX}$ .

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  $\text{\pdfTeX}$  or `os.execute` in  $\text{\LuaTeX}$ .

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{ltxcms}[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{ltxcms}[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@nopdf tex{%
154     \@PackageInfoNoLine{pdf texcmds}{pdfTeX >= 1.30 not detected}%
155     \let\pdf texcmds@nopdf tex\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@nopdf tex
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@nopdf tex
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 \pdfTexcmds@directlua{%
439 require("oberdiek.pdfTexcmds")%
440 }
441 \begingroup
442 \def\x{2011/04/16 v0.15}%
443 \ltx@onelevel@sanitize\x
444 \edef\y{%
445 \pdfTexcmds@directlua{%
446 if oberdiek.pdfTexcmds.getversion then %
447 oberdiek.pdfTexcmds.getversion()%
448 end%
449 }%
450 }%
451 \ifx\x\y
452 \else
453 \@PackageError{pdfTexcmds}{%
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

\pdfTexcmds@toks

```

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

```

\pdfTexcmds@Patch

```

466 \def\pdfTexcmds@Patch{0}
467
468 \ifnum\luatexversion>40 %
469 \ifnum\luatexversion<66 %
470 \def\pdfTexcmds@Patch{1}%
471 \fi
472 \fi
473 \ifcase\pdfTexcmds@Patch
474 \catcode'\&=14 %
475 \else
476 \catcode'\&=9 %

```

\pdfTexcmds@PatchDecode

```

477 \def\pdfTexcmds@PatchDecode#1\@nil{%
478 \pdfTexcmds@DecodeA#1^^A^^A\@nil}%
479 }%

```

\pdfTexcmds@DecodeA

```

480 \def\pdf texcmds@DecodeA#1^^A^A#2\@nil#3{%
481   \ifx\relax#2\relax
482     \ltx@ReturnAfterElseFi{%
483       \pdf texcmds@DecodeB#3#1^^A^B\@nil{%}%
484     }%
485   \else
486     \ltx@ReturnAfterFi{%
487       \pdf texcmds@DecodeA#2\@nil{#3#1^^@}%
488     }%
489   \fi
490 }%

\pdf texcmds@DecodeB
491 \def\pdf texcmds@DecodeB#1^^A^B#2\@nil#3{%
492   \ifx\relax#2\relax%
493     \ltx@ReturnAfterElseFi{%
494       \ltx@zero
495       #3#1%
496     }%
497   \else
498     \ltx@ReturnAfterFi{%
499       \pdf texcmds@DecodeB#2\@nil{#3#1^^A}%
500     }%
501   \fi
502 }%

503 \fi

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

\pdf@strcmp
508 \long\def\pdf@strcmp#1#2{%
509   \directlua0{%
510     oberdiek.pdf texcmds.strptime("\luaescapestring{#1}",%
511       "\luaescapestring{#2}")}%
512   }%
513 }%

514 \pdf@isprimitive

\pdf@escapehex
515 \long\def\pdf@escapehex#1{%
516   \directlua0{%
517     oberdiek.pdf texcmds.escapehex("\luaescapestring{#1}", "byte")%
518   }%
519 }%

\pdf@escapehexnative
520 \long\def\pdf@escapehexnative#1{%
521   \directlua0{%
522     oberdiek.pdf texcmds.escapehex("\luaescapestring{#1}")%
523   }%
524 }%

\pdf@unescapehex
525 \def\pdf@unescapehex#1{%
526   & \romannumeral\expandafter\pdf texcmds@PatchDecode
527   \the\expandafter\pdf texcmds@toks
528   \directlua0{%

```

```

529 oberdiek.pdfTEXcmds.toks="pdfTEXcmds@toks"%
530 oberdiek.pdfTEXcmds.unescapehex("\luaescapestring{#1}", "byte", \pdfTEXcmds@Patch)%
531 }%
532 & \@nil
533 }%

