

The pdf_{TeX}cmds package

Heiko Oberdiek
<heiko.oberdiek at gmail.com>

2011/04/22 v0.16

Abstract

Lua_{TeX} provides most of the commands of pdf_{TeX} 1.40. However a number of utility functions are removed. This package tries to fill the gap and implements some of the missing primitive using Lua.

Contents

1	Documentation	2
1.1	General principles	3
1.2	Macros	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	7
2.3	Load packages	8
2.4	Without Lua _{TeX}	8
2.5	<code>\pdf@primitive</code> , <code>\pdf@ifprimitive</code>	9
2.5.1	Using Lua _{TeX} 's <code>tex.enableprimitives</code>	9
2.5.2	Trying various names to find the primitives	10
2.5.3	Result	11
2.6	X _{TeX}	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>	25
3.3	Test for <code>\pdf@shellescape</code>	26
3.4	Test for escape functions	26
4	Installation	29
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

5	History	30
	[2007/11/11 v0.1]	30
	[2007/11/12 v0.2]	31
	[2007/12/12 v0.3]	31
	[2009/04/10 v0.4]	31
	[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]	32
	[2011/04/22 v0.16]	32
6	Index	32

1 Documentation

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

- `\pdfstrcmp`
- `\pdfunescapehex`
- `\pdfescapehex`
- `\pdfescapename`
- `\pdfescapestring`
- `\pdffilesize`
- `\pdffilemoddate`
- `\pdffiledump`
- `\pdfmdfivesum`
- `\immediate\write18`

The original names of the primitives cannot be used:

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

```

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

```

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

1.1 General principles

Naming convention: Usually this package defines a macro `\pdf@⟨cmd⟩` if pdfTeX provides `\pdf⟨cmd⟩`.

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

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

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

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

1.2 Macros

```
\pdf@strcmp {⟨stringA⟩} {⟨stringB⟩}
```

Same as `\pdfstrcmp{⟨stringA⟩}{⟨stringB⟩}`.

```
\pdf@unescapehex {⟨string⟩}
```

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

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

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

```
\pdf@filesize {⟨filename⟩}
```

Same as `\pdffilesize{⟨filename⟩}`.

```
\pdf@filemoddate {⟨filename⟩}
```

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

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

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

`\pdf@mdfivesum {<string>}`

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

`\pdf@filemdfivesum {<filename>}`

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

`\pdf@draftmode`

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

`\pdf@ifdraftmode {<true>} {<false>}`

If `\pdfdraftmode` is available and enabled, `<true>` is called, otherwise `<false>` is executed.

`\pdf@setdraftmode {<value>}`

Macro `\pdf@setdraftmode` expects the number zero or one as `<value>`. Zero deactivates the mode and one enables the draft mode. The macro does not have an effect, if the feature `\pdfdraftmode` is not available.

`\pdf@shellescape`

Same as `\pdfshellescape`. It is or expands to 1 if external commands can be executed and 0 otherwise. In pdfTeX external commands must be enabled first by command line option or configuration option. In LuaTeX option `--safer` disables the execution of external commands.

In LuaTeX before 0.68.0 `\pdf@shellescape` is not available due to a bug in `os.execute()`. The argumentless form crashes in some circumstances with segmentation fault. (It is fixed in version 0.70.0, revision 4167 of LuaTeX. and packported to some version of 0.67.0).

Hints for usage:

- `\pdf@shellescape` might be a numerical constant, expands to the primitive, or expands to a plain number. Therefore use it in contexts where these differences does not matter.
- Use in comparisons, e.g.:

```
\ifnum\pdf@shellescape=0 ...
```
- Print the number: `\number\pdf@shellescape`

`\pdf@system {<cmdline>}`

It is a wrapper for `\immediate\write18` in pdfTeX or `os.execute` in LuaTeX.

In theory `os.execute` returns a status number. But its meaning is quite undefined. Are there some reliable properties? Does it make sense to provide an user interface to this status exit code?

```
\pdf@primitive \cmd
```

Same as `\pdfprimitive` in pdfTeX or LuaTeX. In XeTeX the primitive is called `\primitive`. Despite the current definition of the command `\cmd`, it's meaning as primitive is used.

```
\pdf@ifprimitive \cmd
```

Same as `\ifpdfprimitive` in pdfTeX or LuaTeX. XeTeX calls it `\ifprimitive`. It is a switch that checks if the command `\cmd` has it's primitive meaning.

