

The pdf texcmds package

Heiko Oberdiek
<heiko.oberdiek at gmail.com>

2011/04/10 v0.14

Abstract

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

Contents

| | | |
|----------|---|-----------|
| 1 | Documentation | 2 |
| 1.1 | General principles | 2 |
| 1.2 | Macros | 3 |
| 1.2.1 | Additional macro: <code>\pdf@isprimitive</code> | 5 |
| 1.2.2 | Experimental | 5 |
| 2 | Implementation | 5 |
| 2.1 | Reload check and package identification | 5 |
| 2.2 | Catcodes | 6 |
| 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 | 10 |
| 2.6 | <code>X_gTeX</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 |
| 3 | Test | 23 |
| 3.1 | Catcode checks for loading | 23 |
| 3.2 | Test for <code>\pdf@isprimitive</code> | 24 |
| 3.3 | Test for <code>\pdf@shellescape</code> | 25 |
| 3.4 | Test for escape functions | 26 |
| 4 | Installation | 28 |
| 4.1 | Download | 28 |
| 4.2 | Bundle installation | 29 |
| 4.3 | Package installation | 29 |
| 4.4 | Refresh file name databases | 29 |
| 4.5 | Some details for the interested | 29 |

| | |
|--------------------|-----------|
| 5 History | 30 |
| [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] | 30 |
| [2009/09/23 v0.6] | 30 |
| [2009/12/12 v0.7] | 30 |
| [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 |
| 6 Index | 31 |

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.

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 its 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^AT_EX:

```

\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^AT_EX.

```

2 \begingroup\catcode61\catcode48\catcode32=10\relax%
3 \catcode13=5 % ^^M
4 \endlinechar=13 %
5 \catcode35=6 % #
6 \catcode39=12 % '
7 \catcode44=12 % ,
8 \catcode45=12 % -
9 \catcode46=12 % .
10 \catcode58=12 % :

```

```

11 \catcode64=11 % @
12 \catcode123=1 % {
13 \catcode125=2 % }
14 \expandafter\let\expandafter\x\csname ver@pdf texcmds.sty\endcsname
15 \ifx\x\relax % plain-TeX, first loading
16 \else
17   \def\empty{}%
18   \ifx\x\empty % LaTeX, first loading,
19     % variable is initialized, but \ProvidesPackage not yet seen
20   \else
21     \expandafter\ifx\csname PackageInfo\endcsname\relax
22       \def\x#1#2{%
23         \immediate\write-1{Package #1 Info: #2.}%
24       }%
25     \else
26       \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
27     \fi
28     \x{pdf texcmds}{The package is already loaded}%
29   \aftergroup\endinput
30 \fi
31 \fi
32 \endgroup%

```

Package identification:

```

33 \begingroup\catcode61\catcode48\catcode32=10\relax%
34 \catcode13=5 % ^^M
35 \endlinechar=13 %
36 \catcode35=6 % #
37 \catcode39=12 % '
38 \catcode40=12 % (
39 \catcode41=12 % )
40 \catcode44=12 % ,
41 \catcode45=12 % -
42 \catcode46=12 % .
43 \catcode47=12 % /
44 \catcode58=12 % :
45 \catcode64=11 % @
46 \catcode91=12 % [
47 \catcode93=12 % ]
48 \catcode123=1 % {
49 \catcode125=2 % }
50 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
51   \def\x#1#2#3[#4]{\endgroup
52     \immediate\write-1{Package: #3 #4}%
53     \xdef#1{#4}%
54   }%
55 \else
56   \def\x#1#2[#3]{\endgroup
57     #2[#{#3}]%
58     \ifx#1\@undefined
59       \xdef#1{#3}%
60     \fi
61     \ifx#1\relax
62       \xdef#1{#3}%
63     \fi
64   }%
65 \fi
66 \expandafter\x\csname ver@pdf texcmds.sty\endcsname
67 \ProvidesPackage{pdf texcmds}%
68 [2011/04/10 v0.14 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\pdftexcmds@AtEnd{%
127   \pdftexcmds@AtEnd
128   \escapechar=\number\escapechar\relax
129   \noexpand\endinput
130 }

```

```
131 \escapechar=92 %
```

2.3 Load packages

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

2.4 Without LuaTeX

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



```

189 }%
190 \else
191   \def\pdf@shellescape{%
192     \pdfshellescape
193   }%
194 \fi
195 \begingroup\expandafter\expandafter\expandafter\endgroup
196 \expandafter\ifx\csname pdffiledump\endcsname\relax
197   \pdfdocmds@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   \pdfdocmds@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_YTeX 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

\pdfdocmds@directlua

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

221 \begingroup
222   \newlinechar=10 %
223   \endlinechar=\newlinechar
224   \pdfdocmds@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}{luatexprimitive}{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}{luatexifprimitive}{ifpdfprimitive}% LuaTeX
258 \pdftexcmds@temp{ifprimitive}{luatexifpdfprimitive}{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_YT_EX

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 \ifx##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\pdf@draftmode=\ltx@one
393     \expandafter\ltx@one
394   \else
395     \expandafter\ltx@zero
396   \fi
397 }%

```

\pdf@draftmode

