

The pdfTeXcmds package

Heiko Oberdiek
<heiko.oberdiek at gmail.com>

2011/11/29 v0.20

Abstract

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

Contents

1	Documentation	2
1.1	General principles	3
1.2	Macros	3
1.2.1	Strings	3
1.2.2	Files	4
1.2.3	Timekeeping	4
1.2.4	Miscellaneous	5
1.2.5	Additional macro: <code>\pdf@isprimitive</code>	6
1.2.6	Experimental	6
2	Implementation	6
2.1	Reload check and package identification	7
2.2	Catcodes	8
2.3	Load packages	9
2.4	Without LuaTeX	9
2.5	<code>\pdf@primitive</code> , <code>\pdf@ifprimitive</code>	10
2.5.1	Using LuaTeX's <code>tex.enableprimitives</code>	10
2.5.2	Trying various names to find the primitives	11
2.5.3	Result	12
2.6	<code>X_YTeX</code>	12
2.7	<code>\pdf@isprimitive</code>	12
2.8	<code>\pdf@draftmode</code>	13
2.9	Load Lua module	15
2.10	Lua functions	16
2.10.1	Helper macros	16
2.10.2	Strings	17
2.10.3	Files	18
2.10.4	Timekeeping	18
2.10.5	Shell escape	19
2.11	Lua module	20
2.11.1	Strings	20
2.11.2	Files	23
2.11.3	Timekeeping	24
2.11.4	Miscellaneous	25

3	Test	26
3.1	Catcode checks for loading	26
3.2	Test for <code>\pdf@isprimitive</code>	28
3.3	Test for <code>\pdf@shellescape</code>	28
3.4	Test for escape functions	29
4	Installation	32
4.1	Download	32
4.2	Bundle installation	32
4.3	Package installation	32
4.4	Refresh file name databases	33
4.5	Some details for the interested	33
5	Catalogue	34
6	References	34
7	History	34
	[2007/11/11 v0.1]	34
	[2007/11/12 v0.2]	34
	[2007/12/12 v0.3]	34
	[2009/04/10 v0.4]	35
	[2009/09/22 v0.5]	35
	[2009/09/23 v0.6]	35
	[2009/12/12 v0.7]	35
	[2010/03/01 v0.8]	35
	[2010/04/01 v0.9]	35
	[2010/11/04 v0.10]	35
	[2010/11/11 v0.11]	35
	[2011/01/30 v0.12]	35
	[2011/03/04 v0.13]	35
	[2011/04/10 v0.14]	35
	[2011/04/16 v0.15]	35
	[2011/04/22 v0.16]	36
	[2011/06/29 v0.17]	36
	[2011/07/01 v0.18]	36
	[2011/07/28 v0.19]	36
	[2011/11/29 v0.20]	36
8	Index	36

1 Documentation

Some primitives of pdfTeX [1] are not defined by LuaTeX [2]. This package implements macro based solutions using Lua code for the following missing pdfTeX primitives;

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

The original names of the primitives cannot be used:

- The syntax for their arguments cannot easily simulated by macros. The primitives using key words such as `file` (`\pdfmdfivesum`) or `offset` and `length` (`\pdffiledump`) and uses *⟨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

1.2.1 Strings [1, “7.15 Strings”]

```
\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 pdf_TE_X. However pdf_TE_X does not know about characters with codes 256 and larger. Thus the string is treated as byte string, characters with more than eight bits are ignored.

1.2.2 Files [1, “7.18 Files”]

```
\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 _TE_X number.

```
\pdf@mdfivesum {<string>}
```

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

```
\pdf@filemdfivesum {<filename>}
```

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

1.2.3 Timekeeping [1, “7.17 Timekeeping”]

The timekeeping macros are based on Andy Thomas’ work [3].

```
\pdf@resettimer
```

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

```
\pdf@elapsedtime
```

Same as `\pdfelapsedtime`. It behaves like a read-only integer. For printing purposes it can be prefixed by `\the` or `\number`. It measures the time in scaled seconds (seconds multiplied with 65536) since the latest call of `\pdf@resettimer` or start of program/package. The resolution, the shortest time interval that can be measured, depends on the program and system.

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

1.2.4 Miscellaneous [1, “7.21 Miscellaneous”]

`\pdf@draftmode`

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

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

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

`\pdf@setdraftmode {<value>}`

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

`\pdf@shellescape`

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

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

Hints for usage:

- Before its use `\pdf@shellescape` should be tested, whether it is available. Example with package `ltxcmds` (loaded by package `pdftexcmds`):

```
\ltx@ifundefined{pdf@shellescape}{%  
  % \pdf@shellescape is undefined  
}%  
% \pdf@shellescape is available  
}
```

Use `\ltx@ifundefined` in expandable contexts.

- `\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.

1.2.5 Additional macro: `\pdf@isprimitive`

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

If `\cmd1` has the primitive meaning given by the primitive name of `\cmd2`, then the argument `<true>` is executed, otherwise `<false>`. The macro `\pdf@isprimitive` is expandable. Internally it checks the result of `\meaning` and is therefore available for all TeX variants, even the original TeX. Example with `\makeatletter`:

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

1.2.6 Experimental

```
\pdf@unescapehexnative {<string>}
\pdf@escapehexnative {<string>}
\pdf@escapenamename {<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 [2011/11/29 v0.20 Utility functions of pdfTeX for LuaTeX (HO)]%