1.2.1 Additional macro: `\pdf@isprimitive`

```
\pdf@isprimitive \cmd1 \cmd2 {<true>} {<false>}
```

If `\cmd1` has the primitive meaning given by the primitive name of `\cmd2`, then the argument `<true>` is executed, otherwise `<false>`. The macro `\pdf@isprimitive` is expandable. Internally it checks the result of `\meaning` and is therefore available for all TeX variants, even the original TeX. Example with L^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@pdftexcmds.sty\endcsname
15 \ifx\x\relax % plain-TeX, first loading
16 \else
17 \def\empty{}%
18 \ifx\x\empty % LaTeX, first loading,
19 % variable is initialized, but \ProvidesPackage not yet seen
20 \else
21 \expandafter\ifx\csname PackageInfo\endcsname\relax
22 \def\x#1#2{%
23 \immediate\write-1{Package #1 Info: #2.}%
24 }%
25 \else
26 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
27 \fi
28 \x{pdftexcmds}{The package is already loaded}%
29 \aftergroup\endinput
30 \fi
31 \fi
32 \endgroup%

```

Package identification:

```

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

```

```

65 \fi
66 \expandafter\x\csname ver@pdftexcmds.sty\endcsname
67 \ProvidesPackage{pdftexcmds}%
68 [2011/04/22 v0.16 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{inwarerr}[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{inwarerr}[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
163       \expandafter##\expandafter{%
164         \csname pdf#1\endcsname
165       }%
166     \fi
167   }%
168   \pdftexcmds@temp{strcmp}%
169   \pdftexcmds@temp{escapehex}%
170   \let\pdf@escapehexnative\pdf@escapehex
171   \pdftexcmds@temp{unescapehex}%
172   \let\pdf@unescapehexnative\pdf@unescapehex
173   \pdftexcmds@temp{escapestring}%
174   \pdftexcmds@temp{escapename}%
175   \pdftexcmds@temp{filesize}%
176   \pdftexcmds@temp{filemoddate}%
177   \begingroup\expandafter\expandafter\expandafter\endgroup
178   \expandafter\ifx\csname pdfshellescape\endcsname\relax
179     \pdftexcmds@nopdftex
180     \ltx@ifundefined{pdftexversion}{%
181       }{%

```



```

182     \ifnum\pdftexversion>120 % 1.21a supports \ifeof18
183     \ifeof18 %
184     \chardef\pdf@shellescape=0 %
185     \else
186     \chardef\pdf@shellescape=1 %
187     \fi
188     \fi
189   }%
190 \else
191   \def\pdf@shellescape{%
192     \pdfshellescape
193   }%
194 \fi
195 \begingroup\expandafter\expandafter\expandafter\endgroup
196 \expandafter\ifx\csname pdffiledump\endcsname\relax
197   \pdftexcmds@nopdftex
198 \else
199   \def\pdf@filedump#1#2#3{%
200     \pdffiledump offset#1 length#2{#3}%
201   }%
202 \fi
203 \begingroup\expandafter\expandafter\expandafter\endgroup
204 \expandafter\ifx\csname pdfmdfivesum\endcsname\relax
205   \pdftexcmds@nopdftex
206 \else
207   \def\pdf@mdfivesum#\pdfmdfivesum}%
208   \let\pdf@mdfivesumnative\pdf@mdfivesum
209   \def\pdf@filemdfivesum#\pdfmdfivesum file}%
210 \fi
211 \def\pdf@system#{%
212   \immediate\write18%
213 }%
214 \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
\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}{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_qTeX

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

```

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{stricmp}%

```

2.7 \pdf@isprimitive

```

313 \def\pdf@isprimitive{%
314 \begingroup\expandafter\expandafter\expandafter\endgroup
315 \expandafter\ifx\csname pdf@stricmp\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\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@cc1v=#1\relax
405     \edef\x{\endgroup
406       \noexpand\pdftexcmds@@setdraftmode{\the\count\ltx@cc1v}}%
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{2011/04/22 v0.16}%
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 \ifnum\luatexversion>40 %
468 \ifnum\luatexversion<66 %
469 \def\pdfTexcmds@Patch{1}%
470 \fi
471 \fi

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

```

`\pdfTexcmds@PatchDecode`

```

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

```

`\pdfTexcmds@DecodeA`

```

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

```

```

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

```

\pdftexcmds@DecodeB

```

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

502 \fi

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

```

\pdf@strcmp

```

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

513 \pdf@isprimitive

```

\pdf@escapehex

```

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

```

\pdf@escapehexnative

```

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

```

\pdf@unescapehex

```

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

```

```

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

```

\pdf@unescapehexnative

```

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