```

\pdf@unescapehexnative

```

534 \def\pdf@unescapehexnative#1{%
535 & \romannumeral\expandafter\pdfTEXcmds@PatchDecode
536 \the\expandafter\pdfTEXcmds@toks
537 \directlua0{%
538 oberdiek.pdfTEXcmds.toks="pdfTEXcmds@toks"%
539 oberdiek.pdfTEXcmds.unescapehex("\luaescapestring{#1}", \pdfTEXcmds@Patch)%
540 }%
541 & \@nil
542 }%

```

\pdf@escapestring

```

543 \long\def\pdf@escapestring#1{%
544 \directlua0{%
545 oberdiek.pdfTEXcmds.escapestring("\luaescapestring{#1}", "byte")%
546 }%
547 }

```

\pdf@escapename

```

548 \long\def\pdf@escapename#1{%
549 \directlua0{%
550 oberdiek.pdfTEXcmds.escapename("\luaescapestring{#1}", "byte")%
551 }%
552 }

```

\pdf@escapenamenative

```

553 \long\def\pdf@escapenamenative#1{%
554 \directlua0{%
555 oberdiek.pdfTEXcmds.escapename("\luaescapestring{#1}")%
556 }%
557 }

```

\pdf@filesize

```

558 \def\pdf@filesize#1{%
559 \directlua0{%
560 oberdiek.pdfTEXcmds.filesize("\luaescapestring{#1}")%
561 }%
562 }

```

\pdf@filemoddate

```

563 \def\pdf@filemoddate#1{%
564 \directlua0{%
565 oberdiek.pdfTEXcmds.filemoddate("\luaescapestring{#1}")%
566 }%
567 }

```

\pdf@filedump

```

568 \def\pdf@filedump#1#2#3{%
569 \directlua0{%
570 oberdiek.pdfTEXcmds.filedump("\luaescapestring{\number#1}",%
571 "\luaescapestring{\number#2}",%
572 "\luaescapestring{#3}")%
573 }%
574 }%

```



```

\pdf@mdfivesum
575 \long\def\pdf@mdfivesum#1{%
576   \directlua0{%
577     oberdiek.pdf_texcmds.mdfivesum("\luaescapestring{#1}", "byte")%
578   }%
579 }%

\pdf@mdfivesumnative
580 \long\def\pdf@mdfivesumnative#1{%
581   \directlua0{%
582     oberdiek.pdf_texcmds.mdfivesum("\luaescapestring{#1}")%
583   }%
584 }%

\pdf@filemdfivesum
585 \def\pdf@filemdfivesum#1{%
586   \directlua0{%
587     oberdiek.pdf_texcmds.filemdfivesum("\luaescapestring{#1}")%
588   }%
589 }%

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

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

\pdf@lastsystemstatus
603 \def\pdf@lastsystemstatus{%
604   \directlua0{%
605     oberdiek.pdf_texcmds.lastsystemstatus()%
606   }%
607 }

\pdf@lastsystemexit
608 \def\pdf@lastsystemexit{%
609   \directlua0{%
610     oberdiek.pdf_texcmds.lastsystemexit()%
611   }%
612 }

613 \catcode'\0=12 %

\pdf@pipe Check availability of io.popen first.
614 \ifnum0%
615   \pdf_texcmds@directlua{%
616     if io.popen then %
617       tex.write("1")%
618     end%
619   }%

```

```

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

631 \pdf texcmds@AtEnd%
632 \package

```

## 2.11 Lua module

```

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

```

```

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

```

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

```

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

```

```

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

```

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

```

748 function escapestring(str, mode)
749   if mode == "byte" then
750     str = utf8_to_byte(str)
751   end
752   tex.write((string.gsub(str, ".",
753     function (ch)
754       local b = string.byte(ch)
755       if b < 33 or b > 126 then
756         return string.format("\\%.3o", b)
757       end
758       if b == 40 or b == 41 or b == 92 then
759         return "\\\" .. ch
760       end

```

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

```

761       return nil
762     end
763   )))
764 end

```

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

```

765 function escapename(str, mode)
766   if mode == "byte" then
767     str = utf8_to_byte(str)
768   end
769   tex.write((string.gsub(str, ".",
770     function (ch)
771       local b = string.byte(ch)
772       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.