```

2.2 Catcodes

```

69 \begingroup\catcode61\catcode48\catcode32=10\relax%
70 \catcode13=5 % ^~M
71 \endlinechar=13 %
72 \catcode123=1 % {
73 \catcode125=2 % }
74 \catcode64=11 % @
75 \def\x{\endgroup
76   \expandafter\edef\csname pdftexcmds@AtEnd\endcsname{%
77     \endlinechar=\the\endlinechar\relax
78     \catcode13=\the\catcode13\relax
79     \catcode32=\the\catcode32\relax
80     \catcode35=\the\catcode35\relax
81     \catcode61=\the\catcode61\relax
82     \catcode64=\the\catcode64\relax
83     \catcode123=\the\catcode123\relax
84     \catcode125=\the\catcode125\relax
85   }%
86 }%
87 \x\catcode61\catcode48\catcode32=10\relax%
88 \catcode13=5 % ^~M
89 \endlinechar=13 %
90 \catcode35=6 % #
91 \catcode64=11 % @
92 \catcode123=1 % {
93 \catcode125=2 % }
94 \def\TMP@EnsureCode#1#2{%
95   \edef\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\relax
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   \pdfdoccmds@nopdfdoc
180   \ltx@ifundefined{pdfdocversion}{%
181     }{%
182     \ifnum\pdfdocversion>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   \pdfdoccmds@nopdfdoc
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   \pdfdoccmds@nopdfdoc
206 \else
207   \def\pdf@mdfivesum#{\pdfmdfivesum}%
208   \let\pdf@mdfivesumnative\pdf@mdfivesum
209   \def\pdf@filemdfivesum#{\pdfmdfivesum file}%
210 \fi
211 \def\pdf@system#{%
212   \immediate\write18%
213 }%
214 \def\pdfdoccmds@temp#1{%
215   \begingroup\expandafter\expandafter\expandafter\endgroup
216   \expandafter\ifx\csname pdf#1\endcsname\relax
217     \pdfdoccmds@nopdfdoc
218   \else
219     \expandafter\let\csname pdf@#1\expandafter\endcsname
220     \csname pdf#1\endcsname
221   \fi
222 }%
223 \pdfdoccmds@temp{resettimer}%
224 \pdfdoccmds@temp{elapsedtime}%
225 \fi

```

2.5 `\pdf@primitive`, `\pdf@ifprimitive`

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

X_YTeX provides them under the name `\primitive` and `\ifprimitive`. Lua_TE_X knows both name variants, but they have possibly to be enabled first (`tex.enableprimitives`).

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

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

2.5.1 Using Lua_TE_X's `tex.enableprimitives`

```

226 \ifluatex

\pdftexcmds@directlua

227 \ifnum\luatexversion<36 %
228   \def\pdftexcmds@directlua{\directlua0 }%
229 \else
230   \let\pdftexcmds@directlua\directlua
231 \fi

232 \begingroup
233   \newlinechar=10 %
234   \endlinechar=\newlinechar
235   \pdftexcmds@directlua{%
236     if tex.enableprimitives then
237       tex.enableprimitives(
238         'pdf@',
239         {'primitive', 'ifprimitive', 'pdfdraftmode'}
240       )
241       tex.enableprimitives('', {'luaescapestring'})
242     end
243   }%
244 \endgroup %

245 \fi

```

2.5.2 Trying various names to find the primitives

```

\pdftexcmds@strip@prefix

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

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

\pdf@primitive

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

\pdf@ifprimitive

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

Disable broken \pdfprimitive.

270 \begingroup
271   \expandafter\ifx\csname pdf@primitive\endcsname\relax
272   \else
273     \expandafter\ifx\csname pdftexversion\endcsname\relax

```

```

274 \else
275 \ifnum\pdftexversion=140 %
276 \expandafter\ifx\csname pdftexrevision\endcsname\relax
277 \else
278 \ifnum\pdftexrevision<4 %
279 \endgroup
280 \let\pdf@primitive\@undefined
281 \@PackageInfoNoLine{pdftexcms}{%
282 \string\pdf@primitive\space disabled, %
283 because\MessageBreak
284 \string\pdfprimitive\space is broken until pdfTeX 1.40.4%
285 }%
286 \begingroup
287 \fi
288 \fi
289 \fi
290 \fi
291 \fi
292 \endgroup

```

2.5.3 Result

```

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

```

2.6 X_YTeX

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

```

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

```

2.7 \pdf@isprimitive

```

325 \def\pdf@isprimitive{%
326 \begingroup\expandafter\expandafter\expandafter\endgroup

```



```

387   }%
388   \fi
389 }
390 \ifcase\pdftexcmds@temp

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

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

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

\pdftexcmds@draftmode
395   \let\pdftexcmds@draftmode\pdf@draftmode

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

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

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

413 \fi

\pdf@setdraftmode
414 \def\pdf@setdraftmode#1{%
415   \begingroup
416     \count\ltx@cc1v=#1\relax
417   \edef\x{\endgroup
418     \noexpand\pdftexcmds@@setdraftmode{\the\count\ltx@cc1v}}%
419   }%
420   \x
421 }

\pdftexcmds@@setdraftmode
422 \def\pdftexcmds@@setdraftmode#1{%
423   \ifcase#1 %
424     \pdftexcmds@setdraftmode{#1}%
425   \or
426     \pdftexcmds@setdraftmode{#1}%
427   \else
428     \@PackageWarning{pdftexcmds}{%

```

```

429     \string\pdf@setdraftmode: Ignoring\MessageBreak
430     invalid value `#1'%
431 }%
432 \fi
433 }

```

2.9 Load Lua module

```

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

438 \begingroup\expandafter\expandafter\expandafter\endgroup
439 \expandafter\ifx\csname RequirePackage\endcsname\relax
440   \def\TMP@RequirePackage#1[#2]{%
441     \begingroup\expandafter\expandafter\expandafter\endgroup
442     \expandafter\ifx\csname ver@#1.sty\endcsname\relax
443       \input #1.sty\relax
444     \fi
445   }%
446   \TMP@RequirePackage{luatex-loader}[2009/04/10]%
447 \else
448   \RequirePackage{luatex-loader}[2009/04/10]%
449 \fi
450 \pdfdocmds@directlua{%
451   require("oberdiek.pdfdocmds")%
452 }
453 \ifnum\luatexversion>37 %
454   \ifnum0%
455     \pdfdocmds@directlua{%
456       if status.ini_version then %
457         tex.write("1")%
458       end%
459     }>0 %
460   \everyjob\expandafter{%
461     \the\everyjob
462     \pdfdocmds@directlua{%
463       require("oberdiek.pdfdocmds")%
464     }%
465   }%
466 \fi
467 \fi
468 \begingroup
469   \def\x{2011/11/29 v0.20}%
470   \ltx@onelevel@sanitize\x
471   \edef\y{%
472     \pdfdocmds@directlua{%
473       if oberdiek.pdfdocmds.getversion then %
474         oberdiek.pdfdocmds.getversion()%
475       end%
476     }%
477   }%
478   \ifx\x\y
479   \else
480     \@PackageError{pdfdocmds}{%
481       Wrong version of lua module.\MessageBreak
482       Package version: \x\MessageBreak
483       Lua module: \y
484     }{\@ehc
485   \fi
486 \endgroup