```

\pdf@escapestring

```

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

```

\pdf@escapename

```

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

```

\pdf@escapenamename

```

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

```

\pdf@filesize

```

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

```

\pdf@filemoddate

```

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

```

\pdf@filedump

```

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

```


`\pdf@mdfivesum`

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

`\pdf@mdfivesumnative`

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

`\pdf@filemdfivesum`

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

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

```
589 \ifnum\luatexversion<68 %
590 \else
591   \def\pdf@shellescape{%
592     \directlua0{%
593       oberdiek.pdftexcmds.shellescape()%
594     }%
595   }%
596 \fi
```

`\pdf@system`

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

`\pdf@lastsystemstatus`

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

`\pdf@lastsystemexit`

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

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

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

```
613 \ifnum0%
614   \pdftexcmds@directlua{%
615     if io.popen then %
616       tex.write("1")%
617     end%
618   }%
```

```

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

630 \pdftexcmds@AtEnd%
631 \</package>

```

2.11 Lua module

```

632 <*lua>

633 module("oberdiek.pdftexcmds", package.seeall)
634 local systemexitstatus
635 function getversion()
636   tex.write("2011/04/22 v0.16")
637 end
638 function strcmp(A, B)
639   if A == B then
640     tex.write("0")
641   elseif A < B then
642     tex.write("-1")
643   else
644     tex.write("1")
645   end
646 end
647 local function utf8_to_byte(str)
648   local i = 0
649   local n = string.len(str)
650   local t = {}
651   while i < n do
652     i = i + 1
653     local a = string.byte(str, i)
654     if a < 128 then
655       table.insert(t, string.char(a))
656     else
657       if a >= 192 and i < n then
658         i = i + 1
659         local b = string.byte(str, i)
660         if b < 128 or b >= 192 then
661           i = i - 1
662         elseif a == 194 then
663           table.insert(t, string.char(b))
664         elseif a == 195 then
665           table.insert(t, string.char(b + 64))
666         end
667       end
668     end
669   end
670   return table.concat(t)
671 end
672 function escapehex(str, mode)
673   if mode == "byte" then
674     str = utf8_to_byte(str)
675   end
676   tex.write((string.gsub(str, ".",

```

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

760     return nil
761   end
762 )))
763 end

```

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

```

764 function escapename(str, mode)
765   if mode == "byte" then
766     str = utf8_to_byte(str)
767   end
768   tex.write((string.gsub(str, ".",
769     function (ch)
770       local b = string.byte(ch)
771       if b == 0 then

```

In Lua 5.0 `nil` could be used for the empty string, But `nil` returns the match in Lua 5.1, thus we use the empty string explicitly.

```

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

```

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

```

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

```

```

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

```

```

853     str = utf8_to_byte(str)
854 end
855 escapehex(md5.sum(str))
856 end
857 function filemdfivesum(filename)
858     local foundfile = kpse.find_file(filename, "tex", true)
859     if foundfile then
860         local filehandle = io.open(foundfile, "r")
861         if filehandle then
862             local contents = filehandle:read("*a")
863             escapehex(md5.sum(contents))
864         end
865     end
866 end
867 function shellescape()
868     if os.execute then
869         if status
870             and status.luatex_version
871             and status.luatex_version >= 68 then
872             tex.write(os.execute())
873         else
874             local result = os.execute()
875             if result == 0 then
876                 tex.write("0")
877             else
878                 if result == nil then
879                     tex.write("0")
880                 else
881                     tex.write("1")
882                 end
883             end
884         end
885     else
886         tex.write("0")
887     end
888 end
889 function system(cmdline)
890     systemexitstatus = nil
891     texio.write_nl("log", "system(" .. cmdline .. ") ")
892     if os.execute then
893         texio.write("log", "executed.")
894         systemexitstatus = os.execute(cmdline)
895     else
896         texio.write("log", "disabled.")
897     end
898 end
899 function lastsystemstatus()
900     local result = tonumber(systemexitstatus)
901     if result then
902         local x = math.floor(result / 256)
903         tex.write(result - 256 * math.floor(result / 256))
904     end
905 end
906 function lastsystemexit()
907     local result = tonumber(systemexitstatus)
908     if result then
909         tex.write(math.floor(result / 256))
910     end
911 end
912 function pipe(cmdline, patch)
913     local result
914     systemexitstatus = nil

```

```

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

```

3 Test

3.1 Catcode checks for loading

```

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

```

```

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

```



```

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

```

3.2 Test for \pdf@isprimitive

```

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

```

```

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

```

3.3 Test for \pdf@shellescape