```

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

```

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

```

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

```

```

793 end
794 end

```

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

```

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

```

```

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

```

```

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

```

## 3 Test

### 3.1 Catcode checks for loading

```

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

```

```

973 \iterate
974 }%
975 \def\iterate{%
976 \body
977 \let\next\iterate
978 \else
979 \let\next\relax
980 \fi
981 \next
982 }%
983 \let\repeat=\fi
984 }%
985 \def\RestoreCatcodes{}
986 \count@=0 %
987 \loop
988 \edef\RestoreCatcodes{%
989 \RestoreCatcodes
990 \catcode\the\count@=\the\catcode\count@\relax
991 }%
992 \ifnum\count@<255 %
993 \advance\count@ 1 %
994 \repeat
995
996 \def\RangeCatcodeInvalid#1#2{%
997 \count@=#1\relax
998 \loop
999 \catcode\count@=15 %
1000 \ifnum\count@<#2\relax
1001 \advance\count@ 1 %
1002 \repeat
1003 }
1004 \def\RangeCatcodeCheck#1#2#3{%
1005 \count@=#1\relax
1006 \loop
1007 \ifnum#3=\catcode\count@
1008 \else
1009 \errmessage{%
1010 Character \the\count@\space
1011 with wrong catcode \the\catcode\count@\space
1012 instead of \number#3%
1013 }%
1014 \fi
1015 \ifnum\count@<#2\relax
1016 \advance\count@ 1 %
1017 \repeat
1018 }
1019 \def\space{ }
1020 \expandafter\ifx\csname LoadCommand\endcsname\relax
1021 \def\LoadCommand{\input pdftexcmds.sty\relax}%
1022 \fi
1023 \def\Test{%
1024 \RangeCatcodeInvalid{0}{47}%
1025 \RangeCatcodeInvalid{58}{64}%
1026 \RangeCatcodeInvalid{91}{96}%
1027 \RangeCatcodeInvalid{123}{255}%
1028 \catcode'\@=12 %
1029 \catcode'\=0 %
1030 \catcode'\%=14 %
1031 \LoadCommand
1032 \RangeCatcodeCheck{0}{36}{15}%
1033 \RangeCatcodeCheck{37}{37}{14}%
1034 \RangeCatcodeCheck{38}{47}{15}%

```



```

1035 \RangeCatcodeCheck{48}{57}{12}%
1036 \RangeCatcodeCheck{58}{63}{15}%
1037 \RangeCatcodeCheck{64}{64}{12}%
1038 \RangeCatcodeCheck{65}{90}{11}%
1039 \RangeCatcodeCheck{91}{91}{15}%
1040 \RangeCatcodeCheck{92}{92}{0}%
1041 \RangeCatcodeCheck{93}{96}{15}%
1042 \RangeCatcodeCheck{97}{122}{11}%
1043 \RangeCatcodeCheck{123}{255}{15}%
1044 \RestoreCatcodes
1045 }
1046 \Test
1047 \csname @@end\endcsname
1048 \end
1049 </test1>

```

### 3.2 Test for \pdf@isprimitive