```

2.10 Lua functions

2.10.1 Helper macros

```
\pdfTEXcmds@toks
487 \begingroup\expandafter\expandafter\expandafter\endgroup
488 \expandafter\ifx\csname newtoks\endcsname\relax
489 \toksdef\pdfTEXcmds@toks=0 %
490 \else
491 \csname newtoks\endcsname\pdfTEXcmds@toks
492 \fi

\pdfTEXcmds@Patch
493 \def\pdfTEXcmds@Patch{0}
494 \ifnum\luatEXversion>40 %
495 \ifnum\luatEXversion<66 %
496 \def\pdfTEXcmds@Patch{1}%
497 \fi
498 \fi

499 \ifcase\pdfTEXcmds@Patch
500 \catcode`\&=14 %
501 \else
502 \catcode`\&=9 %

\pdfTEXcmds@PatchDecode
503 \def\pdfTEXcmds@PatchDecode#1\@nil{%
504 \pdfTEXcmds@DecodeA#1^^A^^A\@nil{}%
505 }%

\pdfTEXcmds@DecodeA
506 \def\pdfTEXcmds@DecodeA#1^^A^^A#2\@nil#3{%
507 \ifx\relax#2\relax
508 \ltx@ReturnAfterElseFi{%
509 \pdfTEXcmds@DecodeB#3#1^^A^^B\@nil{}%
510 }%
511 \else
512 \ltx@ReturnAfterFi{%
513 \pdfTEXcmds@DecodeA#2\@nil{#3#1^^@}%
514 }%
515 \fi
516 }%

\pdfTEXcmds@DecodeB
517 \def\pdfTEXcmds@DecodeB#1^^A^^B#2\@nil#3{%
518 \ifx\relax#2\relax%
519 \ltx@ReturnAfterElseFi{%
520 \ltx@zero
521 #3#1%
522 }%
523 \else
524 \ltx@ReturnAfterFi{%
525 \pdfTEXcmds@DecodeB#2\@nil{#3#1^^A}%
526 }%
527 \fi
528 }%

529 \fi

530 \ifnum\luatEXversion<36 %
531 \else
532 \catcode`\0=9 %
533 \fi
```


2.10.2 Strings [1, “7.15 Strings”]

`\pdf@strcmp`

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

540 `\pdf@isprimitive`

`\pdf@escapehex`

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

`\pdf@escapehexnative`

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

`\pdf@unescapehex`

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

`\pdf@unescapehexnative`

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

`\pdf@escapestring`

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

`\pdf@escapename`

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

`\pdf@escapenamenative`

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

2.10.3 Files [1, “7.18 Files”]

`\pdf@filesize`

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

`\pdf@filemoddate`

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

`\pdf@filedump`

```
594 \def\pdf@filedump#1#2#3{%
595   \directlua0{%
596     oberdiek.pdfdoccmds.filedump("\luaescapestring{\number#1}",%
597       "\luaescapestring{\number#2}",%
598       "\luaescapestring{#3}")%
599   }%
600 }
```

`\pdf@mdfivesum`

```
601 \long\def\pdf@mdfivesum#1{%
602   \directlua0{%
603     oberdiek.pdfdoccmds.mdfivesum("\luaescapestring{#1}", "byte")%
604   }%
605 }
```

`\pdf@mdfivesumnative`

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

`\pdf@filemdfivesum`

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

2.10.4 Timekeeping [1, “7.17 Timekeeping”]

`\protected`

```
616 \let\pdfdoccmds@temp=Y%
617 \begingroup\expandafter\expandafter\expandafter\endgroup
618 \expandafter\ifx\csname protected\endcsname\relax
619   \pdfdoccmds@directlua0{%
620     if tex.enableprimitives then %
```

```

621     tex.enableprimitives(' ', {'protected'})%
622   end%
623 }%
624 \fi
625 \begingroup\expandafter\expandafter\expandafter\endgroup
626 \expandafter\ifx\csname protected\endcsname\relax
627   \let\pdftexcmds@temp=N%
628 \fi

```

`\numexpr`

```

629 \begingroup\expandafter\expandafter\expandafter\endgroup
630 \expandafter\ifx\csname numexpr\endcsname\relax
631   \pdftexcmds@directlua0{%
632     if tex.enableprimitives then %
633       tex.enableprimitives(' ', {'numexpr'})%
634     end%
635   }%
636 \fi
637 \begingroup\expandafter\expandafter\expandafter\endgroup
638 \expandafter\ifx\csname numexpr\endcsname\relax
639   \let\pdftexcmds@temp=N%
640 \fi

641 \ifx\pdftexcmds@temp N%
642   \@PackageWarningNoLine{pdftexcmds}{%
643     Definitions of \ltx@backslashchar pdf@resettimer and%
644     \MessageBreak
645     \ltx@backslashchar pdf@elapsedtime are skipped, because%
646     \MessageBreak
647     e-TeX's \ltx@backslashchar protected or %
648     \ltx@backslashchar numexpr are missing%
649   }%
650 \else

```

`\pdf@resettimer`

```

651   \protected\def\pdf@resettimer{%
652     \pdftexcmds@directlua0{%
653       oberdiek.pdftexcmds.resettimer()%
654     }%
655   }%

```

`\pdf@elapsedtime`

```

656   \protected\def\pdf@elapsedtime{%
657     \numexpr
658     \pdftexcmds@directlua0{%
659       oberdiek.pdftexcmds.elapsedtime()%
660     }%
661   \relax
662   }%

663 \fi

```

2.10.5 Shell escape

`\pdf@shellescape`

Caution: Catcode of digit zero might be ‘ignore’.

```

664 \ifnum\luatexversion<68 %
665 \else
666   \def\pdf@shellescape{%
667     \directlua0{%
668       oberdiek.pdftexcmds.shellescape()%
669     }%
670   }%
671 \fi

```

`\pdf@system`

```
672 \def\pdf@system#1{%
673   \directlua0{%
674     oberdiek.pdfTeXcmds.system("\luaescapestring{#1}")%
675   }%
676 }
```

`\pdf@lastsystemstatus`

```
677 \def\pdf@lastsystemstatus{%
678   \directlua0{%
679     oberdiek.pdfTeXcmds.lastsystemstatus()%
680   }%
681 }
```

`\pdf@lastsystemexit`

```
682 \def\pdf@lastsystemexit{%
683   \directlua0{%
684     oberdiek.pdfTeXcmds.lastsystemexit()%
685   }%
686 }
```