```

1096 <*test-shell>
1097 \catcode'\{=1 %
1098 \catcode'\}=2 %
1099 \catcode'\#=6 %
1100 \catcode'\@=11 %
1101 \input pdftexcmds.sty\relax
1102 \def\msg#\{\immediate\write16}
1103 \def\MaybeEnd{}
1104 \ifx\luatexversion\UnDeFiNeD
1105 \else
1106   \ifnum\luatexversion<68 %
1107     \ifx\pdf@shellescape\undefined
1108       \msg{SHELL=U}%
1109       \msg{OK (LuaTeX < 0.68)}%
1110     \else
1111       \msg{SHELL=defined}%
1112       \errmessage{Failed (LuaTeX < 0.68)}%
1113     \fi
1114     \def\MaybeEnd{\csname @@end\endcsname\end}%
1115   \fi
1116 \fi
1117 \MaybeEnd
1118 \ifx\pdf@shellescape\undefined
1119   \msg{SHELL=U}%
1120 \else
1121   \msg{SHELL=\number\pdf@shellescape}%
1122 \fi
1123 \ifx\expected\undefined
1124 \else
1125   \ifx\expected\relax
1126     \msg{EXPECTED=U}%
1127     \ifx\pdf@shellescape\undefined
1128       \msg{OK}%
1129     \else
1130       \errmessage{Failed}%
1131     \fi
1132   \else
1133     \msg{EXPECTED=\number\expected}%
1134     \ifnum\pdf@shellescape=\expected\relax
1135       \msg{OK}%
1136     \else
1137       \errmessage{Failed}%
1138     \fi
1139   \fi
1140 \fi
1141 \csname @@end\endcsname\end
1142 </test-shell>

```

3.4 Test for escape functions

```

1143 <*test-escape>
1144 \catcode'\{=1 %
1145 \catcode'\}=2 %
1146 \catcode'\#=6 %
1147 \catcode'\^=7 %
1148 \catcode'\@=11 %
1149 \errorcontextlines=1000 %
1150 \input pdftexcmds.sty\relax
1151 \def\msg#1{%

```

```

1152 \begingroup
1153   \escapechar=92 %
1154   \immediate\write16{#1}%
1155 \endgroup
1156 }

1157 \begingroup
1158   \catcode'\@=11 %
1159   \countdef\count@=255 %
1160   \def\space{ }%
1161   \long\def\@whilenum#1\do #2{%
1162     \ifnum #1\relax
1163       #2\relax
1164       \@iwhilenum{#1\relax#2\relax}%
1165     \fi
1166   }%
1167   \long\def\@iwhilenum#1{%
1168     \ifnum #1%
1169       \expandafter\@iwhilenum
1170     \else
1171       \expandafter\ltx@gobble
1172     \fi
1173     {#1}%
1174   }%
1175   \gdef\AllBytes{}%
1176   \count@=0 %
1177   \catcode0=12 %
1178   \@whilenum\count@<256 \do{%
1179     \lccode0=\count@
1180     \ifnum\count@=32 %
1181       \xdef\AllBytes{\AllBytes\space}%
1182     \else
1183       \lowercase{%
1184         \xdef\AllBytes{\AllBytes^^@}%
1185       }%
1186     \fi
1187     \advance\count@ by 1 %
1188   }%
1189 \endgroup

1190 \def\AllBytesHex{%
1191   000102030405060708090A0B0C0D0E0F%
1192   101112131415161718191A1B1C1D1E1F%
1193   202122232425262728292A2B2C2D2E2F%
1194   303132333435363738393A3B3C3D3E3F%
1195   404142434445464748494A4B4C4D4E4F%
1196   505152535455565758595A5B5C5D5E5F%
1197   606162636465666768696A6B6C6D6E6F%
1198   707172737475767778797A7B7C7D7E7F%
1199   808182838485868788898A8B8C8D8E8F%
1200   909192939495969798999A9B9C9D9E9F%
1201   A0A1A2A3A4A5A6A7A8A9AAABACADAFAF%
1202   B0B1B2B3B4B5B6B7B8B9BABBBBCDBEBF%
1203   C0C1C2C3C4C5C6C7C8C9CACBCCDCECF%
1204   D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
1205   E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
1206   F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF%
1207 }
1208 \ltx@onelevel@sanitize\AllBytesHex
1209 \expandafter\lowercase\expandafter{%
1210   \expandafter\def\expandafter\AllBytesHexLC
1211     \expandafter{\AllBytesHex}%
1212 }
1213 \begingroup

```