```

398 \def\pdf@draftmode#1{%
399   \pdf@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\pdf@setdraftmode{\the\count\ltx@cclv}%
407   }%
408   \x
409 }

```

\pdf@setdraftmode

```

410 \def\pdf@setdraftmode#1{%
411   \ifcase#1 %
412     \pdf@setdraftmode{#1}%
413   \or
414     \pdf@setdraftmode{#1}%
415   \else
416     \@PackageWarning{pdf}{%
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\pdf@setdraftmode@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 \pdf texcmds@directlua{%
439   require("oberdiek.pdf texcmds")%
440 }
441 \begingroup
442   \def\x{2011/04/10 v0.14}%
443   \ltx@onelevel@sanitize\x
444   \edef\y{%
445     \pdf texcmds@directlua{%
446       if oberdiek.pdf texcmds.getversion then %
447         oberdiek.pdf texcmds.getversion()%
448       end%
449     }%
450   }%
451   \ifx\x\y
452   \else
453     \@PackageError{pdf texcmds}{%
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

`\pdf texcmds@toks`

```

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

```

`\pdf texcmds@Patch`

```

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

```

`\pdf texcmds@PatchDecode`

```

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

```

`\pdf texcmds@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 }%

\pdftexcmds@DecodeB

491 \def\pdftexcmds@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             \pdftexcmds@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.pdf texcmds.toks="pdf texcmds@toks"%
530         oberdiek.pdf texcmds.unescapehex("\luaescapestring{#1}", "byte", \pdf texcmds@Patch)%
531     }%
532     & \@nil
533 }%

\pdf@unescapehexnative

534 \def\pdf@unescapehexnative#1{%
535     & \romannumeral\expandafter\pdf texcmds@PatchDecode

```

```

536 \the\expandafter\pdf texcmds@toks
537 \directlua0{%
538     oberdiek.pdf texcmds.toks="pdf texcmds@toks"%
539     oberdiek.pdf texcmds.unescapehex("\luaescapestring{#1}", \pdf texcmds@Patch)%
540 }%
541 & \@nil
542 }%

\pdf@escapestring

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

\pdf@escapename

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

\pdf@escapenamenative

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

\pdf@filesize

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

\pdf@filemoddate

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

\pdf@filedump

568 \def\pdf@filedump#1#2#3{%
569     \directlua0{%
570         oberdiek.pdf texcmds.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 \def\pdf@shellescape{%
591   \directlua0{%
592     oberdiek.pdf_texcmds.shellescape()%
593   }%
594 }
```

\pdf@system

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

\pdf@lastsystemstatus

```
600 \def\pdf@lastsystemstatus{%
601   \directlua0{%
602     oberdiek.pdf_texcmds.lastsystemstatus()%
603   }%
604 }
```

\pdf@lastsystemexit

```
605 \def\pdf@lastsystemexit{%
606   \directlua0{%
607     oberdiek.pdf_texcmds.lastsystemexit()%
608   }%
609 }
```

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

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

```
611 \ifnum0%
612   \pdf_texcmds@directlua{%
613     if io.popen then %
614       tex.write("1")%
615     end%
616   }%
617   =1 %
618   \def\pdf@pipe#1{%
619 &   \romannumeral\expandafter\pdf_texcmds@PatchDecode
620   \the\expandafter\pdf_texcmds@toks
621   \pdf_texcmds@directlua{%
622     oberdiek.pdf_texcmds.toks="pdf_texcmds@toks"%
623     oberdiek.pdf_texcmds.pipe("\luaescapestring{#1}", \pdf_texcmds@Patch)%
624   }%
625 &   \@nil
626   }%
627 \fi
```

```

628 \pdfTexcmds@AtEnd%
629 \</package>

```

2.11 Lua module

```

630 \<lua>

631 module("oberdiek.pdfTexcmds", package.seeall)
632 local systemexitstatus
633 function getversion()
634   tex.write("2011/04/10 v0.14")
635 end
636 function strcmp(A, B)
637   if A == B then
638     tex.write("0")
639   elseif A < B then
640     tex.write("-1")
641   else
642     tex.write("1")
643   end
644 end
645 local function utf8_to_byte(str)
646   local i = 0
647   local n = string.len(str)
648   local t = {}
649   while i < n do
650     i = i + 1
651     local a = string.byte(str, i)
652     if a < 128 then
653       table.insert(t, string.char(a))
654     else
655       if a >= 192 and i < n then
656         i = i + 1
657         local b = string.byte(str, i)
658         if b < 128 or b >= 192 then
659           i = i - 1
660         elseif a == 194 then
661           table.insert(t, string.char(b))
662         elseif a == 195 then
663           table.insert(t, string.char(b + 64))
664         end
665       end
666     end
667   end
668   return table.concat(t)
669 end
670 function escapehex(str, mode)
671   if mode == "byte" then
672     str = utf8_to_byte(str)
673   end
674   tex.write((string.gsub(str, ".",
675     function (ch)
676       return string.format("%02X", string.byte(ch))
677     end
678   )))
679 end

```

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

```

680 function unescapehex(str, mode, patch)
681   local a = 0
682   local first = true
683   local result = {}
684   for i = 1, string.len(str), 1 do

```

