

# The pdftextcmds package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2009/12/12 v0.7

## Abstract

LUA<sub>T</sub><sub>E</sub>X provides most of the commands of pdf<sub>T</sub><sub>E</sub>X 1.40. However a number of utility functions are removed. This package tries to fill the gap and implements some of the missing primitive using Lua.

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	General principles . . . . .	2
1.2	Macros . . . . .	3
1.2.1	Additional macro: <code>\pdf@ifprimitive</code> . . . . .	4
1.2.2	Experimental . . . . .	4
<b>2</b>	<b>Implementation</b>	<b>4</b>
2.1	Reload check and package identification . . . . .	5
2.2	Catcodes . . . . .	6
2.3	Load package <code>infwarerr</code> . . . . .	6
2.4	Without LUA <sub>T</sub> <sub>E</sub> X . . . . .	7
2.5	<code>\pdf@primitive</code> , <code>\pdf@ifprimitive</code> . . . . .	8
2.5.1	Using LUA <sub>T</sub> <sub>E</sub> X's <code>tex.enableprimitives</code> . . . . .	8
2.5.2	Trying various names to find the primitives . . . . .	8
2.5.3	Result . . . . .	9
2.6	Xe <sub>T</sub> <sub>E</sub> X . . . . .	9
2.7	<code>\pdf@ifprimitive</code> . . . . .	10
2.8	Load Lua module . . . . .	11
2.9	Lua functions . . . . .	11
2.10	Lua module . . . . .	14
<b>3</b>	<b>Test</b>	<b>18</b>
3.1	Catcode checks for loading . . . . .	18
3.2	Test for <code>\pdf@ifprimitive</code> . . . . .	19
<b>4</b>	<b>Installation</b>	<b>20</b>
4.1	Download . . . . .	20
4.2	Bundle installation . . . . .	20
4.3	Package installation . . . . .	21
4.4	Refresh file name databases . . . . .	21
4.5	Some details for the interested . . . . .	21
<b>5</b>	<b>History</b>	<b>22</b>
	[2007/11/11 v0.1] . . . . .	22
	[2007/11/12 v0.2] . . . . .	22
	[2007/12/12 v0.3] . . . . .	22
	[2009/04/10 v0.4] . . . . .	22
	[2009/09/22 v0.5] . . . . .	22

[2009/09/23 v0.6] . . . . .	22
[2009/12/12 v0.7] . . . . .	22

<b>6 Index</b>	<b>22</b>
----------------	-----------

# 1 Documentation

Some primitives of pdfTeX are not defined by L<sup>A</sup>T<sub>E</sub>X. 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}
```

L<sup>A</sup>T<sub>E</sub>X 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 T<sub>E</sub>X 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 LuaT<sub>E</sub>X:** 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@shellescape`

Same as `\pdfshellescape`. It 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.

`\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 L<sup>A</sup>T<sub>E</sub>X. 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 L<sup>A</sup>T<sub>E</sub>X. 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 T<sub>E</sub>X variants, even the original T<sub>E</sub>X. Example with L<sup>A</sup>T<sub>E</sub>X:

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

### 1.2.2 Experimental

`\pdf@unescapehexnative {\string}`  
`\pdf@escapehexnative {\string}`  
`\pdf@escapepagenative {\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 L<sup>A</sup>T<sub>E</sub>X is running, then its UTF-8 coded strings are used. Thus the full unicode character range is supported. However the result differs from pdfTeX for characters with eight or more bits.

`\pdf@pipe {\cmdline}`

It calls `\cmdline` and returns the output of the external program in the usual manner as byte string (catcode 12, space with catcode 10). The Lua documentation says, that the used `io.popen` may not be available on all platforms. Then macro `\pdf@pipe` is undefined.

## 2 Implementation

1 `\package`

## 2.1 Reload check and package identification

Reload check, especially if the package is not used with L<sup>A</sup>T<sub>E</sub>X.

```
2 \begingroup
3   \catcode44 12 % ,
4   \catcode45 12 % -
5   \catcode46 12 % .
6   \catcode58 12 % :
7   \catcode64 11 % @
8   \catcode123 1 % {
9   \catcode125 2 % }
10  \expandafter\let\expandafter\x\csname ver@pdftexcmds.sty\endcsname
11  \ifx\x\relax % plain-TeX, first loading
12  \else
13    \def\empty{}%
14    \ifx\x\empty % LaTeX, first loading,
15      % variable is initialized, but \ProvidesPackage not yet seen
16    \else
17      \catcode35 6 % #
18      \expandafter\ifx\csname PackageInfo\endcsname\relax
19        \def\x#1#2{%
20          \immediate\write-1{Package #1 Info: #2.}%
21        }%
22      \else
23        \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
24      \fi
25      \x{pdftexcmds}{The package is already loaded}%
26      \aftergroup\endinput
27    \fi
28  \fi
29 \endgroup
```

Package identification:

```
30 \begingroup
31   \catcode35 6 % #
32   \catcode40 12 % (
33   \catcode41 12 % )
34   \catcode44 12 % ,
35   \catcode45 12 % -
36   \catcode46 12 % .
37   \catcode47 12 % /
38   \catcode58 12 % :
39   \catcode64 11 % @
40   \catcode91 12 % [
41   \catcode93 12 % ]
42   \catcode123 1 % {
43   \catcode125 2 % }
44   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
45     \def\x#1#2#3[#4]{\endgroup
46       \immediate\write-1{Package: #3 #4}%
47       \xdef#1{#4}%
48     }%
49   \else
50     \def\x#1#2[#3]{\endgroup
51       #2[#3]}%
52     \ifx#1\@undefined
53       \xdef#1{#3}%
54     \fi
55     \ifx#1\relax
56       \xdef#1{#3}%
57     \fi
58   }%
59 \fi
```

```

60 \expandafter\x\csname ver@pdfTexcmds.sty\endcsname
61 \ProvidesPackage{pdfTexcmds}%
62 [2009/12/12 v0.7 Utility functions of pdfTeX for LuaTeX (H0)]

```

## 2.2 Catcodes

```

63 \begingroup
64   \catcode123 1 % {
65   \catcode125 2 % }
66   \def\x{\endgroup
67     \expandafter\edef\csname pdfTexcmds@AtEnd\endcsname{%
68       \catcode35 \the\catcode35\relax
69       \catcode64 \the\catcode64\relax
70       \catcode123 \the\catcode123\relax
71       \catcode125 \the\catcode125\relax
72     }%
73   }%
74 \x
75 \catcode35 6 % #
76 \catcode64 11 % @
77 \catcode123 1 % {
78 \catcode125 2 % }
79 \def\TMP@EnsureCode#1#2{%
80   \edef\pdfTexcmds@AtEnd{%
81     \pdfTexcmds@AtEnd
82     \catcode#1 \the\catcode#1\relax
83   }%
84   \catcode#1 #2\relax
85 }
86 \TMP@EnsureCode{10}{12}% ^^J
87 \TMP@EnsureCode{33}{12}% !
88 \TMP@EnsureCode{34}{12}% "
89 \TMP@EnsureCode{39}{12}% '
90 \TMP@EnsureCode{40}{12}% (
91 \TMP@EnsureCode{41}{12}% )
92 \TMP@EnsureCode{42}{12}% *
93 \TMP@EnsureCode{43}{12}% +
94 \TMP@EnsureCode{44}{12}% ,
95 \TMP@EnsureCode{45}{12}% -
96 \TMP@EnsureCode{46}{12}% .
97 \TMP@EnsureCode{47}{12}% /
98 \TMP@EnsureCode{58}{12}% :
99 \TMP@EnsureCode{60}{12}% <
100 \TMP@EnsureCode{61}{12}% =
101 \TMP@EnsureCode{62}{12}% >
102 \TMP@EnsureCode{94}{7}% ^ (superscript)
103 \TMP@EnsureCode{95}{12}% _ (other)
104 \TMP@EnsureCode{96}{12}% `
105 \TMP@EnsureCode{126}{12}% ~ (other)
106 \edef\pdfTexcmds@AtEnd{%
107   \pdfTexcmds@AtEnd
108   \escapechar=\number\escapechar\relax
109 }
110 \escapechar=92 %

```

## 2.3 Load package infwarerr

```

111 \begingroup\expandafter\expandafter\expandafter\endgroup
112 \expandafter\ifx\csname RequirePackage\endcsname\relax
113   \input infwarerr.sty\relax
114   \input ifluatex.sty\relax
115   \input ltxcmds.sty\relax
116 \else

```

```

117 \RequirePackage{infwarerr}[2007/09/09]%
118 \RequirePackage{ifluatex}[2009/04/10]%
119 \RequirePackage{ltxcmds}%
120 \fi

```

## 2.4 Without LuaTeX

```

121 \ifluatex
122 \else
123   \@PackageInfoNoLine{pdftexcmds}{LuaTeX not detected}%
124   \def\pdftexcmds@nopdftex{%
125     \@PackageInfoNoLine{pdftexcmds}{pdfTeX >= 1.30 not detected}%
126     \let\pdftexcmds@nopdftex\relax
127   }%
128   \def\pdftexcmds@temp#1{%
129     \begingroup\expandafter\expandafter\expandafter\endgroup
130     \expandafter\ifx\csname pdf#1\endcsname\relax
131       \pdftexcmds@nopdftex
132     \else
133       \expandafter\def\csname pdf#1\endcsname\expandafter\endcsname
134       \expandafter##\expandafter{%
135         \csname pdf#1\endcsname
136       }%
137     \fi
138   }%
139   \pdftexcmds@temp{strcmp}%
140   \pdftexcmds@temp{escapehex}%
141   \let\pdf@escapehexnative\pdf@escapehex
142   \pdftexcmds@temp{unescapehex}%
143   \let\pdf@unescapehexnative\pdf@unescapehex
144   \pdftexcmds@temp{escapestring}%
145   \pdftexcmds@temp{escapename}%
146   \pdftexcmds@temp{filesize}%
147   \pdftexcmds@temp{filemoddate}%
148   \begingroup\expandafter\expandafter\expandafter\endgroup
149   \expandafter\ifx\csname pdfshellescape\endcsname\relax
150     \pdftexcmds@nopdftex
151   \else
152     \def\pdf@shellescape{%
153       \pdfshellescape
154     }%
155   \fi
156   \begingroup\expandafter\expandafter\expandafter\endgroup
157   \expandafter\ifx\csname pdffiledump\endcsname\relax
158     \pdftexcmds@nopdftex
159   \else
160     \def\pdf@filedump#1#2#3{%
161       \pdffiledump offset#1 length#2{#3}%
162     }%
163   \fi
164   \begingroup\expandafter\expandafter\expandafter\endgroup
165   \expandafter\ifx\csname pdfmdfivesum\endcsname\relax
166     \pdftexcmds@nopdftex
167   \else
168     \def\pdf@mdfivesum#\pdfmdfivesum{%
169       \let\pdf@mdfivesumnative\pdf@mdfivesum
170       \def\pdf@filemdfivesum#\pdfmdfivesum file}%
171   \fi
172   \def\pdf@system#1{%
173     \immediate\write18%
174   }%
175 \fi

```

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

Since version 1.40.0 pdf<sub>TEX</sub> has \pdfprimitive and \ifpdfprimitive. And \pdfprimitive was fixed in version 1.40.4.

Xe<sub>TEX</sub> provides them under the name \primitive and \ifprimitive. L<sub>A</sub>T<sub>E</sub>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<sub>TEX</sub>'s tex.enableprimitives