```

1214 \catcode'\#=12 %
1215 \xdef\AllBytesName{%
1216 #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
1217 #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
1218 #20!"#$%&'#28#29**,-.#2F%
1219 0123456789:;#3C=#3E?%
1220 @ABCDEFGHIJKLMNO%
1221 PQRSTUVWXYZ#5B\ltx@backslashchar#5D^_%
1222 'abcdefghijklmno%
1223 pqrstuvwxyz#7B|#7D\string~#7F%
1224 #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
1225 #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
1226 #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
1227 #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
1228 #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
1229 #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
1230 #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
1231 #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
1232 }%
1233 \endgroup
1234 \ltx@onelevel@sanitize\AllBytesName
1235 \edef\AllBytesFromName{\expandafter\ltx@gobble\AllBytes}
1236 \begingroup
1237 \def\|{|}%
1238 \edef%\{\ltx@percentchar}%
1239 \catcode'\|=0 %
1240 \catcode'\#=12 %
1241 \catcode'\~=12 %
1242 \catcode'\|=12 %
1243 |xdef|AllBytesString{%
1244 \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
1245 \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
1246 \040!"#$%&'(\)*+,-./%
1247 0123456789:;<=>?%
1248 @ABCDEFGHIJKLMNO%
1249 PQRSTUVWXYZ[\]^_%
1250 'abcdefghijklmno%
1251 pqrstuvwxyz{|}~\177%
1252 \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
1253 \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
1254 \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
1255 \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
1256 \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
1257 \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
1258 \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
1259 \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
1260 }%
1261 |endgroup
1262 \ltx@onelevel@sanitize\AllBytesString
1263 \def\Test#1#2#3{%
1264 \begingroup
1265 \expandafter\expandafter\expandafter\def
1266 \expandafter\expandafter\expandafter\TestResult
1267 \expandafter\expandafter\expandafter{%
1268 #1{#2}%
1269 }%
1270 \ifx\TestResult#3%
1271 \else
1272 \newlinechar=10 %
1273 \msg{Expect:^^J#3}%
1274 \msg{Result:^^J\TestResult}%
1275 \errmessage{\string#2 -\string#1-> \string#3}%

```

```

1276 \fi
1277 \endgroup
1278 }
1279 \def\test#1#2#3{%
1280 \edef\TestFrom{#2}%
1281 \edef\TestExpect{#3}%
1282 \ltx@onelevel@sanitize\TestExpect
1283 \Test#1\TestFrom\TestExpect
1284 }
1285 \test\pdf@unescapehex{74657374}{test}
1286 \begingroup
1287 \catcode0=12 %
1288 \catcode1=12 %
1289 \test\pdf@unescapehex{740074017400740174}{t^^@t^^At^^t^^At}%
1290 \endgroup
1291 \Test\pdf@escapehex\AllBytes\AllBytesHex
1292 \Test\pdf@unescapehex\AllBytesHex\AllBytes
1293 \Test\pdf@escapename\AllBytes\AllBytesName
1294 \Test\pdf@escapestring\AllBytes\AllBytesString
1295 \csname @@end\endcsname\end
1296 </test-escape>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

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

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

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

TDS refers to the standard “A Directory Structure for \TeX 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 \TeX :

```
tex pdftexcmds.dtx
```

¹<http://ftp.ctan.org/tex-archive/>

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

```

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

```

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

4.4 Refresh file name databases

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

4.5 Some details for the interested

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 \LaTeX . The `.dtx` chooses its action depending on the format:

plain \TeX : Run `docstrip` and extract the files.

\LaTeX : Generate the documentation.

If you insist on using \LaTeX for `docstrip` (really, `docstrip` does not need \LaTeX), then inform the autodetect routine about your intention:

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

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

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

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

An example follows how to generate the documentation with pdf \LaTeX :

```

pdflatex 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 LuaTeX 0.36.

[2009/09/22 v0.5]

- `\pdf@primitive`, `\pdf@ifprimitive` added.
- XeTeX's variants are detected for `\pdf@shellescape`, `\pdf@strcmp`, `\pdf@primitive`, `\pdf@ifprimitive`.

[2009/09/23 v0.6]

- Macro `\pdf@isprimitive` added.

[2009/12/12 v0.7]

- Short info shortened.

[2010/03/01 v0.8]

- Required date for package `ifluatex` updated.

[2010/04/01 v0.9]

- Use `\ifeof18` for defining `\pdf@shellescape` between pdfTeX 1.21a (inclusive) and 1.30.0 (exclusive).

[2010/11/04 v0.10]

- `\pdf@draftmode`, `\pdf@ifdraftmode` and `\pdf@setdraftmode` added.

[2010/11/11 v0.11]

- Missing `\RequirePackage` for package `ifpdf` added.

[2011/01/30 v0.12]

- Already loaded package files are not input in plain TeX.

[2011/03/04 v0.13]

- Improved Lua function `shellescape` that also uses the result of `os.execute()` (thanks to Philipp Stephani).

[2011/04/10 v0.14]

- Version check of loaded module added.
- Patch for bug in LuaTeX between 0.40.6 and 0.65 that is fixed in revision 4096.

[2011/04/16 v0.15]

- LuaTeX: `\pdfshellescape` is only supported for version 0.70.0 and higher due to a bug, `os.execute()` crashes in some circumstances. Fixed in LuaTeX beta-0.70.0, revision 4167.

[2011/04/22 v0.16]

- Previous fix was not working due to a wrong catcode of digit zero (due to easily support the old `\directlua0`). The version border is lowered to 0.68, because some beta-0.67.0 seems also to work.

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> . . .	<i>953, 1052, 1099, 1146, 1214, 1240</i>
<code>\%</code>	<i>1029, 1238</i>
<code>\&</code>	<i>473, 475</i>
<code>\(</code>	<i>1246</i>
<code>\)</code>	<i>1246</i>
<code>\@</code> . . .	<i>954, 1027, 1053, 1100, 1148, 1158</i>
<code>\@PackageError</code>	<i>453</i>
<code>\@PackageInfoNoLine</code>	<i>152,</i> <i>154, 270, 282, 287, 367, 371, 373</i>
<code>\@PackageWarning</code>	<i>416</i>
<code>\@ehc</code>	<i>457</i>
<code>\@firstofone</code>	<i>962, 965</i>
<code>\@gobble</code>	<i>959, 967</i>
<code>\@iwhilenum</code>	<i>1164, 1167, 1169</i>
<code>\@nil</code>	<i>476, 477, 479,</i> <i>482, 486, 490, 498, 531, 540, 627</i>
<code>\@undefined</code>	<i>58, 269, 1107, 1118, 1123, 1127</i>
<code>\@whilenum</code>	<i>1161, 1178</i>
<code>\%</code>	<i>755, 758, 1028, 1242, 1249</i>
<code>\{</code>	<i>951, 1050, 1097, 1144</i>
<code>\}</code>	<i>952, 1051, 1098, 1145</i>
<code>\~</code>	<i>1147</i>
<code>\ </code>	<i>1237, 1239</i>
<code>\^</code>	<i>1241</i>
Numbers	
<code>\0</code>	<i>505, 612, 1244, 1245, 1246</i>
<code>\1</code>	<i>1251</i>
<code>\2</code>	<i>1252, 1253, 1254, 1255</i>
<code>\3</code>	<i>1256, 1257, 1258, 1259</i>
A	
<code>\advance</code>	<i>992, 1000, 1015, 1187</i>
<code>\aftergroup</code>	<i>29</i>
<code>\AllBytes</code>	<i>1175, 1181,</i> <i>1184, 1235, 1291, 1292, 1293, 1294</i>
<code>\AllBytesFromName</code>	<i>1235</i>
<code>\AllBytesHex</code>	<i>1190, 1208, 1211, 1291, 1292</i>
<code>\AllBytesHexLC</code>	<i>1210</i>
<code>\AllBytesName</code>	<i>1215, 1234, 1293</i>
B	