```

685     local ch = string.byte(str, i)
686     if ch >= 48 and ch <= 57 then
687         ch = ch - 48
688     elseif ch >= 65 and ch <= 70 then
689         ch = ch - 55
690     elseif ch >= 97 and ch <= 102 then
691         ch = ch - 87
692     else
693         ch = nil
694     end
695     if ch then
696         if first then
697             a = ch * 16
698             first = false
699         else
700             table.insert(result, a + ch)
701             first = true
702         end
703     end
704 end
705 if not first then
706     table.insert(result, a)
707 end
708 if patch == 1 then
709     local temp = {}
710     for i, a in ipairs(result) do
711         if a == 0 then
712             table.insert(temp, 1)
713             table.insert(temp, 1)
714         else
715             if a == 1 then
716                 table.insert(temp, 1)
717                 table.insert(temp, 2)
718             else
719                 table.insert(temp, a)
720             end
721         end
722     end
723     result = temp
724 end
725 if mode == "byte" then
726     local utf8 = {}
727     for i, a in ipairs(result) do
728         if a < 128 then
729             table.insert(utf8, a)
730         else
731             if a < 192 then
732                 table.insert(utf8, 194)
733                 a = a - 128
734             else
735                 table.insert(utf8, 195)
736                 a = a - 192
737             end
738             table.insert(utf8, a + 128)
739         end
740     end
741     result = utf8
742 end
743 tex.settoks(toks, string.char(unpack(result)))
744 end

```

See procedure `escapestring` in file `utils.c` of pdf_TE_X.
745 function `escapestring(str, mode)`

```

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

```

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

```

758     return nil
759 end
760 )))
761 end

```

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

```

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

```

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

```

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

```

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

```

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

```

792 function filemoddate(filename)
793     local foundfile = kpse.find_file(filename, "tex", true)
794     if foundfile then
795         local date = lfs.attributes(foundfile, "modification")
796         if date then
797             local d = os.date("*t", date)
798             if d.sec >= 60 then
799                 d.sec = 59
800             end

```

```

801     local u = os.date("!*t", date)
802     local off = 60 * (d.hour - u.hour) + d.min - u.min
803     if d.year ~= u.year then
804         if d.year > u.year then
805             off = off + 1440
806         else
807             off = off - 1440
808         end
809     elseif d.yday ~= u.yday then
810         if d.yday > u.yday then
811             off = off + 1440
812         else
813             off = off - 1440
814         end
815     end
816     local timezone
817     if off == 0 then
818         timezone = "Z"
819     else
820         local hours = math.floor(off / 60)
821         local mins = math.abs(off - hours * 60)
822         timezone = string.format("%+03d'%02d'", hours, mins)
823     end
824     tex.write(string.format("D:%04d%02d%02d%02d%02d%s",
825         d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
826 end
827 end
828 end
829 function filedump(offset, length, filename)
830     length = tonumber(length)
831     if length and length > 0 then
832         local foundfile = kpse.find_file(filename, "tex", true)
833         if foundfile then
834             offset = tonumber(offset)
835             if not offset then
836                 offset = 0
837             end
838             local filehandle = io.open(foundfile, "r")
839             if filehandle then
840                 if offset > 0 then
841                     filehandle:seek("set", offset)
842                 end
843                 local dump = filehandle:read(length)
844                 escapehex(dump)
845             end
846         end
847     end
848 end
849 function md5sum(str, mode)
850     if mode == "byte" then
851         str = utf8_to_byte(str)
852     end
853     escapehex(md5.sum(str))
854 end
855 function filemd5sum(filename)
856     local foundfile = kpse.find_file(filename, "tex", true)
857     if foundfile then
858         local filehandle = io.open(foundfile, "r")
859         if filehandle then
860             local contents = filehandle:read("*a")
861             escapehex(md5.sum(contents))
862         end
863     end

```

```

863 end
864 end
865 function shellescape()
866   if os.execute then
867     local result = os.execute()
868     if result == 0 then
869       tex.write("0")
870     else
871       if result == nil then
872         tex.write("0")
873       else
874         tex.write("1")
875       end
876     end
877   else
878     tex.write("0")
879   end
880 end
881 function system(cmdline)
882   systemexitstatus = nil
883   texio.write_nl("log", "system(" .. cmdline .. ") ")
884   if os.execute then
885     texio.write("log", "executed.")
886     systemexitstatus = os.execute(cmdline)
887   else
888     texio.write("log", "disabled.")
889   end
890 end
891 function lastsystemstatus()
892   local result = tonumber(systemexitstatus)
893   if result then
894     local x = math.floor(result / 256)
895     tex.write(result - 256 * math.floor(result / 256))
896   end
897 end
898 function lastsystemexit()
899   local result = tonumber(systemexitstatus)
900   if result then
901     tex.write(math.floor(result / 256))
902   end
903 end
904 function pipe(cmdline, patch)
905   local result
906   systemexitstatus = nil
907   texio.write_nl("log", "pipe(" .. cmdline .. ") ")
908   if io.popen then
909     texio.write("log", "executed.")
910     local handle = io.popen(cmdline, "r")
911     if handle then
912       result = handle:read("*a")
913       handle:close()
914     end
915   else
916     texio.write("log", "disabled.")
917   end
918   if result then
919     if patch == 1 then
920       local temp = {}
921       for i, a in ipairs(result) do
922         if a == 0 then
923           table.insert(temp, 1)
924           table.insert(temp, 1)