```
176 \ifluatex
```

```
\pdftexcmds@directlua
```

```
177 \ifnum\luatexversion<36 %
178   \def\pdftexcmds@directlua{\directlua0 }%
179 \else
180   \let\pdftexcmds@directlua\directlua
181 \fi

182 \begingroup
183   \newlinechar=10 %
184   \endlinechar=\newlinechar
185   \pdftexcmds@directlua{%
186     if tex.enableprimitives then
187       tex.enableprimitives('pdf@', {'primitive', 'ifprimitive'})
188       tex.enableprimitives('', {'luaescapestring'})
189     end
190   }%
191 \endgroup %
192 \fi
```

### 2.5.2 Trying various names to find the primitives

```
\pdftexcmds@strip@prefix
```

```
193 \def\pdftexcmds@strip@prefix#1>{}

194 \def\pdftexcmds@temp#1#2#3{%
195   \begingroup\expandafter\expandafter\expandafter\endgroup
196   \expandafter\ifx\csname pdf@#1\endcsname\relax
197     \begingroup
198       \def\x{#3}%
199       \edef\x{\expandafter\pdftexcmds@strip@prefix\meaning\x}%
200       \escapechar=-1 %
201       \edef\y{\expandafter\meaning\csname#2\endcsname}%
202       \expandafter\endgroup
203       \ifx\x\y
204         \expandafter\let\csname pdf@#1\expandafter\endcsname
205         \csname #2\endcsname
206       \fi
207     \fi
208 }
```

```
\pdf@primitive
```

```
209 \pdftexcmds@temp{primitive}{pdfprimitive}{pdfprimitive}% pdfTeX, LuaTeX
210 \pdftexcmds@temp{primitive}{primitive}{primitive}% XeTeX
211 \pdftexcmds@temp{primitive}{luatexpprimitive}{pdfprimitive}% LuaTeX
212 \pdftexcmds@temp{primitive}{luatexpdfprimitive}{pdfprimitive}% LuaTeX
```

`\pdf@ifprimitive`

```

213 \pdfifprimitive{ifprimitive}{ifpdfprimitive}{ifpdfprimitive}% pdfTeX, LuaTeX
214 \pdfifprimitive{ifprimitive}{ifprimitive}{ifprimitive}% XeTeX
215 \pdfifprimitive{ifprimitive}{luatexifprimitive}{ifpdfprimitive}% LuaTeX
216 \pdfifprimitive{ifprimitive}{luatexifpdfprimitive}{ifpdfprimitive}% LuaTeX

```

Disable broken `\pdfprimitive`.

```

217 \begingroup
218   \expandafter\ifx\csname pdf@primitive\endcsname\relax
219   \else
220     \expandafter\ifx\csname pdftexversion\endcsname\relax
221     \else
222       \ifnum\pdftexversion=140 %
223         \expandafter\ifx\csname pdftexrevision\endcsname\relax
224         \else
225           \ifnum\pdftexrevision<4 %
226           \endgroup
227           \let\pdf@primitive\@undefined
228           \@PackageInfoNoLine{pdftexcmds}{%
229             \string\pdf@primitive disabled, because\MessageBreak
230             \string\pdfprimitive\space is broken until pdfTeX 1.40.4%
231           }%
232         \begingroup
233         \fi
234       \fi
235     \fi
236   \fi
237 \fi
238 \endgroup

```

### 2.5.3 Result

```

239 \begingroup
240   \@PackageInfoNoLine{pdftexcmds}{%
241     \string\pdf@primitive\space is %
242     \expandafter\ifx\csname pdf@primitive\endcsname\relax not \fi
243     available%
244   }%
245   \@PackageInfoNoLine{pdftexcmds}{%
246     \string\pdf@ifprimitive\space is %
247     \expandafter\ifx\csname pdf@ifprimitive\endcsname\relax not \fi
248     available%
249   }%
250 \endgroup

```

## 2.6 XeTeX

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