<code>\body</code>	<i>971, 975</i>
C	
<code>\catcode</code>	<i>2, 3, 5, 6, 7, 8, 9, 10, 11,</i> <i>12, 13, 33, 34, 36, 37, 38, 39, 40,</i> <i>41, 42, 43, 44, 45, 46, 47, 48, 49,</i> <i>69, 70, 72, 73, 74, 78, 79, 80, 81,</i> <i>82, 83, 84, 87, 88, 90, 91, 92, 93,</i> <i>97, 99, 473, 475, 505, 612, 951,</i> <i>952, 953, 954, 989, 998, 1006,</i> <i>1010, 1027, 1028, 1029, 1050,</i> <i>1051, 1052, 1053, 1097, 1098,</i> <i>1099, 1100, 1144, 1145, 1146,</i> <i>1147, 1148, 1158, 1177, 1214,</i> <i>1239, 1240, 1241, 1242, 1287, 1288</i>
<code>\chardef</code>	<i>184, 186</i>
<code>\count</code>	<i>404, 406</i>
<code>\count@</code>	<i>956, 985, 989, 991,</i> <i>992, 996, 998, 999, 1000, 1004,</i> <i>1006, 1009, 1010, 1014, 1015,</i> <i>1159, 1176, 1178, 1179, 1180, 1187</i>
<code>\countdef</code>	<i>956, 1159</i>
<code>\csname</code>	<i>14, 21, 50, 66, 76, 133,</i> <i>136, 159, 162, 164, 178, 196,</i> <i>204, 238, 243, 246, 247, 260,</i> <i>262, 265, 284, 289, 295, 298,</i> <i>304, 306, 315, 350, 427, 430,</i> <i>461, 464, 955, 958, 961, 964,</i> <i>1019, 1046, 1094, 1114, 1141, 1295</i>
D	
<code>\delimiter</code>	<i>323, 329, 347</i>
<code>\directlua</code>	<i>217,</i> <i>219, 508, 515, 520, 527, 536,</i> <i>543, 548, 553, 558, 563, 568,</i> <i>575, 580, 585, 592, 598, 603, 608</i>
<code>\do</code>	<i>1161, 1178</i>
E	
<code>\empty</code>	<i>17, 18</i>
<code>\end</code>	<i>1047, 1094, 1114, 1141, 1295</i>

`\endcsname` . 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, 955, 958, 961, 964,
1019, 1046, 1094, 1114, 1141, 1295
`\endinput` 29, 129
`\endlinechar` 4, 35, 71, 77, 89, 223
`\errmessage` 1008,
1072, 1076, 1112, 1130, 1137, 1275
`\errorcontextlines` 1149
`\escapechar`
. . . 128, 131, 242, 297, 1057, 1153
`\expected` 1123, 1125, 1133, 1134

F

`\foobar` 1084, 1085
`\foobar@hbox` 1087

G

`\gdef` 1175

H

`\hbox` 1086, 1087, 1092, 1093

I

`\if` 1088, 1089, 1090
`\ifcase` 378, 411, 472
`\ifEOF` 182, 183
`\ifluatex` 150, 215, 361, 422
`\ifnum` 182, 216, 264, 267,
323, 353, 385, 392, 467, 468,
503, 589, 613, 991, 999, 1006,
1014, 1106, 1134, 1162, 1168, 1180
`\ifpdf` 369
`\ifx` 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, 480,
491, 955, 958, 961, 964, 1019,
1069, 1075, 1089, 1090, 1104,
1107, 1118, 1123, 1125, 1127, 1270
`\immediate` 23, 52, 212, 1058, 1102, 1154
`\input` . 137, 431, 1020, 1054, 1101, 1150
`\iterate` 972, 974, 976

L

`\lccode` 1179
`\LoadCommand` 1020, 1030
`\loop` 970, 986, 997, 1005
`\lowercase` 1183, 1209
`\ltx@backslashchar` 367, 371, 374, 1221
`\ltx@cclv` 404, 406
`\ltx@firstoftwo` 324, 354, 386
`\ltx@gobble` 1171, 1235
`\ltx@ifUndefined` 180, 366
`\ltx@one` 370, 385, 392, 393
`\ltx@onelevel@sanitize`
. 443, 1208, 1234, 1262, 1282
`\ltx@percentchar` 1238
`\ltx@ReturnAfterElseFi` 481, 492

`\ltx@ReturnAfterFi` 485, 497
`\ltx@secondoftwo` 326, 356, 380, 388
`\ltx@zero` 365, 379, 395, 493
`\luaescapestring`
. . . 509, 510, 516, 521, 529, 538,
544, 549, 554, 559, 564, 569,
570, 571, 576, 581, 586, 599, 625
`\luatexversion`
216, 467, 468, 503, 589, 1104, 1106

M

`\MaybeEnd` 1103, 1114, 1117
`\meaning` 241, 243, 298, 301, 317, 353
`\MessageBreak` 271, 417, 454, 455
`\msg` 1055, 1070, 1078, 1102, 1108,
1109, 1111, 1119, 1121, 1126,
1128, 1133, 1135, 1151, 1273, 1274

N

`\newlinechar` 222, 223, 1272
`\next` 976, 978, 980
`\number` 128, 569, 570, 1011, 1121, 1133

P

`\PackageInfo` 26
`\par` 1091, 1092, 1093