```

```

925         else
926             if a == 1 then
927                 table.insert(temp, 1)
928                 table.insert(temp, 2)
929             else
930                 table.insert(temp, a)
931             end
932         end
933     end
934     result = temp
935 end
936 tex.settoks(toks, result)
937 else
938     tex.settoks(toks, "")
939 end
940 end
941  $\langle$ /lua $\rangle$ 

```

3 Test

3.1 Catcode checks for loading

```

942  $\langle$ *test1 $\rangle$ 
943 \catcode'\{=1 %
944 \catcode'\}=2 %
945 \catcode'\#=6 %
946 \catcode'\@=11 %
947 \expandafter\ifx\csname count@\endcsname\relax
948   \countdef\count@=255 %
949 \fi
950 \expandafter\ifx\csname @gobble\endcsname\relax
951   \long\def\@gobble#1{}%
952 \fi
953 \expandafter\ifx\csname @firstofone\endcsname\relax
954   \long\def\@firstofone#1{#1}%
955 \fi
956 \expandafter\ifx\csname loop\endcsname\relax
957   \expandafter\@firstofone
958 \else
959   \expandafter\@gobble
960 \fi
961 {%
962   \def\loop#1\repeat{%
963     \def\body{#1}%
964     \iterate
965   }%
966   \def\iterate{%
967     \body
968     \let\next\iterate
969   \else
970     \let\next\relax
971   \fi
972   \next
973 }%
974 \let\repeat=\fi
975 }%
976 \def\RestoreCatcodes{}
977 \count@=0 %
978 \loop
979   \edef\RestoreCatcodes{%
980     \RestoreCatcodes
981     \catcode\the\count@=\the\catcode\count@\relax

```

```

982 }%
983 \ifnum\count@<255 %
984 \advance\count@ 1 %
985 \repeat
986
987 \def\RangeCatcodeInvalid#1#2{%
988 \count@=#1\relax
989 \loop
990 \catcode\count@=15 %
991 \ifnum\count@<#2\relax
992 \advance\count@ 1 %
993 \repeat
994 }
995 \def\RangeCatcodeCheck#1#2#3{%
996 \count@=#1\relax
997 \loop
998 \ifnum#3=\catcode\count@
999 \else
1000 \errmessage{%
1001 Character \the\count@\space
1002 with wrong catcode \the\catcode\count@\space
1003 instead of \number#3%
1004 }%
1005 \fi
1006 \ifnum\count@<#2\relax
1007 \advance\count@ 1 %
1008 \repeat
1009 }
1010 \def\space{ }
1011 \expandafter\ifx\csname LoadCommand\endcsname\relax
1012 \def\LoadCommand{\input pdftexcmds.sty\relax}%
1013 \fi
1014 \def\Test{%
1015 \RangeCatcodeInvalid{0}{47}%
1016 \RangeCatcodeInvalid{58}{64}%
1017 \RangeCatcodeInvalid{91}{96}%
1018 \RangeCatcodeInvalid{123}{255}%
1019 \catcode'\@=12 %
1020 \catcode'\=0 %
1021 \catcode'\%=14 %
1022 \LoadCommand
1023 \RangeCatcodeCheck{0}{36}{15}%
1024 \RangeCatcodeCheck{37}{37}{14}%
1025 \RangeCatcodeCheck{38}{47}{15}%
1026 \RangeCatcodeCheck{48}{57}{12}%
1027 \RangeCatcodeCheck{58}{63}{15}%
1028 \RangeCatcodeCheck{64}{64}{12}%
1029 \RangeCatcodeCheck{65}{90}{11}%
1030 \RangeCatcodeCheck{91}{91}{15}%
1031 \RangeCatcodeCheck{92}{92}{0}%
1032 \RangeCatcodeCheck{93}{96}{15}%
1033 \RangeCatcodeCheck{97}{122}{11}%
1034 \RangeCatcodeCheck{123}{255}{15}%
1035 \RestoreCatcodes
1036 }
1037 \Test
1038 \csname @@end\endcsname
1039 \end
1040 </test1>