```

251 \def\pdfifprimitive#1{%
252   \begingroup\expandafter\expandafter\expandafter\endgroup
253   \expandafter\ifx\csname pdf@#1\endcsname\relax
254   \begingroup
255     \escapechar=-1 %
256     \edef\x{\expandafter\meaning\csname#1\endcsname}%
257     \def\y{#1}%
258     \def\z##1->{}%
259     \edef\y{\expandafter\z\meaning\y}%
260   \expandafter\endgroup
261   \ifx\x\y
262     \expandafter\def\csname pdf@#1\endcsname\expandafter\endcsname
263     \expandafter{%
264       \csname#1\endcsname
265     }%

```

```

266 \fi
267 \fi
268 }%
269 \pdfTexcmds@temp{shellescape}%
270 \pdfTexcmds@temp{strcmp}%

```

## 2.7 \pdf@isprimitive

```

271 \def\pdf@isprimitive{%
272 \begingroup\expandafter\expandafter\expandafter\endgroup
273 \expandafter\ifx\csname pdf@strcmp\endcsname\relax
274 \long\def\pdf@isprimitive##1{%
275 \expandafter\pdfTexcmds@isprimitive\expandafter{\meaning##1}%
276 }%
277 \long\def\pdfTexcmds@isprimitive##1##2{%
278 \expandafter\pdfTexcmds@isprimitive\expandafter{\string##2}{##1}%
279 }%
280 \def\pdfTexcmds@isprimitive##1##2{%
281 \ifnum0\pdfTexcmds@equal##1\delimiter##2\delimiter=1 %
282 \expandafter\ltx@firstoftwo
283 \else
284 \expandafter\ltx@secondoftwo
285 \fi
286 }%
287 \def\pdfTexcmds@equal##1##2\delimiter##3##4\delimiter{%
288 \ifx##1##3%
289 \ifx\relax##2##4\relax
290 1%
291 \else
292 \ifx\relax##2\relax
293 \else
294 \ifx\relax##4\relax
295 \else
296 \pdfTexcmds@equalcont{##2}{##4}%
297 \fi
298 \fi
299 \fi
300 \fi
301 }%
302 \def\pdfTexcmds@equalcont##1{%
303 \def\pdfTexcmds@equalcont####1####2##1##1##1##1{%
304 ##1##1##1##1%
305 \pdfTexcmds@equal####1\delimiter####2\delimiter
306 }%
307 }%
308 \expandafter\pdfTexcmds@equalcont\csname fi\endcsname
309 \else
310 \long\def\pdf@isprimitive##1##2{%
311 \ifnum\pdf@strcmp{\meaning##1}{\string##2}=0 %
312 \expandafter\ltx@firstoftwo
313 \else
314 \expandafter\ltx@secondoftwo
315 \fi
316 }%
317 \fi
318 }
319 \ifluatex
320 \else
321 \pdf@isprimitive
322 \pdfTexcmds@AtEnd
323 \expandafter\endinput
324 \fi

```

## 2.8 Load Lua module

```
325 \begingroup\expandafter\expandafter\expandafter\endgroup
326 \expandafter\ifx\csname RequirePackage\endcsname\relax
327   \input luatex-loader.sty\relax
328 \else
329   \RequirePackage{luatex-loader}[2009/04/10]%
330 \fi
331 \pdftexcmds@directlua{%
332   require("oberdiek.pdftexcmds")%
333 }
```

## 2.9 Lua functions

\pdftexcmds@toks

```
334 \begingroup\expandafter\expandafter\expandafter\endgroup
335 \expandafter\ifx\csname newtoks\endcsname\relax
336   \toksdef\pdftexcmds@toks=0 %
337 \else
338   \csname newtoks\endcsname\pdftexcmds@toks
339 \fi

340 \ifnum\luatexversion<36 %
341 \else
342   \catcode'\0=9 %
343 \fi
```

\pdf@strcmp

```
344 \long\def\pdf@strcmp#1#2{%
345   \directlua0{%
346     oberdiek.pdftexcmds.strptime("\luaescapestring{#1}",%
347       "\luaescapestring{#2}")%
348   }%
349 }%

350 \pdf@isprimitive
```

\pdf@escapehex

```
351 \long\def\pdf@escapehex#1{%
352   \directlua0{%
353     oberdiek.pdftexcmds.escapehex("\luaescapestring{#1}", "byte")%
354   }%
355 }%
```

\pdf@escapehexnative

```
356 \long\def\pdf@escapehexnative#1{%
357   \directlua0{%
358     oberdiek.pdftexcmds.escapehex("\luaescapestring{#1}")%
359   }%
360 }%
```

\pdf@unescapehex

```
361 \def\pdf@unescapehex#1{%
362   \the\expandafter\pdftexcmds@toks
363   \directlua0{%
364     oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
365     oberdiek.pdftexcmds.unescapehex("\luaescapestring{#1}", "byte")%
366   }%
367 }%
```

\pdf@unescapehexnative

```
368 \def\pdf@unescapehexnative#1{%
```

```

369 \the\expandafter\pdf texcmds@toks
370 \directlua0{%
371   oberdiek.pdf texcmds.toks="pdf texcmds@toks"%
372   oberdiek.pdf texcmds.unescapehex("\luaescapestring{#1}")%
373 }%
374 }%