```

1050 <*test2>
1051 \catcode'\{=1 %
1052 \catcode'\}=2 %
1053 \catcode'\#=6 %
1054 \catcode'\@=11 %
1055 \input pdftexcmds.sty\relax
1056 \def\msg#1{%
1057   \begingroup
1058     \escapechar=92 %
1059     \immediate\write16{#1}%
1060   \endgroup
1061 }
1062 \long\def\test#1#2#3#4{%
1063   \begingroup
1064     #4%
1065     \def\str{%
1066       Test \string\pdf@isprimitive
1067       {\string #1}{\string #2}{...}: %
1068     }%
1069     \pdf@isprimitive{#1}{#2}{%
1070       \ifx#3Y%
1071         \msg{\str true ==> OK.}%
1072       \else
1073         \errmessage{\str false ==> FAILED}%
1074       \fi
1075     }{%
1076       \ifx#3Y%
1077         \errmessage{\str true ==> FAILED}%
1078       \else
1079         \msg{\str false ==> OK.}%
1080       \fi
1081     }%
1082   \endgroup
1083 }
1084 \test\relax\relax Y{}
1085 \test\foobar\relax Y{\let\foobar\relax}
1086 \test\foobar\relax N{}
1087 \test\hbox\hbox Y{}
1088 \test\foobar@hbox\hbox Y{\let\foobar@hbox\hbox}
1089 \test\if\if Y{}
1090 \test\if\ifx N{}
1091 \test\ifx\if N{}
1092 \test\par\par Y{}
1093 \test\hbox\par N{}
1094 \test\par\hbox N{}

```

```

1095 \csname @@end\endcsname\end
1096 </test2>

```

### 3.3 Test for \pdf@shellescape

```

1097 <*test-shell>
1098 \catcode'\{=1 %
1099 \catcode'\}=2 %
1100 \catcode'\#=6 %
1101 \catcode'\@=11 %
1102 \input pdftexcmds.sty\relax
1103 \def\msg#\{\immediate\write16}
1104 \ifx\pdf@shellescape\undefined
1105   \msg{SHELL=U}%
1106 \else
1107   \msg{SHELL=\number\pdf@shellescape}%
1108 \fi
1109 \ifx\expected\undefined
1110 \else
1111   \ifx\expected\relax
1112     \msg{EXPECTED=U}%
1113     \ifx\pdf@shellescape\undefined
1114       \msg{OK}%
1115     \else
1116       \errmessage{Failed}%
1117     \fi
1118   \else
1119     \msg{EXPECTED=\number\expected}%
1120     \ifnum\pdf@shellescape=\expected\relax
1121       \msg{OK}%
1122     \else
1123       \errmessage{Failed}%
1124     \fi
1125   \fi
1126 \fi
1127 \csname @@end\endcsname\end
1128 </test-shell>

```

### 3.4 Test for escape functions

```

1129 <*test-escape>
1130 \catcode'\{=1 %
1131 \catcode'\}=2 %
1132 \catcode'\#=6 %
1133 \catcode'\^=7 %
1134 \catcode'\@=11 %
1135 \errorcontextlines=1000 %
1136 \input pdftexcmds.sty\relax
1137 \def\msg#1{%
1138   \begingroup
1139     \escapechar=92 %
1140     \immediate\write16{#1}%
1141   \endgroup
1142 }
1143 \begingroup
1144   \catcode'\@=11 %
1145   \countdef\count@=255 %
1146   \def\space{ }%
1147   \long\def\@whilenum#1\do #2{%
1148     \ifnum #1\relax
1149       #2\relax
1150     \@iwhilenum{#1\relax#2\relax}%
1151   \fi
1152 }%

```

```

1153 \long\def\@iwhilenum#1{%
1154     \ifnum #1%
1155         \expandafter\@iwhilenum
1156     \else
1157         \expandafter\ltx@gobble
1158     \fi
1159     {#1}%
1160 }%
1161 \gdef\AllBytes{%
1162 \count@=0 %
1163 \catcode0=12 %
1164 \@whilenum\count@<256 \do{%
1165     \lccode0=\count@
1166     \ifnum\count@=32 %
1167         \xdef\AllBytes{\AllBytes\space}%
1168     \else
1169         \lowercase{%
1170             \xdef\AllBytes{\AllBytes^^@}%
1171         }%
1172     \fi
1173     \advance\count@ by 1 %
1174 }%
1175 \endgroup