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

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

```
688 \ifnum0%
689   \pdfTeXcmds@directlua{%
690     if io.popen then %
691       tex.write("1")%
692     end%
693   }%
694   =1 %
695   \def\pdf@pipe#1{%
696 &   \romannumeral\expandafter\pdfTeXcmds@PatchDecode
697   \the\expandafter\pdfTeXcmds@toks
698   \pdfTeXcmds@directlua{%
699     oberdiek.pdfTeXcmds.toks="pdfTeXcmds@toks"%
700     oberdiek.pdfTeXcmds.pipe("\luaescapestring{#1}", \pdfTeXcmds@Patch)%
701   }%
702 &   \@nil
703   }%
704 \fi

705 \pdfTeXcmds@AtEnd%
706 </package>
```

2.11 Lua module

```
707 < *lua >
708 module("oberdiek.pdfTeXcmds", package.seeall)
709 local systemexitstatus
710 function getversion()
711   tex.write("2011/11/29 v0.20")
712 end
```

2.11.1 Strings [1, “7.15 Strings”]

```
713 function strcmp(A, B)
714   if A == B then
715     tex.write("0")
716   elseif A < B then
717     tex.write("-1")
718   else
719     tex.write("1")
```

```

720 end
721 end
722 local function utf8_to_byte(str)
723 local i = 0
724 local n = string.len(str)
725 local t = {}
726 while i < n do
727 i = i + 1
728 local a = string.byte(str, i)
729 if a < 128 then
730 table.insert(t, string.char(a))
731 else
732 if a >= 192 and i < n then
733 i = i + 1
734 local b = string.byte(str, i)
735 if b < 128 or b >= 192 then
736 i = i - 1
737 elseif a == 194 then
738 table.insert(t, string.char(b))
739 elseif a == 195 then
740 table.insert(t, string.char(b + 64))
741 end
742 end
743 end
744 end
745 return table.concat(t)
746 end
747 function escapehex(str, mode)
748 if mode == "byte" then
749 str = utf8_to_byte(str)
750 end
751 tex.write((string.gsub(str, ".",
752 function (ch)
753 return string.format("%02X", string.byte(ch))
754 end
755 )))
756 end

```

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

```

757 function unescapehex(str, mode, patch)
758 local a = 0
759 local first = true
760 local result = {}
761 for i = 1, string.len(str), 1 do
762 local ch = string.byte(str, i)
763 if ch >= 48 and ch <= 57 then
764 ch = ch - 48
765 elseif ch >= 65 and ch <= 70 then
766 ch = ch - 55
767 elseif ch >= 97 and ch <= 102 then
768 ch = ch - 87
769 else
770 ch = nil
771 end
772 if ch then
773 if first then
774 a = ch * 16
775 first = false
776 else
777 table.insert(result, a + ch)
778 first = true
779 end

```

```

780     end
781 end
782 if not first then
783     table.insert(result, a)
784 end
785 if patch == 1 then
786     local temp = {}
787     for i, a in ipairs(result) do
788         if a == 0 then
789             table.insert(temp, 1)
790             table.insert(temp, 1)
791         else
792             if a == 1 then
793                 table.insert(temp, 1)
794                 table.insert(temp, 2)
795             else
796                 table.insert(temp, a)
797             end
798         end
799     end
800     result = temp
801 end
802 if mode == "byte" then
803     local utf8 = {}
804     for i, a in ipairs(result) do
805         if a < 128 then
806             table.insert(utf8, a)
807         else
808             if a < 192 then
809                 table.insert(utf8, 194)
810                 a = a - 128
811             else
812                 table.insert(utf8, 195)
813                 a = a - 192
814             end
815             table.insert(utf8, a + 128)
816         end
817     end
818     result = utf8
819 end
820 tex.settoks(toks, string.char(unpack(result)))
821 end

```

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

```

822 function escapestring(str, mode)
823     if mode == "byte" then
824         str = utf8_to_byte(str)
825     end
826     tex.write((string.gsub(str, ".",
827         function (ch)
828             local b = string.byte(ch)
829             if b < 33 or b > 126 then
830                 return string.format("\\%.3o", b)
831             end
832             if b == 40 or b == 41 or b == 92 then
833                 return "\\\" .. ch
834             end

```

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

```

835         return nil
836     end
837 )))
838 end

```

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

```

839 function escapename(str, mode)
840   if mode == "byte" then
841     str = utf8_to_byte(str)
842   end
843   tex.write((string.gsub(str, ".",
844     function (ch)
845       local b = string.byte(ch)
846       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.

```

847     return ""
848   end
849   if b <= 32 or b >= 127
850     or b == 35 or b == 37 or b == 40 or b == 41
851     or b == 47 or b == 60 or b == 62 or b == 91
852     or b == 93 or b == 123 or b == 125 then
853     return string.format("#%.2X", b)
854   else

```

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

```

855     return nil
856   end
857 end
858 )))
859 end

```

2.11.2 Files [1, “7.18 Files”]

```

860 function filesize(filename)
861   local foundfile = kpse.find_file(filename, "tex", true)
862   if foundfile then
863     local size = lfs.attributes(foundfile, "size")
864     if size then
865       tex.write(size)
866     end
867   end
868 end

```

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

```

869 function filemoddate(filename)
870   local foundfile = kpse.find_file(filename, "tex", true)
871   if foundfile then
872     local date = lfs.attributes(foundfile, "modification")
873     if date then
874       local d = os.date("!*t", date)
875       if d.sec >= 60 then
876         d.sec = 59
877       end
878       local u = os.date("!*t", date)
879       local off = 60 * (d.hour - u.hour) + d.min - u.min
880       if d.year ~= u.year then
881         if d.year > u.year then
882           off = off + 1440
883         else
884           off = off - 1440
885         end
886       elseif d.yday ~= u.yday then
887         if d.yday > u.yday then
888           off = off + 1440
889         else
890           off = off - 1440
891         end
892       end
893       local timezone

```

```

894     if off == 0 then
895         timezone = "Z"
896     else
897         local hours = math.floor(off / 60)
898         local mins = math.abs(off - hours * 60)
899         timezone = string.format("%+03d'%02d'", hours, mins)
900     end
901     tex.write(string.format("D:%04d%02d%02d%02d%02d%02d%s",
902         d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
903     end
904 end
905 end
906 function filedump(offset, length, filename)
907     length = tonumber(length)
908     if length and length > 0 then
909         local foundfile = kpse.find_file(filename, "tex", true)
910         if foundfile then
911             offset = tonumber(offset)
912             if not offset then
913                 offset = 0
914             end
915             local filehandle = io.open(foundfile, "r")
916             if filehandle then
917                 if offset > 0 then
918                     filehandle:seek("set", offset)
919                 end
920                 local dump = filehandle:read(length)
921                 escapehex(dump)
922             end
923         end
924     end
925 end
926 function md5sum(str, mode)
927     if mode == "byte" then
928         str = utf8_to_byte(str)
929     end
930     escapehex(md5.sum(str))
931 end
932 function filemd5sum(filename)
933     local foundfile = kpse.find_file(filename, "tex", true)
934     if foundfile then
935         local filehandle = io.open(foundfile, "r")
936         if filehandle then
937             local contents = filehandle:read("*a")
938             escapehex(md5.sum(contents))
939         end
940     end
941 end

```

2.11.3 Timekeeping [1, “7.17 Timekeeping”]

The functions for timekeeping are based on Andy Thomas’ work [3]. Changes:

- Overflow check is added.
- `string.format` is used to avoid exponential number representation for sure.
- `tex.write` is used instead of `tex.print` to get tokens with catcode 12 and without appended `\endlinechar`.