\pdf@escapestring
375 \long\def\pdf@escapestring#1{%
376   \directlua0{%
377     oberdiek.pdf texcmds.escapestring("\luaescapestring{#1}", "byte")%
378   }%
379 }

\pdf@escapename
380 \long\def\pdf@escapename#1{%
381   \directlua0{%
382     oberdiek.pdf texcmds.escapename("\luaescapestring{#1}", "byte")%
383   }%
384 }

\pdf@escapenamenative
385 \long\def\pdf@escapenamenative#1{%
386   \directlua0{%
387     oberdiek.pdf texcmds.escapename("\luaescapestring{#1}")%
388   }%
389 }

\pdf@filesize
390 \def\pdf@filesize#1{%
391   \directlua0{%
392     oberdiek.pdf texcmds.filesize("\luaescapestring{#1}")%
393   }%
394 }

\pdf@filemoddate
395 \def\pdf@filemoddate#1{%
396   \directlua0{%
397     oberdiek.pdf texcmds.filemoddate("\luaescapestring{#1}")%
398   }%
399 }

\pdf@filedump
400 \def\pdf@filedump#1#2#3{%
401   \directlua0{%
402     oberdiek.pdf texcmds.filedump("\luaescapestring{\number#1}",%
403                                   "\luaescapestring{\number#2}",%
404                                   "\luaescapestring{\number#3}")%
405   }%
406 }%

\pdf@mdfivesum
407 \long\def\pdf@mdfivesum#1{%
408   \directlua0{%
409     oberdiek.pdf texcmds.mdfivesum("\luaescapestring{#1}", "byte")%
410   }%
411 }%

\pdf@mdfivesumnative
412 \long\def\pdf@mdfivesumnative#1{%
413   \directlua0{%

```

```

414 oberdiek.pdf texcmds.mdfivesum("\luaescapestring{#1}")%
415 }%
416 }%

\pdf@filemdfivesum

417 \def\pdf@filemdfivesum#1{%
418 \directlua0{%
419 oberdiek.pdf texcmds.filemdfivesum("\luaescapestring{#1}")%
420 }%
421 }%

\pdf@shellescape

422 \def\pdf@shellescape{%
423 \directlua0{%
424 oberdiek.pdf texcmds.shellescape()%
425 }%
426 }

\pdf@system

427 \def\pdf@system#1{%
428 \directlua0{%
429 oberdiek.pdf texcmds.system("\luaescapestring{#1}")%
430 }%
431 }

\pdf@lastsystemstatus

432 \def\pdf@lastsystemstatus{%
433 \directlua0{%
434 oberdiek.pdf texcmds.lastsystemstatus()%
435 }%
436 }

\pdf@lastsystemexit

437 \def\pdf@lastsystemexit{%
438 \directlua0{%
439 oberdiek.pdf texcmds.lastsystemexit()%
440 }%
441 }

442 \catcode'\0=12 %

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

443 \ifnum0%
444 \pdf texcmds@directlua{%
445 if io.popen then %
446 tex.write("1")%
447 end%
448 }%
449 =1 %
450 \def\pdf@pipe#1{%
451 \the\expandafter\pdf texcmds@toks
452 \pdf texcmds@directlua{%
453 oberdiek.pdf texcmds.toks="pdf texcmds@toks"%
454 oberdiek.pdf texcmds.pipe("\luaescapestring{#1}")%
455 }%
456 }%
457 \fi

458 \pdf texcmds@AtEnd
459 \endpackage

```

## 2.10 Lua module

```
460 (*lua)
461 module("oberdiek.pdfdoccmds", package.seeall)
462 local systemexitstatus
463 function strcmp(A, B)
464   if A == B then
465     tex.write("0")
466   elseif A < B then
467     tex.write("-1")
468   else
469     tex.write("1")
470   end
471 end
472 local function utf8_to_byte(str)
473   local i = 0
474   local n = string.len(str)
475   local t = {}
476   while i < n do
477     i = i + 1
478     local a = string.byte(str, i)
479     if a < 128 then
480       table.insert(t, string.char(a))
481     else
482       if a >= 192 and i < n then
483         i = i + 1
484         local b = string.byte(str, i)
485         if b < 128 or b >= 192 then
486           i = i - 1
487         elseif a == 194 then
488           table.insert(t, string.char(b))
489         elseif a == 195 then
490           table.insert(t, string.char(b + 64))
491         end
492       end
493     end
494   end
495   return table.concat(t)
496 end
497 function escapehex(str, mode)
498   if mode == "byte" then
499     str = utf8_to_byte(str)
500   end
501   tex.write((string.gsub(str, ".",
502     function (ch)
503       return string.format("%02X", string.byte(ch))
504     end
505   )))
506 end
```

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

```
507 function unescapehex(str, mode)
508   local a = 0
509   local first = true
510   local result = {}
511   for i = 1, string.len(str), 1 do
512     local ch = string.byte(str, i)
513     if ch >= 48 and ch <= 57 then
514       ch = ch - 48
515     elseif ch >= 65 and ch <= 70 then
516       ch = ch - 55
517     elseif ch >= 97 and ch <= 102 then
```