1176 \def\AllBytesHex{%
1177 000102030405060708090A0B0C0D0E0F%
1178 101112131415161718191A1B1C1D1E1F%
1179 202122232425262728292A2B2C2D2E2F%
1180 303132333435363738393A3B3C3D3E3F%
1181 404142434445464748494A4B4C4D4E4F%
1182 505152535455565758595A5B5C5D5E5F%
1183 606162636465666768696A6B6C6D6E6F%
1184 707172737475767778797A7B7C7D7E7F%
1185 808182838485868788898A8B8C8D8E8F%
1186 909192939495969798999A9B9C9D9E9F%
1187 A0A1A2A3A4A5A6A7A8A9AABACADA EAF%
1188 B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF%
1189 C0C1C2C3C4C5C6C7C8C9CACBCCDCECF%
1190 D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
1191 E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
1192 F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF%
1193 }
1194 \ltx@onelevel@sanitize\AllBytesHex
1195 \expandafter\lowercase\expandafter{%
1196 \expandafter\def\expandafter\AllBytesHexLC
1197     \expandafter{\AllBytesHex}%
1198 }
1199 \begingroup
1200 \catcode'\#=12 %
1201 \xdef\AllBytesName{%
1202     #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
1203     #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
1204     #20!"#23$#25&'#28#29*+,-.#2F%
1205     0123456789:;#3C=#3E?%
1206     @ABCDEFGHIJKLMNO%
1207     PQRSTUVWXYZ#5B\ltx@backslashchar#5D^_%
1208     'abcdefghijklmnop%
1209     pqrstuvwxyz7B|#7D\string~#7F%
1210     #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
1211     #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
1212     #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
1213     #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
1214     #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%

```

```

1215     #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
1216     #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
1217     #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
1218 }%
1219 \endgroup
1220 \ltx@onelevel@sanitize\AllBytesName
1221 \edef\AllBytesFromName{\expandafter\ltx@gobble\AllBytes}
1222 \begingroup
1223   \def\|{|}%
1224   \edef\%{\ltx@percentchar}%
1225   \catcode'\|=0 %
1226   \catcode'\#=12 %
1227   \catcode'\~=12 %
1228   \catcode'\|=12 %
1229   \xdef\AllBytesString{%
1230     \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
1231     \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
1232     \040!"#$%&'(\)*+,-./%
1233     0123456789:;<=>?%
1234     @ABCDEFGHIJKLMNO%
1235     PQRSTUVWXYZ[\]^_%
1236     'abcdefghijklmnopqrstuvwxyz%
1237     pqrstuvwxyz{|}~\177%
1238     \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
1239     \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
1240     \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
1241     \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
1242     \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
1243     \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
1244     \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
1245     \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
1246 }%
1247 \endgroup
1248 \ltx@onelevel@sanitize\AllBytesString
1249 \def\Test#1#2#3{%
1250   \begingroup
1251     \expandafter\expandafter\expandafter\def
1252     \expandafter\expandafter\expandafter\TestResult
1253     \expandafter\expandafter\expandafter{%
1254       #1{#2}%
1255     }%
1256     \ifx\TestResult#3%
1257     \else
1258       \newlinechar=10 %
1259       \msg{Expect:^^J#3}%
1260       \msg{Result:^^J\TestResult}%
1261       \errmessage{\string#2 -\string#1-> \string#3}%
1262     \fi
1263   \endgroup
1264 }
1265 \def\test#1#2#3{%
1266   \edef\TestFrom{#2}%
1267   \edef\TestExpect{#3}%
1268   \ltx@onelevel@sanitize\TestExpect
1269   \Test#1\TestFrom\TestExpect
1270 }
1271 \test\pdf@unescapehex{74657374}{test}
1272 \begingroup
1273   \catcode0=12 %
1274   \catcode1=12 %
1275   \test\pdf@unescapehex{740074017400740174}{t^^@t^^@t^^@t^^@}%
1276 \endgroup

```

```

1277 \Test\pdf@escapehex\AllBytes\AllBytesHex
1278 \Test\pdf@unescapehex\AllBytesHex\AllBytes
1279 \Test\pdf@escapename\AllBytes\AllBytesName
1280 \Test\pdf@escapestring\AllBytes\AllBytesString
1281 \csname @@end\endcsname\end
1282 </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