```

942 local basetime = 0
943 function resettimer()
944     basetime = os.clock()
945 end
946 function elapsedtime()

```



```

947 local val = (os.clock() - basetime) * 65536 + .5
948 if val > 2147483647 then
949     val = 2147483647
950 end
951 tex.write(string.format("%d", val))
952 end

```

2.11.4 Miscellaneous [1, “7.21 Miscellaneous”]

```

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

```

```

1007     handle:close()
1008   end
1009 else
1010   texio.write("log", "disabled.")
1011 end
1012 if result then
1013   if patch == 1 then
1014     local temp = {}
1015     for i, a in ipairs(result) do
1016       if a == 0 then
1017         table.insert(temp, 1)
1018         table.insert(temp, 1)
1019       else
1020         if a == 1 then
1021           table.insert(temp, 1)
1022           table.insert(temp, 2)
1023         else
1024           table.insert(temp, a)
1025         end
1026       end
1027     end
1028     result = temp
1029   end
1030   tex.settoks(toks, result)
1031 else
1032   tex.settoks(toks, "")
1033 end
1034 end
1035 </lua>

```

3 Test

3.1 Catcode checks for loading

```

1036 <*test1>
1037 \catcode\{=1 %
1038 \catcode\}=2 %
1039 \catcode\#=6 %
1040 \catcode\@=11 %
1041 \expandafter\ifx\csname count@\endcsname\relax
1042   \countdef\count@=255 %
1043 \fi
1044 \expandafter\ifx\csname @gobble\endcsname\relax
1045   \long\def@gobble#1{}%
1046 \fi
1047 \expandafter\ifx\csname @firstofone\endcsname\relax
1048   \long\def@firstofone#1{#1}%
1049 \fi
1050 \expandafter\ifx\csname loop\endcsname\relax
1051   \expandafter\@firstofone
1052 else
1053   \expandafter@gobble
1054 \fi
1055 {%
1056   \def\loop#1\repeat{%
1057     \def\body{#1}%
1058     \iterate
1059   }%
1060   \def\iterate{%
1061     \body
1062     \let\next\iterate
1063   \else

```

```

1064     \let\next\relax
1065     \fi
1066     \next
1067 }%
1068 \let\repeat=\fi
1069 }%
1070 \def\RestoreCatcodes{}
1071 \count@=0 %
1072 \loop
1073   \edef\RestoreCatcodes{%
1074     \RestoreCatcodes
1075     \catcode\the\count@=\the\catcode\count@\relax
1076   }%
1077 \ifnum\count@<255 %
1078   \advance\count@ 1 %
1079 \repeat
1080
1081 \def\RangeCatcodeInvalid#1#2{%
1082   \count@=#1\relax
1083   \loop
1084     \catcode\count@=15 %
1085   \ifnum\count@<#2\relax
1086     \advance\count@ 1 %
1087   \repeat
1088 }
1089 \def\RangeCatcodeCheck#1#2#3{%
1090   \count@=#1\relax
1091   \loop
1092     \ifnum#3=\catcode\count@
1093     \else
1094       \errmessage{%
1095         Character \the\count@\space
1096         with wrong catcode \the\catcode\count@\space
1097         instead of \number#3%
1098       }%
1099     \fi
1100   \ifnum\count@<#2\relax
1101     \advance\count@ 1 %
1102   \repeat
1103 }
1104 \def\space{ }
1105 \expandafter\ifx\csname LoadCommand\endcsname\relax
1106 \def\LoadCommand{\input pdftexcmds.sty\relax}%
1107 \fi
1108 \def\Test{%
1109   \RangeCatcodeInvalid{0}{47}%
1110   \RangeCatcodeInvalid{58}{64}%
1111   \RangeCatcodeInvalid{91}{96}%
1112   \RangeCatcodeInvalid{123}{255}%
1113   \catcode`\@=12 %
1114   \catcode`\=0 %
1115   \catcode`\%=14 %
1116   \LoadCommand
1117   \RangeCatcodeCheck{0}{36}{15}%
1118   \RangeCatcodeCheck{37}{37}{14}%
1119   \RangeCatcodeCheck{38}{47}{15}%
1120   \RangeCatcodeCheck{48}{57}{12}%
1121   \RangeCatcodeCheck{58}{63}{15}%
1122   \RangeCatcodeCheck{64}{64}{12}%
1123   \RangeCatcodeCheck{65}{90}{11}%
1124   \RangeCatcodeCheck{91}{91}{15}%
1125   \RangeCatcodeCheck{92}{92}{0}%

```

```

1126 \RangeCatcodeCheck{93}{96}{15}%
1127 \RangeCatcodeCheck{97}{122}{11}%
1128 \RangeCatcodeCheck{123}{255}{15}%
1129 \RestoreCatcodes
1130 }
1131 \Test
1132 \csname @@end\endcsname
1133 \end
1134 </test1>

```

3.2 Test for \pdf@isprimitive