```

518     ch = ch - 87
519 else
520     ch = nil
521 end
522 if ch then
523     if first then
524         a = ch * 16
525         first = false
526     else
527         table.insert(result, a + ch)
528         first = true
529     end
530 end
531 end
532 if not first then
533     table.insert(result, a)
534 end
535 if mode == "byte" then
536     local utf8 = {}
537     for i, a in ipairs(result) do
538         if a < 128 then
539             table.insert(utf8, a)
540         else
541             if a < 192 then
542                 table.insert(utf8, 194)
543                 a = a - 128
544             else
545                 table.insert(utf8, 195)
546                 a = a - 192
547             end
548             table.insert(utf8, a + 128)
549         end
550     end
551     result = utf8
552 end
553 tex.settoks(toks, string.char(unpack(result)))
554 end

```

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

```

555 function escapestring(str, mode)
556     if mode == "byte" then
557         str = utf8_to_byte(str)
558     end
559     tex.write((string.gsub(str, ".",
560         function (ch)
561             local b = string.byte(ch)
562             if b < 33 or b > 126 then
563                 return string.format("\\%.3o", b)
564             end
565             if b == 40 or b == 41 or b == 92 then
566                 return "\\" .. ch
567             end

```

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

```

568         return nil
569     end
570 )))
571 end

```

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

```

572 function escapename(str, mode)
573     if mode == "byte" then
574         str = utf8_to_byte(str)
575     end
576     tex.write((string.gsub(str, ".",

```

```

577     function (ch)
578         local b = string.byte(ch)
579         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.

```

580         return ""
581     end
582     if b <= 32 or b >= 127
583         or b == 35 or b == 37 or b == 40 or b == 41
584         or b == 47 or b == 60 or b == 62 or b == 91
585         or b == 93 or b == 123 or b == 125 then
586         return string.format("#%.2X", b)
587     else

```

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

```

588         return nil
589     end
590 end
591 )))
592 end
593 function filesize(filename)
594     local foundfile = kpse.find_file(filename, "tex", true)
595     if foundfile then
596         local size = lfs.attributes(foundfile, "size")
597         if size then
598             tex.write(size)
599         end
600     end
601 end

```

See procedure makepdftime in file utils.c of pdfTeX.

```

602 function filemoddate(filename)
603     local foundfile = kpse.find_file(filename, "tex", true)
604     if foundfile then
605         local date = lfs.attributes(foundfile, "modification")
606         if date then
607             local d = os.date("*t", date)
608             if d.sec >= 60 then
609                 d.sec = 59
610             end
611             local u = os.date("!*t", date)
612             local off = 60 * (d.hour - u.hour) + d.min - u.min
613             if d.year ~= u.year then
614                 if d.year > u.year then
615                     off = off + 1440
616                 else
617                     off = off - 1440
618                 end
619             elseif d.yday ~= u.yday then
620                 if d.yday > u.yday then
621                     off = off + 1440
622                 else
623                     off = off - 1440
624                 end
625             end
626             local timezone
627             if off == 0 then
628                 timezone = "Z"
629             else
630                 local hours = math.floor(off / 60)
631                 local mins = math.abs(off - hours * 60)
632                 timezone = string.format("%+03d'%02d'", hours, mins)
633             end
634             tex.write(string.format("D:%04d%02d%02d%02d%02d%02d%s",

```

```

635         d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
636     end
637 end
638 end
639 function filedump(offset, length, filename)
640     length = tonumber(length)
641     if length and length > 0 then
642         local foundfile = kpse.find_file(filename, "tex", true)
643         if foundfile then
644             offset = tonumber(offset)
645             if not offset then
646                 offset = 0
647             end
648             local filehandle = io.open(foundfile, "r")
649             if filehandle then
650                 if offset > 0 then
651                     filehandle:seek("set", offset)
652                 end
653                 local dump = filehandle:read(length)
654                 escapehex(dump)
655             end
656         end
657     end
658 end
659 function md5sum(str, mode)
660     if mode == "byte" then
661         str = utf8_to_byte(str)
662     end
663     escapehex(md5.sum(str))
664 end
665 function filemd5sum(filename)
666     local foundfile = kpse.find_file(filename, "tex", true)
667     if foundfile then
668         local filehandle = io.open(foundfile, "r")
669         if filehandle then
670             local contents = filehandle:read("*a")
671             escapehex(md5.sum(contents))
672         end
673     end
674 end
675 function shellescape()
676     if os.execute then
677         tex.write("1")
678     else
679         tex.write("0")
680     end
681 end
682 function system(cmdline)
683     systemexitstatus = nil
684     texio.write_nl("log", "system(" .. cmdline .. ") ")
685     if os.execute then
686         texio.write("log", "executed.")
687         systemexitstatus = os.execute(cmdline)
688     else
689         texio.write("log", "disabled.")
690     end
691 end
692 function lastsystemstatus()
693     local result = tonumber(systemexitstatus)
694     if result then
695         local x = math.floor(result / 256)
696         tex.write(result - 256 * math.floor(result / 256))