`\#` . . . 954, 1053, 1100, 1132, 1200, 1226<sub>31</sub>

<code>\%</code> .....	1030, 1224	1007, 1010, 1011, 1015, 1016,
<code>\&amp;</code> .....	474, 476	1145, 1162, 1164, 1165, 1166, 1173
<code>\(</code> .....	1232	<code>\countdef</code> .....
<code>\)</code> .....	1232	957, 1145
<code>\@</code> ...	955, 1028, 1054, 1101, 1134, 1144	<code>\csname</code> .....
<code>\@PackageError</code> .....	453	14, 21, 50, 66,
<code>\@PackageInfoNoLine</code> .....	152,	76, 133, 136, 159, 162, 164, 178,
154, 270, 282, 287, 367, 371, 373		196, 204, 238, 243, 246, 247,
<code>\@PackageWarning</code> .....	416	260, 262, 265, 284, 289, 295,
<code>\@ehc</code> .....	457	298, 304, 306, 315, 350, 427,
<code>\@firstofone</code> .....	963, 966	430, 461, 464, 956, 959, 962,
<code>\@gobble</code> .....	960, 968	965, 1020, 1047, 1095, 1127, 1281
<code>\@iwhilenum</code> .....	1150, 1153, 1155	<b>D</b>
<code>\@nil</code> .....	477, 478, 480,	<code>\delimiter</code> .....
483, 487, 491, 499, 532, 541, 628		323, 329, 347
<code>\@undefined</code> ..	58, 269, 1104, 1109, 1113	<code>\directlua</code> .....
<code>\@whilenum</code> .....	1147, 1164	217,
<code>\</code> .....	756, 759, 1029, 1228, 1235	219, 509, 516, 521, 528, 537,
<code>\{</code> .....	952, 1051, 1098, 1130	544, 549, 554, 559, 564, 569,
<code>\}</code> .....	953, 1052, 1099, 1131	576, 581, 586, 593, 599, 604, 609
<code>\^</code> .....	1133	<code>\do</code> .....
<code>\ </code> .....	1223, 1225	1147, 1164
<code>\~</code> .....	1227	<b>E</b>
<b>Numbers</b>		<code>\empty</code> .....
<code>\0</code> .....	506, 613, 1230, 1231, 1232	17, 18
<code>\1</code> .....	1237	<code>\end</code> .....
<code>\2</code> .....	1238, 1239, 1240, 1241	1048, 1095, 1127, 1281
<code>\3</code> .....	1242, 1243, 1244, 1245	<code>\endcsname</code> .....
<b>A</b>		14, 21, 50, 66,
<code>\advance</code> .....	993, 1001, 1016, 1173	76, 133, 136, 159, 162, 164, 178,
<code>\aftergroup</code> .....	29	196, 204, 238, 243, 246, 247,
<code>\AllBytes</code> .....	1161, 1167,	260, 262, 265, 284, 289, 295,
1170, 1221, 1277, 1278, 1279, 1280		298, 304, 306, 315, 350, 427,
<code>\AllBytesFromName</code> .....	1221	430, 461, 464, 956, 959, 962,
<code>\AllBytesHex</code> .....		965, 1020, 1047, 1095, 1127, 1281
....	1176, 1194, 1197, 1277, 1278	<code>\endinput</code> .....
<code>\AllBytesHexLC</code> .....	1196	29, 129
<code>\AllBytesName</code> .....	1201, 1220, 1279	<code>\endlinechar</code> ....
<code>\AllBytesString</code> .....	1248, 1280	4, 35, 71, 77, 89, 223
<b>B</b>		<code>\errmessage</code> .....
<code>\body</code> .....	972, 976	1009, 1073, 1077, 1116, 1123, 1261
<b>C</b>		<code>\errorcontextlines</code> .....
<code>\catcode</code> ..	2, 3, 5, 6, 7, 8, 9, 10, 11,	1135
12, 13, 33, 34, 36, 37, 38, 39, 40,		<code>\escapechar</code> .....
41, 42, 43, 44, 45, 46, 47, 48, 49,		... 128, 131, 242, 297, 1058, 1139
69, 70, 72, 73, 74, 78, 79, 80, 81,		<code>\expected</code> ....
82, 83, 84, 87, 88, 90, 91, 92, 93,		1109, 1111, 1119, 1120
97, 99, 474, 476, 506, 613, 952,		<b>F</b>
953, 954, 955, 990, 999, 1007,		<code>\foobar</code> .....
1011, 1028, 1029, 1030, 1051,		1085, 1086
1052, 1053, 1054, 1098, 1099,		<code>\foobar@hbox</code> .....
1100, 1101, 1130, 1131, 1132,		1088
1133, 1134, 1144, 1163, 1200,		<b>G</b>
1225, 1226, 1227, 1228, 1273, 1274		<code>\gdef</code> .....
<code>\chardef</code> .....	184, 186	1161
<code>\count</code> .....	404, 406	<b>H</b>
<code>\count@</code> .....	957, 986, 990, 992,	<code>\hbox</code> .....
993, 997, 999, 1000, 1001, 1005,		1087, 1088, 1093, 1094
<b>I</b>		<b>I</b>
<code>\if</code> .....	1089, 1090, 1091	<code>\if</code> .....
<code>\ifcase</code> .....	378, 411, 473	1089, 1090, 1091
<code>\ifeof</code> .....	182, 183	<code>\ifcase</code> .....
<code>\ifluatex</code> .....	150, 215, 361, 422	378, 411, 473
<code>\ifnum</code> .....	182, 216, 264,	<code>\ifeof</code> .....
267, 323, 353, 385, 392, 468,		182, 183
469, 504, 590, 614, 992, 1000,		<code>\ifluatex</code> .....
1007, 1015, 1120, 1148, 1154, 1166		150, 215, 361, 422
<code>\ifpdf</code> .....	369	<code>\ifnum</code> .....
<code>\ifx</code> .....	15, 18, 21, 50,	182, 216, 264,
58, 61, 133, 136, 159, 178, 196,		267, 323, 353, 385, 392, 468,
204, 238, 245, 260, 262, 265,		469, 504, 590, 614, 992, 1000,
284, 289, 295, 303, 315, 330,		1007, 1015, 1120, 1148, 1154, 1166
331, 334, 336, 427, 430, 451,		<code>\ifpdf</code> .....