```

3.2 Test for \pdf@isprimitive

```

1041 <*test2>

```



```

1042 \catcode'\{=1 %
1043 \catcode'\}=2 %
1044 \catcode'\#=6 %
1045 \catcode'\@=11 %
1046 \input pdftexcmds.sty\relax
1047 \def\msg#1{%
1048   \begingroup
1049     \escapechar=92 %
1050     \immediate\write16{#1}%
1051   \endgroup
1052 }
1053 \long\def\test#1#2#3#4{%
1054   \begingroup
1055     #4%
1056     \def\str{%
1057       Test \string\pdf@isprimitive
1058       {\string #1}{\string #2}{...}: %
1059     }%
1060     \pdf@isprimitive{#1}{#2}{%
1061       \ifx#3Y%
1062         \msg{\str true ==> OK.}%
1063       \else
1064         \errmessage{\str false ==> FAILED}%
1065       \fi
1066     }{%
1067       \ifx#3Y%
1068         \errmessage{\str true ==> FAILED}%
1069       \else
1070         \msg{\str false ==> OK.}%
1071       \fi
1072     }%
1073   \endgroup
1074 }
1075 \test\relax\relax Y{}
1076 \test\foobar\relax Y{\let\foobar\relax}
1077 \test\foobar\relax N{}
1078 \test\hbox\hbox Y{}
1079 \test\foobar@hbox\hbox Y{\let\foobar@hbox\hbox}
1080 \test\if\if Y{}
1081 \test\if\ifx N{}
1082 \test\ifx\if N{}
1083 \test\par\par Y{}
1084 \test\hbox\par N{}
1085 \test\par\hbox N{}
1086 \csname @@end\endcsname\end
1087 </test2>

```

3.3 Test for \pdf@shellescape

```

1088 <*test-shell>
1089 \catcode'\{=1 %
1090 \catcode'\}=2 %
1091 \catcode'\#=6 %
1092 \catcode'\@=11 %
1093 \input pdftexcmds.sty\relax
1094 \def\msg#{\immediate\write16}
1095 \ifx\pdf@shellescape\@undefined
1096   \msg{SHELL=U}%
1097 \else
1098   \msg{SHELL=\number\pdf@shellescape}%
1099 \fi
1100 \ifx\expected\@undefined
1101 \else

```

```

1102 \ifx\expected\relax
1103   \msg{EXPECTED=U}%
1104   \ifx\pdf@shellescape\undefined
1105     \msg{OK}%
1106   \else
1107     \errmessage{Failed}%
1108   \fi
1109 \else
1110   \msg{EXPECTED=\number\expected}%
1111   \ifnum\pdf@shellescape=\expected\relax
1112     \msg{OK}%
1113   \else
1114     \errmessage{Failed}%
1115   \fi
1116 \fi
1117 \fi
1118 \csname @@end\endcsname\end
1119 </test-shell>

```

3.4 Test for escape functions

```

1120 <*test-escape>
1121 \catcode'\{=1 %
1122 \catcode'\}=2 %
1123 \catcode'\#=6 %
1124 \catcode'\^=7 %
1125 \catcode'\@=11 %
1126 \errorcontextlines=1000 %
1127 \input pdftexcmds.sty\relax
1128 \def\msg#1{%
1129   \begingroup
1130     \escapechar=92 %
1131     \immediate\write16{#1}%
1132   \endgroup
1133 }

1134 \begingroup
1135   \catcode'\@=11 %
1136   \countdef\count@=255 %
1137   \def\space{ }%
1138   \long\def\@whilenum#1\do #2{%
1139     \ifnum #1\relax
1140       #2\relax
1141       \@iwhilenum{#1\relax#2\relax}%
1142     \fi
1143   }%
1144   \long\def\@iwhilenum#1{%
1145     \ifnum #1%
1146       \expandafter\@iwhilenum
1147     \else
1148       \expandafter\ltx@gobble
1149     \fi
1150     {#1}%
1151   }%
1152   \gdef\AllBytes{}%
1153   \count@=0 %
1154   \catcode0=12 %
1155   \@whilenum\count@<256 \do{%
1156     \lccode0=\count@
1157     \ifnum\count@=32 %
1158       \xdef\AllBytes{\AllBytes\space}%
1159     \else
1160       \lowercase{%
1161         \xdef\AllBytes{\AllBytes^^@}%

```

```

1162     }%
1163     \fi
1164     \advance\count@ by 1 %
1165 }%
1166 \endgroup

1167 \def\AllBytesHex{%
1168 000102030405060708090A0B0C0D0E0F%
1169 101112131415161718191A1B1C1D1E1F%
1170 202122232425262728292A2B2C2D2E2F%
1171 303132333435363738393A3B3C3D3E3F%
1172 404142434445464748494A4B4C4D4E4F%
1173 505152535455565758595A5B5C5D5E5F%
1174 606162636465666768696A6B6C6D6E6F%
1175 707172737475767778797A7B7C7D7E7F%
1176 808182838485868788898A8B8C8D8E8F%
1177 909192939495969798999A9B9C9D9E9F%
1178 A0A1A2A3A4A5A6A7A8A9AAABACADAEAF%
1179 B0B1B2B3B4B5B6B7B8B9ABBBBCBDBEBF%
1180 C0C1C2C3C4C5C6C7C8C9CACBCCDCECF%
1181 D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
1182 E0E1E2E3E4E5E6E7E8E9AEBECEDEEEF%
1183 F0F1F2F3F4F5F6F7F8F9AFBFCFDFFEFF%
1184 }
1185 \ltx@onelevel@sanitize\AllBytesHex
1186 \expandafter\lowercase\expandafter{%
1187   \expandafter\def\expandafter\AllBytesHexLC
1188     \expandafter{\AllBytesHex}%
1189 }
1190 \begingroup
1191   \catcode'\#=12 %
1192   \xdef\AllBytesName{%
1193     #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
1194     #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
1195     #20!"#$%&'#28#29*+,-.#2F%
1196     0123456789:;#3C=#3E?%
1197     @ABCDEFGHIJKLMNO%
1198     PQRSTUVWXYZ#5B\ltx@backslashchar#5D^_%
1199     'abcdefghijklmno%
1200     pqrstuvwxyz#7B|#7D|string~#7F%
1201     #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
1202     #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
1203     #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
1204     #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
1205     #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
1206     #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
1207     #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
1208     #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
1209   }%
1210 \endgroup
1211 \ltx@onelevel@sanitize\AllBytesName
1212 \edef\AllBytesFromName{\expandafter\ltx@gobble\AllBytes}
1213 \begingroup
1214   \def\{|}%
1215   \edef%\{\ltx@percentchar}%
1216   \catcode'\|=0 %
1217   \catcode'\#=12 %
1218   \catcode'\~=12 %
1219   \catcode'\|=12 %
1220   \xdef\AllBytesString{%
1221     \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
1222     \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
1223     \040!"#$%&'(\)*+,-./%

