

The pdfescape package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/11/11 v1.8

Abstract

This package implements pdfTeX's escape features (`\pdfescapehex`, `\pdfunescapehex`, `\pdfescapechar`, `\pdfescapestring`) using TeX or ϵ -TeX.

Contents

1	Documentation	2
1.1	Additional unescape macros	2
1.2	Sanitizing macro	3
2	Implementation	3
2.1	Reload check and package identification	3
2.2	Catcodes	4
2.3	Sanitizing	4
2.3.1	Space characters	5
2.3.2	Space normalization	5
2.4	<code>\EdefUnescapeName</code>	5
2.5	<code>\EdefUnescapeString</code>	7
2.6	User macros (pdfTeXanalogues)	10
2.7	Help macros	11
2.7.1	Characters	11
2.7.2	Switch for ϵ -TeX	11
2.8	Conversions	11
2.8.1	Conversion to hex string	11
2.8.2	Character code to octal number	12
2.8.3	Unpack hex string	13
2.8.4	Conversion to PDF name	14
2.8.5	Conversion to PDF string	15
3	Test	16
3.1	Catcode checks for loading	16
3.2	Macro tests	17
3.3	Test with <code>\pdfescape...</code> commands	17
3.4	Test without <code>\pdfescape...</code> , with ϵ -TeX	17
3.5	Test without <code>\pdfescape...</code> and ϵ -TeX	17
3.6	Test with L ^A T _E X	17
3.7	Check/ensure test preconditions	18
3.7.1	Check <code>\pdfescape...</code>	18
3.7.2	Check ϵ -TeX	18
3.7.3	Check L ^A T _E X	18
3.8	Common part	18

4	Installation	24
4.1	Download	24
4.2	Bundle installation	24
4.3	Package installation	25
4.4	Refresh file name databases	25
4.5	Some details for the interested	25
5	History	26
[2007/02/21 v1.0]		26
[2007/02/25 v1.1]		26
[2007/03/20 v1.2]		26
[2007/04/11 v1.3]		26
[2007/04/21 v1.4]		26
[2007/08/27 v1.5]		26
[2007/09/09 v1.6]		26
[2007/10/27 v1.7]		26
[2007/11/11 v1.8]		27
6	Index	27

1 Documentation

```
\EdefEscapeHex {⟨cmd⟩} {⟨string⟩}
\EdefUnescapeHex {⟨cmd⟩} {⟨string⟩}
\EdefEscapeName {⟨cmd⟩} {⟨string⟩}
\EdefEscapeString {⟨cmd⟩} {⟨string⟩}
```

These commands converts $\langle string \rangle$ and stores the result in macro $\langle cmd \rangle$. The conversion result is the same as the conversion of the corresponding pdfTeX's primitives. Note that the argument $\langle string \rangle$ is expanded before the conversion.

For example, if pdfTeX ≥ 1.30 is present, then `\EdefEscapeHex` becomes to:

```
\def\EdefEscapeHex#1#2{%
  \edef#1{\pdfescapehex{#2}}%
}
```

The package provides implementations for the case that pdfTeX is not present (or too old). Even ε -TeX can be missing, however it is used if it is detected.

Babel. The input strings may contain shorthand characters of package `babel`.

1.1 Additional unescape macros

```
\EdefUnescapeName {⟨cmd⟩} {⟨string⟩}
```

Sequences of a hash sign with two hexadecimal digits are converted to the corresponding character (PDF-1.2). A hash sign that is not followed by two hexadecimal digits is left unchanged. The catcodes in the result string follow TeX's conventions. The space has catcode 10 (space) and the other characters have catcode 12 (other).

```
\EdefUnescapeString {⟨cmd⟩} {⟨string⟩}
```

Macro $\langle cmd \rangle$ stores the unescaped string in $\langle string \rangle$. All the rules for literal strings are implemented, see PDF specification. The catcodes in the result string follow TeX's conventions.

1.2 Sanitizing macro

`\EdefSanitize {\langle cmd \rangle} {\langle string \rangle}`

Argument $\langle string \rangle$ is expanded, converted to a string of tokens with catcode 12 (other) and space tokens, and stored in macro $\langle cmd \rangle$.