461, 481, 492, 956, 959, 962, 965, 1020, 1070, 1076, 1090, 1091, 1104, 1109, 1111, 1113, 1256	
\immediate	23, 52, 212, 1059, 1103, 1140
\input	137, 431, 1021, 1055, 1102, 1136
\iterate	973, 975, 977
<b>L</b>	
\lccode	1165
\LoadCommand	1021, 1031
\loop	971, 987, 998, 1006
\lowercase	1169, 1195
\ltx@backslashchar	367, 371, 374, 1207
\ltx@ccclv	404, 406
\ltx@firstoftwo	324, 354, 386
\ltx@gobble	1157, 1221
\ltx@ifUndefined	180, 366
\ltx@one	370, 385, 392, 393
\ltx@onelevel@sanitize	443, 1194, 1220, 1248, 1268
\ltx@percentchar	1224
\ltx@ReturnAfterElseFi	482, 493
\ltx@ReturnAfterFi	486, 498
\ltx@secondoftwo	326, 356, 380, 388
\ltx@zero	365, 379, 395, 494
\luaescapestring	510, 511, 517, 522, 530, 539, 545, 550, 555, 560, 565, 570, 571, 572, 577, 582, 587, 600, 626
\luatexversion	216, 468, 469, 504, 590
<b>M</b>	
\meaning	241, 243, 298, 301, 317, 353
\MessageBreak	271, 417, 454, 455
\msg	1056, 1071, 1079, 1103, 1105, 1107, 1112, 1114, 1119, 1121, 1137, 1259, 1260
<b>N</b>	
\newlinechar	222, 223, 1258
\next	977, 979, 981
\number	128, 570, 571, 1012, 1107, 1119
<b>P</b>	
\PackageInfo	26
\par	1092, 1093, 1094
\pdf@draftmode	4, 379, 391
\pdf@escapehex	3, 170, 515, 1277
\pdf@escapehexnative	170, 520
\pdf@escapename	548, 1279
\pdf@escapenamenative	553
\pdf@escapestring	543, 1280
\pdf@filedump	3, 199, 568
\pdf@filemdfivesum	4, 209, 585
\pdf@filemoddate	3, 563
\pdf@filesize	3, 558
\pdf@ifdraftmode	4, 380, 384
\pdf@ifprimitive	5, 255, 288
\pdf@isprimitive	5, 313, 316, 352, 363, 514, 1066, 1069
\pdf@lastsystemexit	608
\pdf@lastsystemstatus	603
\pdf@mdfivesum	4, 207, 208, 575
\pdf@mdfivesumnative	208, 580
\pdf@pipe	5, 614
\pdf@primitive	5, 251, 269, 271, 283
\pdf@setdraftmode	4, 402, 417
\pdf@shellescape	4, 184, 186, 191, 590, 1104, 1107, 1113, 1120
\pdf@strcmp	3, 353, 508
\pdf@system	4, 211, 598
\pdf@unescapehex	3, 172, 525, 1271, 1275, 1278
\pdf@unescapehexnative	5, 172, 534
\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, 631
\pdf@texcmds@DecodeA	478, 480
\pdf@texcmds@DecodeB	483, 491
\pdf@texcmds@directlua	216, 224, 438, 445, 615, 624
\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, 473, 530, 539, 626
\pdf@texcmds@PatchDecode	477, 526, 535, 622
\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, 527, 536, 623
\pdf@texrevision	267
\pdf@texversion	182, 264
\ProvidesPackage	19, 67
<b>R</b>	
\RangeCatcodeCheck	1004, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043
\RangeCatcodeInvalid	996, 1024, 1025, 1026, 1027
\repeat	971, 983, 994, 1002, 1017
\RequirePackage	145, 146, 147, 148, 436
\RestoreCatcodes	985, 988, 989, 1044
\romannumeral	526, 535, 622
<b>S</b>	
\space	272, 283, 288, 1010, 1011, 1019, 1146, 1167
\str	1065, 1071, 1073, 1077, 1079