```

```

697 end
698 end
699 function lastsysteemexit()
700   local result = tonumber(systemexitstatus)
701   if result then
702     tex.write(math.floor(result / 256))
703   end
704 end
705 function pipe(cmdline)
706   local result
707   systemexitstatus = nil
708   texio.write_nl("log", "pipe(" .. cmdline .. ") ")
709   if io.popen then
710     texio.write("log", "executed.")
711     local handle = io.popen(cmdline, "r")
712     if handle then
713       result = handle:read("*a")
714       handle:close()
715     end
716   else
717     texio.write("log", "disabled.")
718   end
719   if result then
720     tex.settoks(toks, result)
721   else
722     tex.settoks(toks, "")
723   end
724 end
725  $\langle$ /lua $\rangle$ 

```

### 3 Test

#### 3.1 Catcode checks for loading

```

726  $\langle$ *test1 $\rangle$ 
727 \catcode'\{=1 %
728 \catcode'\}=2 %
729 \catcode'\#=6 %
730 \catcode'\@=11 %
731 \expandafter\ifx\csname count@\endcsname\relax
732   \countdef\count@=255 %
733 \fi
734 \expandafter\ifx\csname @gobble\endcsname\relax
735   \long\def\@gobble#1{}%
736 \fi
737 \expandafter\ifx\csname @firstofone\endcsname\relax
738   \long\def\@firstofone#1{#1}%
739 \fi
740 \expandafter\ifx\csname loop\endcsname\relax
741   \expandafter\@firstofone
742 \else
743   \expandafter\@gobble
744 \fi
745 {%
746   \def\loop#1\repeat{%
747     \def\body{#1}%
748     \iterate
749   }%
750   \def\iterate{%
751     \body
752     \let\next\iterate
753   \else

```

```

754     \let\next\relax
755     \fi
756     \next
757 }%
758 \let\repeat=\fi
759 }%
760 \def\RestoreCatcodes{}
761 \count@=0 %
762 \loop
763   \edef\RestoreCatcodes{%
764     \RestoreCatcodes
765     \catcode\the\count@=\the\catcode\count@\relax
766   }%
767 \ifnum\count@<255 %
768   \advance\count@ 1 %
769 \repeat
770
771 \def\RangeCatcodeInvalid#1#2{%
772   \count@=#1\relax
773   \loop
774     \catcode\count@=15 %
775   \ifnum\count@<#2\relax
776     \advance\count@ 1 %
777   \repeat
778 }
779 \expandafter\ifx\csname LoadCommand\endcsname\relax
780   \def\LoadCommand{\input pdftexcmds.sty\relax}%
781 \fi
782 \def\Test{%
783   \RangeCatcodeInvalid{0}{47}%
784   \RangeCatcodeInvalid{58}{64}%
785   \RangeCatcodeInvalid{91}{96}%
786   \RangeCatcodeInvalid{123}{255}%
787   \catcode'\@=12 %
788   \catcode'\=0 %
789   \catcode'\{=1 %
790   \catcode'\}=2 %
791   \catcode'\#=6 %
792   \catcode'\[=12 %
793   \catcode'\]=12 %
794   \catcode'\%=14 %
795   \catcode'\ =10 %
796   \catcode13=5 %
797   \LoadCommand
798   \RestoreCatcodes
799 }
800 \Test
801 \csname @@end\endcsname
802 \end
803 </test1>

```

### 3.2 Test for \pdf@isprimitive

```

804 <*test2>
805 \catcode'\{=1 %
806 \catcode'\}=2 %
807 \catcode'\#=6 %
808 \catcode'\@=11 %
809 \input pdftexcmds.sty\relax
810 \def\msg#1{%
811   \begingroup
812     \escapechar=92 %
813     \immediate\write16{#1}%

```

```

814 \endgroup
815 }
816 \long\def\test#1#2#3#4{%
817 \begingroup
818   #4%
819   \def\str{%
820     Test \string\pdf@isprimitive
821     {\string #1}{\string #2}{...}: %
822   }%
823   \pdf@isprimitive{#1}{#2}{%
824     \ifx#3Y%
825       \msg{\str true ==> OK.}%
826     \else
827       \errmessage{\str false ==> FAILED}%
828     \fi
829   }{%
830     \ifx#3Y%
831       \errmessage{\str true ==> FAILED}%
832     \else
833       \msg{\str false ==> OK.}%
834     \fi
835   }%
836 \endgroup
837 }
838 \test\relax\relax Y{}
839 \test\foobar\relax Y{\let\foobar\relax}
840 \test\foobar\relax N{}
841 \test\hbox\hbox Y{}
842 \test\foobar@hbox\hbox Y{\let\foobar@hbox\hbox}
843 \test\if\if Y{}
844 \test\if\ifx N{}
845 \test\ifx\if N{}
846 \test\par\par Y{}
847 \test\hbox\par N{}
848 \test\par\hbox N{}
849 \csname @@end\endcsname\end
850 </test2>

```

## 4 Installation

### 4.1 Download

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

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

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

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

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

*TDS* refers to the standard “A Directory Structure for  $\TeX$  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
```

---

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

**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- $\text{\TeX}$ :

```
tex pdftexcmds.dtx
```

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