2 Implementation

```
1 \langle *package \rangle
```

2.1 Reload check and package identification

Reload check, especially if the package is not used with L^AT_EX.

```
2 \begingroup
3 \catcode44 12 % ,
4 \catcode45 12 % -
5 \catcode46 12 % .
6 \catcode58 12 % :
7 \catcode64 11 % @
8 \expandafter\let\expandafter\x\csname ver@pdfescape.sty\endcsname
9 \ifcase 0%
10 \ifx\x\relax % plain
11 \else
12 \ifx\x\empty % LaTeX
13 \else
14 1%
15 \fi
16 \fi
17 \else
18 \catcode35 6 % #
19 \catcode123 1 % {
20 \catcode125 2 % }
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{pdfescape}{The package is already loaded}%
29 \endgroup
30 \expandafter\endinput
31 \fi
32 \endgroup
```

Package identification:

```
33 \begingroup
34 \catcode35 6 % #
35 \catcode40 12 % (
36 \catcode41 12 % )
37 \catcode44 12 % ,
38 \catcode45 12 % -
39 \catcode46 12 % .
40 \catcode47 12 % /
41 \catcode58 12 % :
42 \catcode64 11 % @
43 \catcode123 1 % {
44 \catcode125 2 % }
45 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
46 \def\x#1#2#3[#4]{\endgroup
47 \immediate\write-1{Package: #3 #4}%
```

```

48     \xdef#1{#4}%
49   }%
50   \else
51     \def\x#1#2[#3]{\endgroup
52       #2[#{#3}]%
53       \ifx#1\relax
54         \xdef#1{#3}%
55       \fi
56     }%
57   \fi
58 \expandafter\x\csname ver@pdfescape.sty\endcsname
59 \ProvidesPackage{pdfescape}%
60 [2007/11/11 v1.8 Provides hex, PDF name and string conversions (H0)]

```

2.2 Catcodes

```

61 \expandafter\edef\csname PE@AtEnd\endcsname{%
62   \catcode64 \the\catcode64\relax
63 }
64 \catcode64 11 % @
65 \def\TMP@EnsureCode#1#2#3{%
66   \edef\PE@AtEnd{%
67     \PE@AtEnd
68     #1#2 \the#1#2\relax
69   }%
70   #1#2 #3\relax
71 }
72 \TMP@EnsureCode\catcode{0}{12}% ^^@
73 \TMP@EnsureCode\catcode{34}{12}% "
74 \TMP@EnsureCode\catcode{39}{12}% '
75 \TMP@EnsureCode\catcode{42}{12}% *
76 \TMP@EnsureCode\catcode{45}{12}% -
77 \TMP@EnsureCode\catcode{46}{12}% .
78 \TMP@EnsureCode\catcode{47}{12}% /
79 \TMP@EnsureCode\catcode{60}{12}% <
80 \TMP@EnsureCode\catcode{61}{12}% =
81 \TMP@EnsureCode\catcode{62}{12}% >
82 \TMP@EnsureCode\catcode{94}{7}% ^
83 \TMP@EnsureCode\catcode{96}{12}% `
84 \TMP@EnsureCode\uccode{34}{0}% "
85 \TMP@EnsureCode\uccode{48}{0}% 0
86 \TMP@EnsureCode\uccode{61}{0}% =

```

2.3 Sanitizing

`\EdefSanitize` Macro `\EdefSanitize` takes #2, entirely converts it to token with catcode 12 (other) and stores the result in macro #1.

```

87 \begingroup\expandafter\expandafter\expandafter\endgroup
88 \expandafter\ifx\csname detokenize\endcsname\relax
89   \long\def\EdefSanitize#1#2{%
90     \begingroup
91       \csname @safe@activestruel\endcsname
92       \edef#1{#2}%
93       \PE@onelevel@sanitize#1%
94     \expandafter\endgroup
95     \expandafter\def\expandafter#1\expandafter{#1}%
96   }%
97 \begingroup\expandafter\expandafter\expandafter\endgroup
98 \expandafter\ifx\csname @onelevel@sanitize\endcsname\relax
99   \def\PE@onelevel@sanitize#1{%
100     \edef#1{\expandafter\PE@strip@prefix\meaning#1}%
101   }%
102   \def\PE@strip@prefix#1>{%

```

```

103 \else
104   \let\PE@onelevel@sanitize\@onelevel@sanitize
105 \fi
106 \else
107   \long\def\EdefSanitize#1#2{%
108     \begingroup
109     \csname @safe@activestruel\endcsname
110     \edef#1{#2}%
111     \expandafter\endgroup
112     \expandafter\def\expandafter#1\expandafter{%
113       \detokenize\expandafter{#1}%
114     }%
115   }%
116   \def\PE@onelevel@sanitize#1{%
117     \edef#1{\detokenize\expandafter{#1}}%
118   }%
119 \fi

```

`\PE@sanitize` Macro `\PE@sanitize` is only defined for compatibility with version 1.4. Its use is deprecated.

```

120 \let\PE@sanitize\EdefSanitize

```

2.3.1 Space characters

`\PE@space@other`

```

121 \begingroup
122 \catcode'\ =12\relax%
123 \def\x{\endgroup\def\PE@space@other{ }}\x\relax

```

`\PE@space@space`

```

124 \def\PE@space@space{ }

```

2.3.2 Space normalization

`\PE@SanitizeSpaceOther`

```

125 \def\PE@SanitizeSpaceOther#1{%
126   \edef#1{\expandafter\PE@SpaceToOther#1 \relax}%
127 }