`\pdf@draftmode` 4, 379, 391
`\pdf@escapehex` 3, 170, 514, 1291
`\pdf@escapehexnative` 170, 519
`\pdf@escapename` 547, 1293
`\pdf@escapenamenative` 552
`\pdf@escapestring` 542, 1294
`\pdf@filedump` 3, 199, 567
`\pdf@filemdfivesum` 4, 209, 584
`\pdf@filemoddate` 3, 562
`\pdf@filesize` 3, 557
`\pdf@ifdraftmode` 4, 380, 384
`\pdf@ifprimitive` 5, 255, 288
`\pdf@isprimitive` 5,
313, 316, 352, 363, 513, 1065, 1068
`\pdf@lastsystemexit` 607
`\pdf@lastsystemstatus` 602
`\pdf@mdfivesum` 4, 207, 208, 574
`\pdf@mdfivesumnative` 208, 579
`\pdf@pipe` 5, 613
`\pdf@primitive` 5, 251, 269, 271, 283
`\pdf@setdraftmode` 4, 402, 417
`\pdf@shellescape` 4, 184, 186, 191,
589, 1107, 1118, 1121, 1127, 1134
`\pdf@strcmp` 3, 353, 507
`\pdf@system` 4, 211, 597
`\pdf@unescapehex`
. 3, 172, 524, 1285, 1289, 1292
`\pdf@unescapehexnative` 5, 172, 533
`\pdfdraftmode` 383
`\pdffiledump` 200
`\pdfmdfivesum` 207, 209
`\pdfprimitive` 272
`\pdfshellescape` 192
`\pdftexcmds@isprimitive` 320, 322
`\pdftexcmds@setdraftmode` 406, 410
`\pdftexcmds@AtEnd`
. 95, 96, 126, 127, 424, 630

<code>\pdfdoccmds@DecodeA</code>	477, 479		
<code>\pdfdoccmds@DecodeB</code>	482, 490		
<code>\pdfdoccmds@directlua</code>	216, 224, 438, 445, 614, 623		
<code>\pdfdoccmds@draftmode</code>	383, 385, 392, 399		
<code>\pdfdoccmds@equal</code>	323, 329, 347		
<code>\pdfdoccmds@equalcont</code>	338, 344, 345, 350		
<code>\pdfdoccmds@isprimitive</code>	317, 319		
<code>\pdfdoccmds@nopdfTeX</code>	153, 155, 160, 179, 197, 205		
<code>\pdfdoccmds@Patch</code>	466, 472, 529, 538, 625		
<code>\pdfdoccmds@PatchDecode</code>	476, 525, 534, 621		
<code>\pdfdoccmds@setdraftmode</code>	381, 398, 412, 414		
<code>\pdfdoccmds@strip@prefix</code>	235, 241		
<code>\pdfdoccmds@temp</code>	157, 168, 169, 171, 173, 174, 175, 176, 236, 251, 252, 253, 254, 255, 256, 257, 258, 293, 311, 312, 365, 370, 378		
<code>\pdfdoccmds@toks</code>	460, 526, 535, 622		
<code>\pdfTeXrevision</code>	267		
<code>\pdfTeXversion</code>	182, 264		
<code>\ProvidesPackage</code>	19, 67		
R			
<code>\RangeCatcodeCheck</code>	1003, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042		
<code>\RangeCatcodeInvalid</code>	995, 1023, 1024, 1025, 1026		
<code>\repeat</code>	970, 982, 993, 1001, 1016		
<code>\RequirePackage</code>	145, 146, 147, 148, 436		
<code>\RestoreCatcodes</code>	984, 987, 988, 1043		
<code>\romannumeral</code>	525, 534, 621		
S			
<code>\space</code>	272, 283, 288, 1009, 1010, 1018, 1160, 1181		
<code>\str</code>	1064, 1070, 1072, 1076, 1078		
T			
<code>\Test</code>	1022, 1045, 1263, 1283, 1291, 1292, 1293, 1294		
<code>\test</code>	1061, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1279, 1285, 1289		
<code>\TestExpect</code>	1281, 1282, 1283		
<code>\TestFrom</code>	1280, 1283		
<code>\TestResult</code>	1266, 1270, 1274		
<code>\the</code>	77, 78, 79, 80, 81, 82, 83, 84, 97, 406, 526, 535, 622, 989, 1009, 1010		
<code>\TMP@EnsureCode</code>	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		
<code>\TMP@RequirePackage</code>	134, 140, 141, 142, 143, 428, 434		
<code>\toksdef</code>	462		
U			
<code>\UnDeFiNeD</code>	1104		
W			
<code>\write</code>	23, 52, 212, 1058, 1102, 1154		
X			
<code>\x</code>	14, 15, 18, 22, 26, 28, 51, 56, 66, 75, 87, 240, 241, 245, 298, 303, 405, 408, 442, 443, 451, 455		
Y			
<code>\y</code>	243, 245, 299, 301, 303, 444, 451, 456		
Z			
<code>\z</code>	300, 301		