```

```

1224 0123456789:;<=>?%
1225 @ABCDEFGHIJKLMNO%
1226 PQRSTUVWXYZ[\]^_%
1227 'abcdefghijklmno%
1228 pqrstuvwxyz{|}~\177%
1229 \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
1230 \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
1231 \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
1232 \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
1233 \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
1234 \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
1235 \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
1236 \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
1237 }%
1238 \endgroup
1239 \ltx@onelevel@sanitize\AllBytesString

1240 \def\Test#1#2#3{%
1241   \begingroup
1242     \expandafter\expandafter\expandafter\def
1243     \expandafter\expandafter\expandafter\TestResult
1244     \expandafter\expandafter\expandafter{%
1245       #1{#2}%
1246     }%
1247     \ifx\TestResult#3%
1248     \else
1249       \newlinechar=10 %
1250       \msg{Expect:^^J#3}%
1251       \msg{Result:^^J\TestResult}%
1252       \errmessage{\string#2 -\string#1-> \string#3}%
1253     \fi
1254   \endgroup
1255 }

1256 \def\test#1#2#3{%
1257   \edef\TestFrom{#2}%
1258   \edef\TestExpect{#3}%
1259   \ltx@onelevel@sanitize\TestExpect
1260   \Test#1\TestFrom\TestExpect
1261 }

1262 \test\pdf@unescapehex{74657374}{test}
1263 \begingroup
1264   \catcode0=12 %
1265   \catcode1=12 %
1266   \test\pdf@unescapehex{740074017400740174}{t^^t^^t^^t^^t}%
1267 \endgroup
1268 \Test\pdf@escapehex\AllBytes\AllBytesHex
1269 \Test\pdf@unescapehex\AllBytesHex\AllBytes
1270 \Test\pdf@escapename\AllBytes\AllBytesName
1271 \Test\pdf@escapestring\AllBytes\AllBytesString

1272 \csname @@end\endcsname\end
1273 </test-escape>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/oberdiek/pdftexcmds.dtx](https://ctan.org/ctan/packages/macros/latex/contrib/oberdiek/pdftexcmds.dtx) The source file.

[CTAN:macros/latex/contrib/oberdiek/pdftexcmds.pdf](https://ctan.org/ctan/packages/macros/latex/contrib/oberdiek/pdftexcmds.pdf) Documentation.

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

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

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

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

4.2 Bundle installation

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

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

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

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

4.3 Package installation

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

```
tex pdftexcmds.dtx
```

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

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

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

4.4 Refresh file name databases

If your T_EX distribution (teT_EX, miK_TE_X, ...) relies on file name databases, you must refresh these. For example, teT_EX users run `texhash` or `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 L^AT_EX. The .dtx chooses its action depending on the format:

plain T_EX: Run docstrip and extract the files.

L^AT_EX: Generate the documentation.

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

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

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

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

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

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

```
pdflatex pdftexcmds.dtx
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 LuaT_EX 0.36.

[2009/09/22 v0.5]

- \pdf@primitive, \pdf@ifprimitive added.
- X_YT_EX'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.

6 Index

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

| Symbols | | | |
|----------------------------------|--|--------------------------------|--|
| <code>\#</code> | ... 945, 1044, 1091, 1123, 1191, 1217 | <code>\}</code> | ... 944, 1043, 1090, 1122 |
| <code>\%</code> | ... 1021, 1215 | <code>\^</code> | ... 1124 |
| <code>\&</code> | ... 474, 476 | <code>\ </code> | ... 1214, 1216 |
| <code>\(</code> | ... 1223 | <code>\~</code> | ... 1218 |
| <code>\)</code> | ... 1223 | Numbers | |
| <code>\@</code> | ... 946, 1019, 1045, 1092, 1125, 1135 | <code>\0</code> | ... 506, 610, 1221, 1222, 1223 |
| <code>\@PackageError</code> | ... 453 | <code>\1</code> | ... 1228 |
| <code>\@PackageInfoNoLine</code> | ... 152, 154, 270, 282, 287, 367, 371, 373 | <code>\2</code> | ... 1229, 1230, 1231, 1232 |
| <code>\@PackageWarning</code> | ... 416 | <code>\3</code> | ... 1233, 1234, 1235, 1236 |
| <code>\@ehc</code> | ... 457 | A | |
| <code>\@firstofone</code> | ... 954, 957 | <code>\advance</code> | ... 984, 992, 1007, 1164 |
| <code>\@gobble</code> | ... 951, 959 | <code>\aftergroup</code> | ... 29 |
| <code>\@iwhilenum</code> | ... 1141, 1144, 1146 | <code>\AllBytes</code> | ... 1152, 1158, 1161, 1212, 1268, 1269, 1270, 1271 |
| <code>\@nil</code> | ... 477, 478, 480, 483, 487, 491, 499, 532, 541, 625 | <code>\AllBytesFromName</code> | ... 1212 |
| <code>\@undefined</code> | ... 58, 269, 1095, 1100, 1104 | <code>\AllBytesHex</code> | ... 1167, 1185, 1188, 1268, 1269 |
| <code>\@whilenum</code> | ... 1138, 1155 | <code>\AllBytesHexLC</code> | ... 1187 |
| <code>\@</code> | ... 753, 756, 1020, 1219, 1226 | <code>\AllBytesName</code> | ... 1192, 1211, 1270 |
| <code>\{</code> | ... 943, 1042, 1089, 1121 | | |