```

`\PE@SpaceToOther`

```

128 \def\PE@SpaceToOther#1 #2\relax{%
129   #1%
130   \ifx\#2\%
131   \else
132     \PE@space@other
133     \@ReturnAfterFi{%
134       \PE@SpaceToOther#2\relax
135     }%
136   \fi
137 }

```

`\@ReturnAfterFi`

```

138 \long\def\@ReturnAfterFi#1\fi{\fi#1}

```

2.4 \EdefUnescapeName

`\EdefUnescapeName`

```

139 \def\EdefUnescapeName#1#2{%
140   \EdefSanitize#1{#2}%
141   \PE@SanitizeSpaceOther#1%

```

```

142 \PE@UnescapeName#1%
143 \PE@onelevel@sanitize#1%
144 }

```

\PE@UnescapeName

```

145 \begingroup
146 \catcode'\$=6 % hash
147 \catcode'\#=12 % other
148 \gdef\PE@UnescapeName$1{%
149   \begingroup
150   \PE@InitUccodeHexDigit
151   \def\PE@result{}%
152   \expandafter\PE@DeName$1#\relax\relax
153   \expandafter\endgroup
154   \expandafter\def\expandafter$1\expandafter{\PE@result}%
155 }%
156 \gdef\PE@DeName$1#$2$3{%
157   \ifx\relax$2%
158     \edef\PE@result{\PE@result$1}%
159     \let\PE@next\relax
160   \else
161     \ifx\relax$3%
162       % wrong escape sequence in input
163       \edef\PE@result{\PE@result$1#}%
164       \let\PE@next\relax
165     \else
166       \uppercase{%
167         \def\PE@testA{$2}%
168         \def\PE@testB{$3}%
169       }%
170       \ifcase\ifcase\expandafter\PE@TestUcHexDigit\PE@testA
171         \ifcase\expandafter\PE@TestUcHexDigit\PE@testB
172           \z@
173         \else
174           \@ne
175         \fi
176       \else
177         \@ne
178       \fi
179       \uccode\z@="\PE@testA\PE@testB\relax
180       \uppercase{%
181         \def\PE@temp{^~@}%
182       }%
183       \uccode\z@=\z@
184       \edef\PE@result{\PE@result$1\PE@temp}%
185       \let\PE@next\PE@DeName
186     \else
187       % wrong escape sequence in input
188       \edef\PE@result{\PE@result$1#}%
189       \def\PE@next{\PE@DeName$2$3}%
190     \fi
191   \fi
192   \fi
193   \PE@next
194 }%
195 \endgroup

```

\PE@InitUccodeHexDigit

```

196 \def\PE@InitUccodeHexDigit{%
197   \uccode'a='A\relax
198   \uccode'b='B\relax
199   \uccode'c='C\relax

```

```

200 \uccode'd='D\relax
201 \uccode'e='E\relax
202 \uccode'f='F\relax
203 \uccode'A=\z@
204 \uccode'B=\z@
205 \uccode'C=\z@
206 \uccode'D=\z@
207 \uccode'E=\z@
208 \uccode'F=\z@
209 \uccode'0=\z@
210 \uccode'1=\z@
211 \uccode'2=\z@
212 \uccode'3=\z@
213 \uccode'4=\z@
214 \uccode'5=\z@
215 \uccode'6=\z@
216 \uccode'7=\z@
217 \uccode'8=\z@
218 \uccode'9=\z@
219 }

```

\PE@TestUcHexDigit

```

220 \def\PE@TestUcHexDigit#1{%
221   \ifnum'#1<48 % 0
222     \@ne
223   \else
224     \ifnum'#1>70 % F
225       \@ne
226     \else
227       \ifnum'#1>57 % 9
228         \ifnum'#1<65 % A
229           \@ne
230         \else
231           \z@
232         \fi
233       \else
234         \z@
235       \fi
236     \fi
237   \fi
238 }

```

2.5 \EdefUnescapeString

\EdefUnescapeString

```

239 \def\EdefUnescapeString#1#2{%
240   \EdefSanitize#1{#2}%
241   \PE@SanitizeSpaceOther#1%
242   \PE@NormalizeLineEnd#1%
243   \PE@UnescapeString#1%
244   \PE@onelevel@sanitize#1%
245 }

246 \begingroup
247   \uccode'\8=10 % lf
248   \uccode'\9=13 % cr
249 \def\x#1#2{\endgroup

```

\PE@NormalizeLineEnd

```

250 \def\PE@NormalizeLineEnd##1{%
251   \def\PE@result{}%
252   \expandafter\PE@@NormalizeLineEnd##1#2\relax