```

1135 <*test2>
1136 \catcode`\{=1 %
1137 \catcode`\}=2 %
1138 \catcode`\#=6 %
1139 \catcode`\@=11 %
1140 \input pdftexcmds.sty\relax
1141 \def\msg#1{%
1142   \begingroup
1143     \escapechar=92 %
1144     \immediate\write16{#1}%
1145   \endgroup
1146 }
1147 \long\def\test#1#2#3#4{%
1148   \begingroup
1149     #4%
1150     \def\str{%
1151       Test \string\pdf@isprimitive
1152       {\string #1}{\string #2}{...}: %
1153     }%
1154     \pdf@isprimitive{#1}{#2}{%
1155       \ifx#3Y%
1156         \msg{\str true ==> OK.}%
1157       \else
1158         \errmessage{\str false ==> FAILED}%
1159       \fi
1160     }{%
1161       \ifx#3Y%
1162         \errmessage{\str true ==> FAILED}%
1163       \else
1164         \msg{\str false ==> OK.}%
1165       \fi
1166     }%
1167   \endgroup
1168 }
1169 \test\relax\relax Y{}
1170 \test\foobar\relax Y{\let\foobar\relax}
1171 \test\foobar\relax N{}
1172 \test\hbox\hbox Y{}
1173 \test\foobar@hbox\hbox Y{\let\foobar@hbox\hbox}
1174 \test\if\if Y{}
1175 \test\if\ifx N{}
1176 \test\ifx\if N{}
1177 \test\par\par Y{}
1178 \test\hbox\par N{}
1179 \test\par\hbox N{}
1180 \csname @@end\endcsname\end
1181 </test2>

```

3.3 Test for \pdf@shellescape

```

1182 <*test-shell>
1183 \catcode`\{=1 %

```

```

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

```

3.4 Test for escape functions

```

1229 <*test-escape>
1230 \catcode`\{=1 %
1231 \catcode`\}=2 %
1232 \catcode`\#=6 %
1233 \catcode`\^=7 %
1234 \catcode`\@=11 %
1235 \errorcontextlines=1000 %
1236 \input pdftexcmds.sty\relax
1237 \def\msg#1{%
1238 \begingroup
1239 \escapechar=92 %
1240 \immediate\write16{#1}%
1241 \endgroup
1242 }
1243 \begingroup

```

```

1244 \catcode`\@=11 %
1245 \countdef\count@=255 %
1246 \def\space{ }%
1247 \long\def\@whilenum#1\do #2{%
1248   \ifnum #1\relax
1249     #2\relax
1250   \@iwhilenum{#1\relax#2\relax}%
1251   \fi
1252 }%
1253 \long\def\@iwhilenum#1{%
1254   \ifnum #1%
1255     \expandafter\@iwhilenum
1256   \else
1257     \expandafter\ltx@gobble
1258   \fi
1259   {#1}%
1260 }%
1261 \gdef\AllBytes{}%
1262 \count@=0 %
1263 \catcode0=12 %
1264 \@whilenum\count@<256 \do{%
1265   \lccode0=\count@
1266   \ifnum\count@=32 %
1267     \xdef\AllBytes{\AllBytes\space}%
1268   \else
1269     \lowercase{%
1270       \xdef\AllBytes{\AllBytes^^@}%
1271     }%
1272   \fi
1273   \advance\count@ by 1 %
1274 }%
1275 \endgroup

1276 \def\AllBytesHex{%
1277 000102030405060708090A0B0C0D0E0F%
1278 101112131415161718191A1B1C1D1E1F%
1279 202122232425262728292A2B2C2D2E2F%
1280 303132333435363738393A3B3C3D3E3F%
1281 404142434445464748494A4B4C4D4E4F%
1282 505152535455565758595A5B5C5D5E5F%
1283 606162636465666768696A6B6C6D6E6F%
1284 707172737475767778797A7B7C7D7E7F%
1285 808182838485868788898A8B8C8D8E8F%
1286 909192939495969798999A9B9C9D9E9F%
1287 A0A1A2A3A4A5A6A7A8A9AAABACADAFAF%
1288 B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF%
1289 C0C1C2C3C4C5C6C7C8C9CACBCCDCECF%
1290 D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
1291 E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
1292 F0F1F2F3F4F5F6F7F8F9FABFBCFDFEFF%
1293 }
1294 \ltx@onelevel@sanitize\AllBytesHex
1295 \expandafter\lowercase\expandafter{%
1296 \expandafter\def\expandafter\AllBytesHexLC
1297   \expandafter{\AllBytesHex}%
1298 }
1299 \begingroup
1300 \catcode`\#=12 %
1301 \xdef\AllBytesName{%
1302 #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
1303 #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
1304 #20!"#23$#25&'#28#29*+,-.#2F%
1305 0123456789:;#3C=#3E?%

```

```

1306 @ABCDEFGHJKLMNO%
1307 PQRSTUVWXYZ#5B\ltx@backslashchar#5D^_%
1308 `abcdefghijklmno%
1309 pqrstuvwxyz#7B|#7D\string~#7F%
1310 #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
1311 #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
1312 #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
1313 #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
1314 #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
1315 #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
1316 #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
1317 #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
1318 }%
1319 \endgroup
1320 \ltx@onelevel@sanitize\AllBytesName
1321 \edef\AllBytesFromName{\expandafter\ltx@gobble\AllBytes}
1322 \begingroup
1323 \def\|{|}%
1324 \edef%\{\ltx@percentchar}%
1325 \catcode`\|=0 %
1326 \catcode`\#=12 %
1327 \catcode`\~ =12 %
1328 \catcode`\|=12 %
1329 |xdef|AllBytesString{%
1330 \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
1331 \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
1332 \040!"#$%&'(\)*+,-./%
1333 0123456789 ;<=>?%
1334 @ABCDEFGHJKLMNO%
1335 PQRSTUVWXYZ[\]^_ %
1336 `abcdefghijklmno%
1337 pqrstuvwxyz{|}~\177%
1338 \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
1339 \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
1340 \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
1341 \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
1342 \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
1343 \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
1344 \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
1345 \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
1346 }%
1347 |endgroup
1348 \ltx@onelevel@sanitize\AllBytesString
1349 \def\Test#1#2#3{%
1350 \begingroup
1351 \expandafter\expandafter\expandafter\def
1352 \expandafter\expandafter\expandafter\TestResult
1353 \expandafter\expandafter\expandafter{%
1354 #1#2}%
1355 }%
1356 \ifx\TestResult#3%
1357 \else
1358 \newlinechar=10 %
1359 \msg{Expect:^^J#3}%
1360 \msg{Result:^^J\TestResult}%
1361 \errmessage{\string#2 -\string#1-> \string#3}%
1362 \fi
1363 \endgroup
1364 }
1365 \def\test#1#2#3{%
1366 \edef\TestFrom{#2}%
1367 \edef\TestExpect{#3}%

```

```

1368 \ltx@onelevel@sanitize\TestExpect
1369 \Test#1\TestFrom\TestExpect
1370 }
1371 \test\pdf@unescapehex{74657374}{test}
1372 \begingroup
1373 \catcode0=12 %
1374 \catcode1=12 %
1375 \test\pdf@unescapehex{740074017400740174}{t^^@t^^At^^@t^^At}%
1376 \endgroup
1377 \Test\pdf@escapehex\AllBytes\AllBytesHex
1378 \Test\pdf@unescapehex\AllBytesHex\AllBytes
1379 \Test\pdf@escapename\AllBytes\AllBytesName
1380 \Test\pdf@escapestring\AllBytes\AllBytesString

1381 \csname @@end@endcsname@end
1382 </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[ftp://ftp.ctan.org/tex-archive/](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 `mktexlsr`.