| | | | |
|---|---|---|---|
| <code>\AllBytesString</code> | 1239, 1271 | G | |
| B | | <code>\gdef</code> | 1152 |
| <code>\body</code> | 963, 967 | H | |
| C | | <code>\hbox</code> | 1078, 1079, 1084, 1085 |
| <code>\catcode</code> | 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 69, 70, 72, 73, 74, 78, 79, 80, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 97, 99, 474, 476, 506, 610, 943, 944, 945, 946, 981, 990, 998, 1002, 1019, 1020, 1021, 1042, 1043, 1044, 1045, 1089, 1090, 1091, 1092, 1121, 1122, 1123, 1124, 1125, 1135, 1154, 1191, 1216, 1217, 1218, 1219, 1264, 1265 | I | |
| <code>\chardef</code> | 184, 186 | <code>\if</code> | 1080, 1081, 1082 |
| <code>\count</code> | 404, 406 | <code>\ifcase</code> | 378, 411, 473 |
| <code>\count@</code> | 948, 977, 981, 983, 984, 988, 990, 991, 992, 996, 998, 1001, 1002, 1006, 1007, 1136, 1153, 1155, 1156, 1157, 1164 | <code>\ifeof</code> | 182, 183 |
| <code>\countdef</code> | 948, 1136 | <code>\ifluatex</code> | 150, 215, 361, 422 |
| <code>\csname</code> | 14, 21, 50, 66, 76, 133, 136, 159, 162, 164, 178, 196, 204, 238, 243, 246, 247, 260, 262, 265, 284, 289, 295, 298, 304, 306, 315, 350, 427, 430, 461, 464, 947, 950, 953, 956, 1011, 1038, 1086, 1118, 1272 | <code>\ifnum</code> | 182, 216, 264, 267, 323, 353, 385, 392, 468, 469, 504, 611, 983, 991, 998, 1006, 1111, 1139, 1145, 1157 |
| D | | <code>\ifpdf</code> | 369 |
| <code>\delimiter</code> | 323, 329, 347 | <code>\ifx</code> | 15, 18, 21, 50, 58, 61, 133, 136, 159, 178, 196, 204, 238, 245, 260, 262, 265, 284, 289, 295, 303, 315, 330, 331, 334, 336, 427, 430, 451, 461, 481, 492, 947, 950, 953, 956, 1011, 1061, 1067, 1081, 1082, 1095, 1100, 1102, 1104, 1247 |
| <code>\directlua</code> | 217, 219, 509, 516, 521, 528, 537, 544, 549, 554, 559, 564, 569, 576, 581, 586, 591, 596, 601, 606 | <code>\immediate</code> | 23, 52, 212, 1050, 1094, 1131 |
| <code>\do</code> | 1138, 1155 | <code>\input</code> | 137, 431, 1012, 1046, 1093, 1127 |
| E | | <code>\iterate</code> | 964, 966, 968 |
| <code>\empty</code> | 17, 18 | L | |
| <code>\end</code> | 1039, 1086, 1118, 1272 | <code>\lccode</code> | 1156 |
| <code>\endcsname</code> | 14, 21, 50, 66, 76, 133, 136, 159, 162, 164, 178, 196, 204, 238, 243, 246, 247, 260, 262, 265, 284, 289, 295, 298, 304, 306, 315, 350, 427, 430, 461, 464, 947, 950, 953, 956, 1011, 1038, 1086, 1118, 1272 | <code>\LoadCommand</code> | 1012, 1022 |
| <code>\endinput</code> | 29, 129 | <code>\loop</code> | 962, 978, 989, 997 |
| <code>\endlinechar</code> | 4, 35, 71, 77, 89, 223 | <code>\lowercase</code> | 1160, 1186 |
| <code>\errmessage</code> | 1000, 1064, 1068, 1107, 1114, 1252 | <code>\ltx@backslashchar</code> | 367, 371, 374, 1198 |
| <code>\errorcontextlines</code> | 1126 | <code>\ltx@ccclv</code> | 404, 406 |
| <code>\escapechar</code> | 128, 131, 242, 297, 1049, 1130 | <code>\ltx@firstoftwo</code> | 324, 354, 386 |
| <code>\expected</code> | 1100, 1102, 1110, 1111 | <code>\ltx@gobble</code> | 1148, 1212 |
| F | | <code>\ltx@ifUndefined</code> | 180, 366 |
| <code>\foobar</code> | 1076, 1077 | <code>\ltx@one</code> | 370, 385, 392, 393 |
| <code>\foobar@hbox</code> | 1079 | <code>\ltx@onelevel@sanitize</code> | 443, 1185, 1211, 1239, 1259 |
| | | <code>\ltx@percentchar</code> | 1215 |
| | | <code>\ltx@ReturnAfterElseFi</code> | 482, 493 |
| | | <code>\ltx@ReturnAfterFi</code> | 486, 498 |
| | | <code>\ltx@secondoftwo</code> | 326, 356, 380, 388 |
| | | <code>\ltx@zero</code> | 365, 379, 395, 494 |
| | | <code>\luaescapestring</code> | 510, 511, 517, 522, 530, 539, 545, 550, 555, 560, 565, 570, 571, 572, 577, 582, 587, 597, 623 |
| | | <code>\luatexversion</code> | 216, 468, 469, 504 |
| | | M | |
| | | <code>\meaning</code> | 241, 243, 298, 301, 317, 353 |
| | | <code>\MessageBreak</code> | 271, 417, 454, 455 |
| | | <code>\msg</code> | 1047, 1062, 1070, 1094, 1096, 1098, 1103, 1105, 1110, 1112, 1128, 1250, 1251 |
| | | N | |
| | | <code>\newlinechar</code> | 222, 223, 1249 |
| | | <code>\next</code> | 968, 970, 972 |
| | | <code>\number</code> | 128, 570, 571, 1003, 1098, 1110 |