```

```

253     \let##1\PE@result
254 }%

\PE@@NormalizeLineEnd

255 \def\PE@@NormalizeLineEnd##1#2##2{%
256     \ifx\relax##2%
257         \edef\PE@result{\PE@result##1}%
258         \let\PE@next\relax
259     \else
260         \edef\PE@result{\PE@result##1#1}%
261         \ifx#1##2% \if
262             \let\PE@next\PE@@NormalizeLineEnd
263         \else
264             \def\PE@next{\PE@@NormalizeLineEnd##2}%
265         \fi
266     \fi
267     \PE@next
268 }%
269 }%
270 \uppercase{%
271     \x 89%
272 }

273 \begingroup
274 \catcode'\|=0 %
275 \catcode'\|=12 %

\PE@UnescapeString

276 |gdef|PE@UnescapeString#1{%
277     |begingroup
278     |def|PE@result{}%
279     |expandafter|PE@DeString#1\|relax
280     |expandafter|endgroup
281     |expandafter|def|expandafter#1|expandafter{|PE@result}%
282 }%

\PE@DeString

283 |gdef|PE@DeString#1\#2{%
284     |ifx|relax#2%
285         |edef|PE@result{|PE@result#1}%
286         |let|PE@next|relax
287     |else
288         |if n#2%
289             |uccode|z@=10 %
290         |else|if r#2%
291             |uccode|z@=13 %
292         |else|if t#2%
293             |uccode|z@=9 %
294         |else|if b#2%
295             |uccode|z@=8 %
296         |else|if f#2%
297             |uccode|z@=12 %
298         |else
299             |uccode|z@=|z@
300             |fi|fi|fi|fi|fi
301             |ifnum|uccode|z@>|z@
302             |uppercase{%
303                 |edef|PE@temp{^^@}%
304             }%
305             |edef|PE@result{|PE@result#1|PE@temp}%
306             |let|PE@next|PE@DeString
307         |else

```



```

308         |if\#2% backslash
309         |edef|PE@result{|PE@result#1}%
310         |let|PE@next|PE@CheckEndBackslash
311     |else
312         |ifnum'#2=10 % linefeed
313         |edef|PE@result{|PE@result#1}%
314         |let|PE@next|PE@DeString
315     |else
316         |ifcase|PE@TestOctDigit#2%
317         |edef|PE@result{|PE@result#1}%
318         |def|PE@next{|PE@OctI#2}%
319     |else
320         |edef|PE@result{|PE@result#1#2}%
321         |let|PE@next|PE@DeString
322     |fi
323 |fi
324 |fi
325 |fi
326 |fi
327 |PE@next
328 }%

```

\PE@CheckEndBackslash

```

329 |gdef|PE@CheckEndBackslash#1{%
330     |ifx|relax#1%
331     |else
332         |edef|PE@result{|PE@result\}%
333         |expandafter|PE@DeString|expandafter#1%
334     |fi
335 }%
336 |endgroup

```

\PE@TestOctDigit

```

337 \def\PE@TestOctDigit#1{%
338     \ifnum'#1<48 % 0
339         \@ne
340     \else
341         \ifnum'#1>55 % 7
342             \@ne
343         \else
344             \z@
345         \fi
346     \fi
347 }

```

\PE@OctI

```

348 \def\PE@OctI#1#2{%
349     \ifcase\PE@TestOctDigit#2%
350         \def\PE@next{\PE@OctII{#1#2}}%
351     \else
352         \def\PE@next{\PE@OctAll#1#2}%
353     \fi
354     \PE@next
355 }

```

\PE@OctII

```

356 \def\PE@OctII#1#2{%
357     \ifcase\PE@TestOctDigit#2%
358         \def\PE@next{\PE@OctAll{#1#2}}%
359     \else
360         \def\PE@next{\PE@OctAll{#1}#2}%

```

```

361 \fi
362 \PE@next
363 }

\PE@OctAll

364 \def\PE@OctAll#1{%
365 \uccode\z@=#1\relax
366 \uppercase{%
367 \edef\PE@result{\PE@result^^@}%
368 }%
369 \PE@DeString
370 }