4.5 Some details for the interested

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

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

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

5 Catalogue

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

```
1383 (*catalogue)
1384 <?xml version='1.0' encoding='us-ascii'?>
1385 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
1386 <entry datestamp='$Date$' modifier='$Author$' id='pdftexcmds'>
1387   <name>pdftexcmds</name>
1388   <caption>LuaTeX support for pdfTeX utility functions.</caption>
1389   <authorref id='auth:oberdiek'>
1390     <copyright owner='Heiko Oberdiek' year='2007,2009-2011'>/>
1391     <license type='lppl1.3'>/>
1392     <version number='0.20'>/>
1393     <description>
1394       LuaTeX provides most of the commands of
1395       <xref refid='pdftex'>pdfTeX</xref> 1.40. However, a number of
1396       utility functions are not available. This package tries to fill
1397       the gap and implements some of the missing primitives using Lua.
1398     <p/>
1399     The package is part of the <xref refid='oberdiek'>oberdiek</xref>
1400     bundle.
1401   </description>
1402   <documentation details='Package documentation'
1403     href='ctan:/macros/latex/contrib/oberdiek/pdftexcmds.pdf'>/>
1404   <ctan file='true' path='/macros/latex/contrib/oberdiek/pdftexcmds.dtx'>/>
1405   <miktex location='oberdiek'>/>
1406   <texlive location='oberdiek'>/>
1407   <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip'>/>
1408 </entry>
1409 </catalogue>
```

6 References

- [1] Hàn Thê Thành et al. *The pdfT_EX user manual*. Version 655 (1.40.11). 2010-11-23. URL: <http://mirror.ctan.org/systems/pdftex/manual/pdftex-a.pdf> (visited on 2011-11-29).
- [2] LuaT_EX development team. *LuaT_EX Reference*. Version beta 0.71.0. 2011-10-11. URL: <http://www.luatex.org/svn/trunk/manual/luatexref-t.pdf> (visited on 2011-11-29).
- [3] Andy Thomas. *Analog of \pdfelapsedtime for LuaT_EX and X_YT_EX*. URL: <http://tex.stackexchange.com/a/32531> (visited on 2011-11-29).

7 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.
- X_YTeX's variants are detected for `\pdf@shellescape`, `\pdf@strcmp`, `\pdf@primitive`, `\pdf@ifprimitive`.

[2009/09/23 v0.6]

- Macro `\pdf@isprimitive` added.

[2009/12/12 v0.7]

- Short info shortened.

[2010/03/01 v0.8]

- Required date for package `ifluatex` updated.

[2010/04/01 v0.9]

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

[2010/11/04 v0.10]

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

[2010/11/11 v0.11]

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

[2011/01/30 v0.12]

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

[2011/03/04 v0.13]

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

[2011/04/10 v0.14]

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

[2011/04/16 v0.15]

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

1092, 1095, 1096, 1100, 1101, 1245, 1262, 1264, 1265, 1266, 1273	\ifx 15, 18, 21, 50, 58, 61, 133, 136, 159, 178, 196, 204, 216, 249, 256, 271, 273, 276, 296, 301, 307, 315, 327, 342, 343, 346, 348, 439, 442, 478, 488, 507, 518, 618, 626, 630, 638, 641, 1041, 1044, 1047, 1050, 1105, 1155, 1161, 1175, 1176, 1190, 1193, 1204, 1209, 1211, 1213, 1356
\countdef 1042, 1245	\immediate 23, 52, 212, 1144, 1188, 1240
\csname 14, 21, 50, 66, 76, 133, 136, 159, 162, 164, 178, 196, 204, 216, 219, 220, 249, 254, 257, 258, 271, 273, 276, 296, 301, 307, 310, 316, 318, 327, 362, 439, 442, 488, 491, 618, 626, 630, 638, 1041, 1044, 1047, 1050, 1105, 1132, 1180, 1200, 1227, 1381	\input . 137, 443, 1106, 1140, 1187, 1236
	\iterate 1058, 1060, 1062
D	L
\delimiter 335, 341, 359	\lccode 1265
\directlua 228, 230, 535, 542, 547, 554, 563, 570, 575, 580, 585, 590, 595, 602, 607, 612, 667, 673, 678, 683	\LoadCommand 1106, 1116
\do 1247, 1264	\loop 1056, 1072, 1083, 1091
E	\lowercase 1269, 1295
\empty 17, 18	\ltx@backslashchar 379, 383, 386, 643, 645, 647, 648, 1307
\end 1133, 1180, 1200, 1227, 1381	\ltx@cclv 416, 418
\endcsname 14, 21, 50, 66, 76, 133, 136, 159, 162, 164, 178, 196, 204, 216, 219, 220, 249, 254, 257, 258, 271, 273, 276, 296, 301, 307, 310, 316, 318, 327, 362, 439, 442, 488, 491, 618, 626, 630, 638, 1041, 1044, 1047, 1050, 1105, 1132, 1180, 1200, 1227, 1381	\ltx@firstoftwo 336, 366, 398
\endinput 29, 129	\ltx@gobble 1257, 1321
\endlinechar ... 4, 35, 71, 77, 89, 234	\ltx@ifUndefined 180, 378
\errmessage 1094, 1158, 1162, 1198, 1216, 1223, 1361	\ltx@one 382, 397, 404, 405
\errorcontextlines 1235	\ltx@onelevel@sanitize 470, 1294, 1320, 1348, 1368
\escapechar 128, 131, 253, 309, 1143, 1239	\ltx@percentchar 1324
\everyjob 460, 461	\ltx@ReturnAfterElseFi 508, 519
\expected 1209, 1211, 1219, 1220	\ltx@ReturnAfterFi 512, 524
F	\ltx@secondoftwo .. 338, 368, 392, 400
\foobar 1170, 1171	\ltx@zero 377, 391, 407, 520
\foobar@hbox 1173	\luaescapestring 536, 537, 543, 548, 556, 565, 571, 576, 581, 586, 591, 596, 597, 598, 603, 608, 613, 674, 700
G	\luatexversion 227, 453, 494, 495, 530, 664, 1190, 1192
\gdef 1261	M
H	\MaybeEnd 1189, 1200, 1203
\hbox 1172, 1173, 1178, 1179	\meaning .. 252, 254, 310, 313, 329, 365
I	\MessageBreak 283, 429, 481, 482, 644, 646
\if 1174, 1175, 1176	\msg . 1141, 1156, 1164, 1188, 1194, 1195, 1197, 1205, 1207, 1212, 1214, 1219, 1221, 1237, 1359, 1360
\ifcase 390, 423, 499	N
\ifeof 182, 183	\newlinechar 233, 234, 1358
\ifluatex 150, 226, 373, 434	\next 1062, 1064, 1066
\ifnum . 182, 227, 275, 278, 335, 365, 397, 404, 453, 454, 494, 495, 530, 664, 688, 1077, 1085, 1092, 1100, 1192, 1220, 1248, 1254, 1266	\number . 128, 596, 597, 1097, 1207, 1219
\ifpdf 381	\numexpr 629, 657
	P
	\PackageInfo 26
	\par 1177, 1178, 1179
	\pdf@draftmode 5, 391, 403
	\pdf@elapseddtime 4, 656
	\pdf@escapehex 4, 170, 541, 1377
	\pdf@escapehexnative 170, 546
	\pdf@escapename 574, 1379
	\pdf@escapenamenative 579