```
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
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  $\text{\TeX}$  distribution (`te $\text{\TeX}$` , `mik $\text{\TeX}$` , ...) relies on file name databases, you must refresh these. For example, `te $\text{\TeX}$`  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  $\text{\LaTeX}$ .** The `.dtx` chooses its action depending on the format:

**plain- $\text{\TeX}$ :** Run `docstrip` and extract the files.

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

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

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

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

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

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

An example follows how to generate the documentation with pdfL<sup>A</sup>T<sub>E</sub>X:

```
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 L<sup>A</sup>T<sub>E</sub>X 0.36.

[2009/09/22 v0.5]

- `\pdf@primitive`, `\pdf@ifprimitive` added.
- Xe<sub>T</sub><sub>E</sub>X'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.

## 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; numbers in *roman* refer to the code lines where the entry is used.

	<b>Symbols</b>	.....	123, 125, 228, 240, 245
<code>\#</code>	.....	729, 791, 807	<code>\@firstofone</code> ..... 738, 741
<code>\%</code>	.....	794	<code>\@gobble</code> ..... 735, 743
<code>\@</code>	.....	730, 787, 808	<code>\@undefined</code> ..... 52, 227
<code>\@PackageInfoNoLine</code>	.....		<code>\[</code> ..... 792

\\	563, 566, 788	I	
\{	727, 789, 805	\if	843, 844, 845
\}	728, 790, 806	\ifluatex	121, 176, 319
\]	793	\ifnum	177, 222, 225, 281, 311, 340, 443, 767, 775
<b>Numbers</b>		\ifx	11, 14, 18, 44, 52, 55, 112, 130, 149, 157, 165, 196, 203, 218, 220, 223, 242, 247, 253, 261, 273, 288, 289, 292, 294, 326, 335, 731, 734, 737, 740, 779, 824, 830, 844, 845
\0	342, 442	\immediate	20, 46, 173, 813
\_	795	\input	113, 114, 115, 327, 780, 809
<b>A</b>		\iterate	748, 750, 752
\advance	768, 776	<b>L</b>	
\aftergroup	26	\LoadCommand	780, 797
<b>B</b>		\loop	746, 762, 773
\body	747, 751	\ltx@firstoftwo	282, 312
<b>C</b>		\ltx@secondoftwo	284, 314
\catcode	3, 4, 5, 6, 7, 8, 9, 17, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 64, 65, 68, 69, 70, 71, 75, 76, 77, 78, 82, 84, 342, 442, 727, 728, 729, 730, 765, 774, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 805, 806, 807, 808	\luaescapestring	346, 347, 353, 358, 365, 372, 377, 382, 387, 392, 397, 402, 403, 404, 409, 414, 419, 429, 454
\count@	732, 761, 765, 767, 768, 772, 774, 775, 776	\luatexversion	177, 340
\countdef	732	<b>M</b>	
\csname	10, 18, 44, 60, 67, 112, 130, 133, 135, 149, 157, 165, 196, 201, 204, 205, 218, 220, 223, 242, 247, 253, 256, 262, 264, 273, 308, 326, 335, 338, 731, 734, 737, 740, 779, 801, 849	\meaning	199, 201, 256, 259, 275, 311
<b>D</b>		\MessageBreak	229
\delimiter	281, 287, 305	\msg	810, 825, 833
\directlua	178, 180, 345, 352, 357, 363, 370, 376, 381, 386, 391, 396, 401, 408, 413, 418, 423, 428, 433, 438	<b>N</b>	
<b>E</b>		\newlinechar	183, 184
\empty	13, 14	\next	752, 754, 756
\end	802, 849	\number	108, 402, 403
\endcsname	10, 18, 44, 60, 67, 112, 130, 133, 135, 149, 157, 165, 196, 201, 204, 205, 218, 220, 223, 242, 247, 253, 256, 262, 264, 273, 308, 326, 335, 338, 731, 734, 737, 740, 779, 801, 849	<b>P</b>	
\endinput	26, 323	\PackageInfo	23
\endlinechar	184	\par	846, 847, 848
\errmessage	827, 831	\pdf@escapehex	3, 141, 351
\escapechar	108, 110, 200, 255, 812	\pdf@escapehexnative	141, 356
<b>F</b>		\pdf@escapename	380
\foobar	839, 840	\pdf@escapenamenative	385
\foobar@hbox	842	\pdf@escapestring	375
<b>H</b>		\pdf@filedump	3, 160, 400
\hbox	841, 842, 847, 848	\pdf@filemdfivesum	3, 170, 417
		\pdf@filemoddate	3, 395
		\pdf@filesize	3, 390
		\pdf@ifprimitive	4, 213, 246
		\pdf@isprimitive	4, 271, 274, 310, 321, 350, 820, 823
		\pdf@lastsystemexit	437
		\pdf@lastsystemstatus	432
		\pdf@mdfivesum	3, 168, 169, 407
		\pdf@mdfivesumnative	169, 412
		\pdf@pipe	4, 443
		\pdf@primitive	4, 209, 227, 229, 241
		\pdf@shellescape	3, 152, 422
		\pdf@strcmp	3, 311, 344
		\pdf@system	3, 172, 427
		\pdf@unescapehex	3, 143, 361
		\pdf@unescapehexnative	4, 143, 368
		\pdffiledump	161