```

2.6 User macros (pdfTeX analogues)

```

371 \begingroup\expandafter\expandafter\expandafter\endgroup
372 \expandafter\ifx\csname RequirePackage\endcsname\relax
373 \input pdftexcmds.sty\relax
374 \else
375 \RequirePackage{pdftexcmds}[2007/11/11]%
376 \fi

377 \begingroup\expandafter\expandafter\expandafter\endgroup
378 \expandafter\ifx\csname pdf@escapehex\endcsname\relax

\EdefEscapeHex

379 \long\def\EdefEscapeHex#1#2{%
380 \EdefSanitize#1{#2}%
381 \PE@SanitizeSpaceOther#1%
382 \PE@EscapeHex#1%
383 }%

\EdefUnescapeHex

384 \def\EdefUnescapeHex#1#2{%
385 \EdefSanitize#1{#2}%
386 \PE@UnescapeHex#1%
387 }%

\EdefEscapeName

388 \long\def\EdefEscapeName#1#2{%
389 \EdefSanitize#1{#2}%
390 \PE@SanitizeSpaceOther#1%
391 \PE@EscapeName#1%
392 }%

\EdefEscapeString

393 \long\def\EdefEscapeString#1#2{%
394 \EdefSanitize#1{#2}%
395 \PE@SanitizeSpaceOther#1%
396 \PE@EscapeString#1%
397 }%

398 \else

\PE@edefbabel Help macro that adds support for babel's shorthand characters.

399 \long\def\PE@edefbabel#1#2#3{%
400 \begingroup
401 \csname @save@activetrue\endcsname
402 \edef#1{#2{#3}}%
403 \expandafter\endgroup
404 \expandafter\def\expandafter#1\expandafter{#1}%
405 }%

```

`\EdefEscapeHex`

```
406 \long\def\EdefEscapeHex#1#2{%
407   \PE@edefbabel#1\pdf@escapehex{#2}%
408 }%
```

`\EdefUnescapeHex`

```
409 \def\EdefUnescapeHex#1#2{%
410   \PE@edefbabel#1\pdf@unescapehex{#2}%
411 }%
```

`\EdefEscapeName`

```
412 \long\def\EdefEscapeName#1#2{%
413   \PE@edefbabel#1\pdf@escapename{#2}%
414 }%
```

`\EdefEscapeString`

```
415 \long\def\EdefEscapeString#1#2{%
416   \PE@edefbabel#1\pdf@escapestring{#2}%
417 }%

418 \PE@AtEnd
419 \expandafter\endinput
420 \fi
```

2.7 Help macros

2.7.1 Characters

Special characters with catcode 12 (other) are created and stored in macros.

`\PE@hash`

```
421 \edef\PE@hash{\string#}
```

`\PE@backslash`

```
422 \begingroup
423   \escapechar=-1 %
424 \edef\x{\endgroup
425   \def\noexpand\PE@backslash{\string\\}%
426 }
427 \x
```

2.7.2 Switch for ε -T_EX

```
428 \newif\ifPE@etex
429 \begingroup\expandafter\expandafter\expandafter\endgroup
430 \expandafter\ifx\csname numexpr\endcsname\relax
431   \else
432     \PE@etextrue
433 \fi
```

2.8 Conversions

2.8.1 Conversion to hex string

`\PE@EscapeHex`

```
434 \ifPE@etex
435   \def\PE@EscapeHex#1{%
436     \edef#1{\expandafter\PE@ToHex#1\relax}%
437   }%
438 \else
439   \def\PE@EscapeHex#1{%
440     \def\PE@result{%
```

```

441     \expandafter\PE@ToHex#1\relax
442     \let#1\PE@result
443 }%
444 \fi

\PE@ToHex

445 \def\PE@ToHex#1{%
446     \ifx\relax#1%
447     \else
448         \PE@HexChar{#1}%
449         \expandafter\PE@ToHex
450     \fi
451 }%

\PE@HexChar

452 \ifPE@etex
453     \def\PE@HexChar#1{%
454         \PE@HexDigit{\numexpr\dimexpr.0625\dimexpr'#1sp\relax\relax\relax}%
455         \PE@HexDigit{%
456             \numexpr'#1-16*\dimexpr.0625\dimexpr'#1sp\relax\relax\relax
457         }%
458     }%
459 \else
460     \def\PE@HexChar#1{%
461         \dimen0='#1sp%
462         \dimen2=.0625\dimen0 %
463         \advance\dimen0-16\dimen2 %
464         \edef\PE@result{%
465             \PE@result
466             \PE@HexDigit{\dimen2 }%
467             \PE@HexDigit{\dimen0 }%
468         }%
469     }%
470 \fi

\PE@HexDigit

471 \def\PE@HexDigit#1{%
472     \expandafter\string
473     \ifcase#1%
474         0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or 9\or
475         A\or B\or C\or D\or E\or F%
476     \fi
477 }

```

2.8.2 Character code to octal number

```

\PE@OctChar

478 \ifPE@etex
479     \def\PE@OctChar#1{%
480         \expandafter\PE@@OctChar
481         \the\numexpr\dimexpr.015625\dimexpr'#1sp\relax\relax
482         \expandafter\relax
483         \expandafter\relax
484         \the\numexpr\dimexpr.125\dimexpr'#1sp\relax\relax\relax
485         \relax
486         #1%
487     }%
488     \def\PE@@OctChar#1\relax#2\relax#3{%
489         \PE@backslash
490         #1%
491         \the\numexpr#2-8*#1\relax