| | |
|---------------------------------|---|
| P | |
| \PackageInfo | 26 |
| \par | 1083, 1084, 1085 |
| \pdf@draftmode | 4, 379, 391 |
| \pdf@escapehex | 3, 170, 515, 1268 |
| \pdf@escapehexnative | 170, 520 |
| \pdf@escapename | 548, 1270 |
| \pdf@escapenamenative | 553 |
| \pdf@escapestring | 543, 1271 |
| \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, 1057, 1060 |
| \pdf@lastsystemexit | 605 |
| \pdf@lastsystemstatus | 600 |
| \pdf@mdfivesum | 3, 207, 208, 575 |
| \pdf@mdfivesumnative | 208, 580 |
| \pdf@pipe | 5, 611 |
| \pdf@primitive .. | 4, 251, 269, 271, 283 |
| \pdf@setdraftmode | 4, 402, 417 |
| \pdf@shellescape | 4, 184, 186, 191, 590, 1095, 1098, 1104, 1111 |
| \pdf@strcmp | 3, 353, 508 |
| \pdf@system | 4, 211, 595 |
| \pdf@unescapehex | 3, 172, 525, 1262, 1266, 1269 |
| \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, 628 |
| \pdf@texcmds@DecodeA | 478, 480 |
| \pdf@texcmds@DecodeB | 483, 491 |
| \pdf@texcmds@directlua | 216, 224, 438, 445, 612, 621 |
| \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, 623 |
| \pdf@texcmds@PatchDecode | 477, 526, 535, 619 |
| \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, 620 |
| \pdf@tex@revision | 267 |
| \pdf@tex@version | 182, 264 |
| \ProvidesPackage | 19, 67 |
| R | |
| \RangeCatcodeCheck | 995, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034 |
| \RangeCatcodeInvalid | 987, 1015, 1016, 1017, 1018 |
| \repeat | 962, 974, 985, 993, 1008 |
| \RequirePackage | 145, 146, 147, 148, 436 |
| \RestoreCatcodes .. | 976, 979, 980, 1035 |
| \romannumeral | 526, 535, 619 |
| S | |
| \space | 272, 283, 288, 1001, 1002, 1010, 1137, 1158 |
| \str | 1056, 1062, 1064, 1068, 1070 |
| T | |
| \Test | 1014, 1037, 1240, 1260, 1268, 1269, 1270, 1271 |
| \test | 1053, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1256, 1262, 1266 |
| \TestExpect | 1258, 1259, 1260 |
| \TestFrom | 1257, 1260 |
| \TestResult | 1243, 1247, 1251 |
| \the | 77, 78, 79, 80, 81, 82, 83, 84, 97, 406, 527, 536, 620, 981, 1001, 1002 |
| \TMP@EnsureCode | 94, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125 |
| \TMP@RequirePackage | 134, 140, 141, 142, 143, 428, 434 |
| \toksdef | 462 |
| W | |
| \write ... | 23, 52, 212, 1050, 1094, 1131 |
| X | |
| \x ... | 14, 15, 18, 22, 26, 28, 51, 56, 66, 75, 87, 240, 241, 245, 298, 303, 405, 408, 442, 443, 451, 455 |
| Y | |
| \y | 243, 245, 299, 301, 303, 444, 451, 456 |
| Z | |
| \z | 300, 301 |