	<b>T</b>		<b>\TMP@RequirePackage</b> .....
<b>\Test</b> .....	1023, 1046,		. 134, 140, 141, 142, 143, 428, 434
	1249, 1269, 1277, 1278, 1279, 1280	<b>\toksdef</b> .....	462
<b>\test</b> .....	1062, 1084, 1085, 1086,		
	1087, 1088, 1089, 1090, 1091,	<b>W</b>	
	1092, 1093, 1094, 1265, 1271, 1275	<b>\write</b> ...	23, 52, 212, 1059, 1103, 1140
<b>\TestExpect</b> .....	1267, 1268, 1269		
<b>\TestFrom</b> .....	1266, 1269	<b>X</b>	
<b>\TestResult</b> .....	1252, 1256, 1260	<b>\x</b> ...	14, 15, 18, 22, 26, 28, 51, 56,
<b>\the</b> 77, 78, 79, 80, 81, 82, 83, 84, 97,			66, 75, 87, 240, 241, 245, 298,
406, 527, 536, 623, 990, 1010, 1011			303, 405, 408, 442, 443, 451, 455
<b>\TMP@EnsureCode</b> .....	94,	<b>Y</b>	
101, 102, 103, 104, 105, 106,		<b>\y</b>	243, 245, 299, 301, 303, 444, 451, 456
107, 108, 109, 110, 111, 112,			
113, 114, 115, 116, 117, 118,		<b>Z</b>	
119, 120, 121, 122, 123, 124, 125		<b>\z</b> .....	300, 301