```

```

492 \the\numexpr\dimexpr'#3sp\relax-8*#2\relax
493 }%
494 \else
495 \def\PE@OctChar#1{%
496 \dimen0='#1sp%
497 \dimen2=.125\dimen0 %
498 \dimen4=.125\dimen2 %
499 \advance\dimen0-8\dimen2 %
500 \advance\dimen2-8\dimen4 %
501 \edef\PE@result{%
502 \PE@result
503 \PE@backslash
504 \number\dimen4 %
505 \number\dimen2 %
506 \number\dimen0 %
507 }%
508 }%
509 \fi

```

2.8.3 Unpack hex string

\PE@UnescapeHex

```

510 \def\PE@UnescapeHex#1{%
511 \begingroup
512 \PE@InitUccodeHexDigit
513 \def\PE@result{%
514 \expandafter\PE@DeHex#1\relax\relax
515 \expandafter\endgroup
516 \expandafter\def\expandafter#1\expandafter{\PE@result}%
517 }

```

\PE@DeHex

```

518 \def\PE@DeHex#1#2{%
519 \ifx#2\relax
520 \ifx#1\relax
521 \let\PE@next\relax
522 \else
523 \uppercase{%
524 \def\PE@testA{#1}%
525 }%
526 \ifcase\expandafter\PE@TestUcHexDigit\PE@testA
527 \def\PE@next{%
528 \PE@DeHex#10\relax\relax
529 }%
530 \else
531 \let\PE@next\relax
532 \fi
533 \fi
534 \else
535 \uppercase{%
536 \def\PE@testA{#1}%
537 \def\PE@testB{#2}%
538 }%
539 \ifcase\expandafter\PE@TestUcHexDigit\PE@testA
540 \ifcase\expandafter\PE@TestUcHexDigit\PE@testB
541 \uccode\z@="\PE@testA\PE@testB\relax
542 \ifnum\uccode\z@=32 %
543 \let\PE@temp\PE@space@space
544 \else
545 \uppercase{%
546 \def\PE@temp{^^}%
547 }%

```



```

604     \fi
605     \expandafter\PE@EscapeNameTokens
606   \fi
607 }%
608 \def\PE@EscapeNameHashChar#1#2{%
609   \PE@EscapeNameAdd{\PE@hash\string#1\string#2}%
610 }%

```

\PE@EscapeNameAdd

```

611 \ifPE@etex
612   \def\PE@EscapeNameAdd#1{#1}%
613 \else
614   \def\PE@EscapeNameAdd#1{%
615     \edef\PE@result{%
616       \PE@result
617       #1%
618     }%
619   }%
620 \fi

```

2.8.5 Conversion to PDF string

\PE@EscapeString

```

621 \ifPE@etex
622   \def\PE@EscapeString#1{%
623     \edef#1{\expandafter\PE@EscapeStringTokens#1\relax}%
624   }%
625 \else
626   \def\PE@EscapeString#1{%
627     \begingroup
628     \def\PE@result{}%
629     \expandafter\PE@EscapeStringTokens#1\relax
630     \expandafter\endgroup
631     \expandafter\def\expandafter#1\expandafter{\PE@result}%
632   }%
633 \fi

```

\PE@EscapeStringTokens

```

634 \def\PE@EscapeStringTokens#1{%
635   \ifx\relax#1%
636   \else
637     \ifnum'#1<33 %
638       \PE@OctChar#1%
639     \else
640       \ifnum'#1>126 %
641         \PE@OctChar#1%
642       \else \ifnum'#1=40 \PE@EscapeStringAdd{\string\}% (
643         \else\ifnum'#1=41 \PE@EscapeStringAdd{\string\)}% )
644         \else\ifnum'#1=92 \PE@EscapeStringAdd{\string\\}% \
645         \else
646           \PE@EscapeStringAdd{#1}%
647         \fi\fi\fi
648       \fi
649     \fi
650     \expandafter\PE@EscapeStringTokens
651   \fi
652 }%

```

\PE@EscapeStringAdd

```

653 \ifPE@etex
654   \def\PE@EscapeStringAdd#1{#1}%

```

```

655 \else
656   \def\PE@EscapeStringAdd#1{%
657     \edef\PE@result{%
658       \PE@result
659       #1%
660     }%
661   }%
662 \fi

663 \PE@AtEnd
664 \</package>