<code>\pdf@escapestring</code>	569, 1380	<code>\pdftexcmds@toks</code>	487, 553, 562, 697
<code>\pdf@filedump</code>	4, 199, 594	<code>\pdftexrevision</code>	278
<code>\pdf@filemdfivesum</code>	4, 209, 611	<code>\pdftexversion</code>	182, 275
<code>\pdf@filemoddate</code>	4, 589	<code>\protected</code>	616, 651, 656
<code>\pdf@filesize</code>	4, 584	<code>\ProvidesPackage</code>	19, 67
<code>\pdf@ifdraftmode</code>	5, 392, 396		
<code>\pdf@ifprimitive</code>	6, 266, 300		
<code>\pdf@isprimitive</code>	6, 325, 328, 364, 375, 540, 1151, 1154		
<code>\pdf@lastsystemexit</code>	682		
<code>\pdf@lastsystemstatus</code>	677		
<code>\pdf@mdfivesum</code>	4, 207, 208, 601		
<code>\pdf@mdfivesumnative</code>	208, 606		
<code>\pdf@pipe</code>	6, 688		
<code>\pdf@primitive</code>	6, 262, 280, 282, 295		
<code>\pdf@resettimer</code>	4, 651		
<code>\pdf@setdraftmode</code>	5, 414, 429		
<code>\pdf@shellescape</code>	5, 184, 186, 191, 664, 1193, 1204, 1207, 1213, 1220		
<code>\pdf@strcmp</code>	3, 365, 534		
<code>\pdf@system</code>	5, 211, 672		
<code>\pdf@unescapehex</code>	3, 172, 551, 1371, 1375, 1378		
<code>\pdf@unescapehexnative</code>	6, 172, 560		
<code>\pdfdraftmode</code>	395		
<code>\pdffiledump</code>	200		
<code>\pdfmdfivesum</code>	207, 209		
<code>\pdfprimitive</code>	284		
<code>\pdfshellescape</code>	192		
<code>\pdftexcmds@isprimitive</code>	332, 334		
<code>\pdftexcmds@setdraftmode</code>	418, 422		
<code>\pdftexcmds@AtEnd</code>	95, 96, 126, 127, 436, 705		
<code>\pdftexcmds@DecodeA</code>	504, 506		
<code>\pdftexcmds@DecodeB</code>	509, 517		
<code>\pdftexcmds@directlua</code>	227, 235, 450, 455, 462, 472, 619, 631, 652, 658, 689, 698		
<code>\pdftexcmds@draftmode</code>	395, 397, 404, 411		
<code>\pdftexcmds@equal</code>	335, 341, 359		
<code>\pdftexcmds@equalcont</code>	350, 356, 357, 362		
<code>\pdftexcmds@isprimitive</code>	329, 331		
<code>\pdftexcmds@nopdftex</code>	153, 155, 160, 179, 197, 205, 217		
<code>\pdftexcmds@Patch</code>	493, 499, 556, 565, 700		
<code>\pdftexcmds@PatchDecode</code>	503, 552, 561, 696		
<code>\pdftexcmds@setdraftmode</code>	393, 410, 424, 426		
<code>\pdftexcmds@strip@prefix</code>	246, 252		
<code>\pdftexcmds@temp</code>	157, 168, 169, 171, 173, 174, 175, 176, 214, 223, 224, 247, 262, 263, 264, 265, 266, 267, 268, 269, 305, 323, 324, 377, 382, 390, 616, 627, 639, 641		
		R	
		<code>\RangeCatcodeCheck</code>	1089, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128
		<code>\RangeCatcodeInvalid</code>	1081, 1109, 1110, 1111, 1112
		<code>\repeat</code>	1056, 1068, 1079, 1087, 1102
		<code>\RequirePackage</code>	145, 146, 147, 148, 448
		<code>\RestoreCatcodes</code>	1070, 1073, 1074, 1129
		<code>\romannumeral</code>	552, 561, 696
		S	
		<code>\space</code>	282, 284, 295, 300, 1095, 1096, 1104, 1246, 1267
		<code>\str</code>	1150, 1156, 1158, 1162, 1164
		T	
		<code>\Test</code>	1108, 1131, 1349, 1369, 1377, 1378, 1379, 1380
		<code>\test</code>	1147, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1365, 1371, 1375
		<code>\TestExpect</code>	1367, 1368, 1369
		<code>\TestFrom</code>	1366, 1369
		<code>\TestResult</code>	1352, 1356, 1360
		<code>\the</code>	77, 78, 79, 80, 81, 82, 83, 84, 97, 418, 461, 553, 562, 697, 1075, 1095, 1096
		<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, 440, 446
		<code>\toksdef</code>	489
		U	
		<code>\UnDeFiNeD</code>	1190
		W	
		<code>\write</code>	23, 52, 212, 1144, 1188, 1240
		X	
		<code>\x</code>	14, 15, 18, 22, 26, 28, 51, 56, 66, 75, 87, 251, 252, 256, 310, 315, 417, 420, 469, 470, 478, 482
		Y	
		<code>\y</code>	254, 256, 311, 313, 315, 471, 478, 483
		Z	
		<code>\z</code>	312, 313