```

3 Test

3.1 Catcode checks for loading

```

665 \<test1>

666 \catcode'\{=1 %
667 \catcode'\}=2 %
668 \catcode'\#=6 %
669 \catcode'\@=11 %
670 \expandafter\ifx\csname count@\endcsname\relax
671   \countdef\count@=255 %
672 \fi
673 \expandafter\ifx\csname @gobble\endcsname\relax
674   \long\def\@gobble#1{%
675 \fi
676 \expandafter\ifx\csname @firstofone\endcsname\relax
677   \long\def\@firstofone#1{#1}%
678 \fi
679 \expandafter\ifx\csname loop\endcsname\relax
680   \expandafter\@firstofone
681 \else
682   \expandafter\@gobble
683 \fi
684 {%
685   \def\loop#1\repeat{%
686     \def\body{#1}%
687     \iterate
688   }%
689   \def\iterate{%
690     \body
691     \let\next\iterate
692   \else
693     \let\next\relax
694   \fi
695   \next
696 }%
697 \let\repeat=\fi
698 }%
699 \def\RestoreCatcodes{}
700 \count@=0 %
701 \loop
702   \edef\RestoreCatcodes{%
703     \RestoreCatcodes
704     \catcode\the\count@=\the\catcode\count@\relax
705   }%
706 \ifnum\count@<255 %
707   \advance\count@ 1 %
708 \repeat
709

```



```

710 \def\RangeCatcodeInvalid#1#2{%
711   \count@=#1\relax
712   \loop
713     \catcode\count@=15 %
714   \ifnum\count@<#2\relax
715     \advance\count@ 1 %
716   \repeat
717 }
718 \def\Test{%
719   \RangeCatcodeInvalid{0}{47}%
720   \RangeCatcodeInvalid{58}{64}%
721   \RangeCatcodeInvalid{91}{96}%
722   \RangeCatcodeInvalid{123}{255}%
723   \catcode'\@=12 %
724   \catcode'\=0 %
725   \catcode'\{=1 %
726   \catcode'\}=2 %
727   \catcode'\#=6 %
728   \catcode'\[=12 %
729   \catcode'\]=12 %
730   \catcode'\%=14 %
731   \catcode'\_ =10 %
732   \catcode13=5 %
733   \input pdfescape.sty\relax
734   \RestoreCatcodes
735 }
736 \Test
737 \csname @@end\endcsname
738 \end
739 </test1>

```

3.2 Macro tests

```

740 <*test2 | test3 | test4 | test5>
741 \NeedsTeXFormat{LaTeX2e}
742 \makeatletter

```

3.3 Test with \pdfescape... commands

```

743 <*test2>
744 \ProvidesFile{pdfescape-test2.tex}%
745   [2007/11/11 v1.8 Test with \string\pdfescape... commands]%
746 </test2>

```

3.4 Test without \pdfescape..., with ε -TeX

```

747 <*test3>
748 \ProvidesFile{pdfescape-test3.tex}%
749   [2007/11/11 v1.8 Test without \string\pdfescape..., with e-TeX]%
750 </test3>

```

3.5 Test without \pdfescape... and ε -TeX

```

751 <*test4>
752 \ProvidesFile{pdfescape-test4.tex}%
753   [2007/11/11 v1.8 Test without \string\pdfescape... and e-TeX]%
754 </test4>

```

3.6 Test with LuaTeX

```

755 <*test5>
756 \ProvidesFile{pdfescape-test5.tex}%
757   [2007/11/11 v1.8 Test with LuaTeX]%
758 </test5>

```

3.7 Check/ensure test preconditions

3.7.1 Check `\pdfescape...`

```
759 <*test2>
760 \@ifundefined{pdfescapehex}{%
761   \PackageError{pdfescape-test2}{%
762     Missing \string\pdfescape... commands%
763   }{Test aborted.}%
764   \stop
765 }{}
766 </test2>

767 <*test3 | test4>
768 \let\pdfescapehex\undefined
769 \let\pdfunescapehex\undefined
770 \let\pdfescapename\undefined
771 \let\pdfescapestring\undefined
772 </test3 | test4>
```

3.7.2 Check ε - \TeX

```
773 <*test3>
774 \@ifundefined{numexpr}{%
775   \PackageError{pdfescape-test3}{%
776     Missing \eTeX
777   }{Test aborted.}%
778   \stop
779 }{}
780 </test3>
```

Package `qstest` uses ε - \TeX , thus ε - \TeX 's features can only be disabled later during loading of package `pdfescape`.

3.7.3 Check \LuaTeX

```
781 <*test5>
782 \@ifundefined{directlua}{%
783   \PackageError{pdfescape-test5}{%
784     Missing LuaTeX%
785   }{Test aborted.}%
786   \stop
787 }{}
788 </test5>
```

3.8 Common part

The files for testing uses the framework, provided by the new package `qstest` of David Kastrup.

```
789 \RequirePackage{qstest}
790 \IncludeTests{*}
791 \LogTests{log}{*}{*}
792
793 \newcommand*{\ExpectVar}[2]{%
794   \ifx#1#2%
795   \else
796     \begingroup
797       \@onelevel@sanitize#1%
798       \@onelevel@sanitize#2%
799       \typeout{[#1] <> [#2]}% hash-ok
800     \endgroup
801   \fi
802   \Expect*{\ifx#1#2true\else false\fi}{true}%
803 }
804
805 \makeatletter
806 \begingroup
```

```

807 \gdef\AllBytes{%
808 \count@=0 %
809 \catcode0=12 %
810 \@whilenum\count@<256 \do{%
811 \lccode0=\count@
812 \ifnum\count@=32 %
813 \xdef\AllBytes{\AllBytes\space}%
814 \else
815 \lowercase{%
816 \xdef\AllBytes{\AllBytes^^@}%
817 }%
818 \fi
819 \advance\count@ by 1 %
820 }%
821 \endgroup
822 \newcommand*{\AllBytesHex}{%
823 000102030405060708090A0B0C0D0E0F%
824 101112131415161718191A1B1C1D1E1F%
825 202122232425262728292A2B2C2D2E2F%
826 303132333435363738393A3B3C3D3E3F%
827 404142434445464748494A4B4C4D4E4F%
828 505152535455565758595A5B5C5D5E5F%
829 606162636465666768696A6B6C6D6E6F%
830 707172737475767778797A7B7C7D7E7F%
831 808182838485868788898A8B8C8D8E8F%
832 909192939495969798999A9B9C9D9E9F%
833 A0A1A2A3A4A5A6A7A8A9AABACADAEEAF%
834 B0B1B2B3B4B5B6B7B8B9BABBCBDBEBF%
835 C0C1C2C3C4C5C6C7C8C9CACBCDCECF%
836 D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
837 E0E1E2E3E4E5E6E7E8E9EAECEDEEEF%
838 F0F1F2F3F4F5F6F7F8F9FABFCFDFEFF%
839 }
840 \@onelevel@sanitize\AllBytesHex
841 \expandafter\lowercase\expandafter{%
842 \expandafter\newcommand\expandafter*\expandafter\AllBytesHexLC
843 \expandafter{\AllBytesHex}%
844 }
845 \newcommand*{\AllBytesName}{%
846 \begingroup
847 \catcode'\# =12 %
848 \xdef\AllBytesName{%
849 #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
850 #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
851 #20!"#23$#25&'#28#29*+,-.#2F%
852 0123456789:;#3C=#3E?%
853 @ABCDEFGHIJKLMNO%
854 PQRSTUVWXYZ#5B\@backslashchar#5D^_%
855 'abcdefghijklmnopqrstuvwxyz%
856 pqrstuvwxyz7B|#7D\string~#7F%
857 #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
858 #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
859 #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
860 #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
861 #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
862 #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
863 #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
864 #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
865 }%
866 \endgroup
867 \@onelevel@sanitize\AllBytesName
868

```

```

869 \newcommand*{\AllBytesString}{%
870 \begingroup
871   \def\|{|}%
872   \edef\%{\@percentchar}%
873   \catcode'\|=0 %
874   \catcode'\#=12 %
875   \catcode'\~=12 %
876   \catcode'\=12 %
877   |xdef|AllBytesString{%
878     \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
879     \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
880     \040!"#$%&'(\)*+,-./%
881     0123456789:;<=>?%
882     @ABCDEFGHIJKLMNO%
883     PQRSTUVWXYZ[\]^_%
884     'abcdefghijklmno%
885     pqrstuvwxyz{|}~\177%
886     \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
887     \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
888     \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
889     \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
890     \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
891     \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
892     \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
893     \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
894   }%
895 \endgroup
896 \@onelevel@sanitize\AllBytesString
897
898 <*test4>
899 \let\org@detokenize\detokenize
900 \let\detokenize\@undefined
901 \let\org@numexpr\numexpr
902 \let\numexpr\@undefined
903 </test4>
904 \RequirePackage{pdfescape}
905 <*test4>
906 \let\detokenize\org@detokenize
907 \let\numexpr\org@numexpr
908 </test4>
909
910 \begin{qstest}{all-hex}{\AllBytes, escapehex}
911   \EdefEscapeHex\x{\AllBytes}%
912   \Expect*{\x}*{\AllBytesHex}%
913   \ExpectVar\x\AllBytesHex
914 \end{qstest}
915
916 \begin{qstest}{all-unhex}{\AllBytesHex, unescapehex}
917   \EdefUnescapeHex\x{\AllBytesHex}%
918   \Expect*{\x}*{\AllBytes}%
919   \ExpectVar\x\AllBytes
920 \end{qstest}
921
922 \begin{qstest}{all-unhex-lc}{\AllBytesHexLC, unescapehex, lowercase}
923   \EdefUnescapeHex\x{\AllBytesHexLC}%
924   \Expect*{\x}*{\AllBytes}%
925   \ExpectVar\x\AllBytes
926 \end{qstest}
927
928 \begin{qstest}{unhex-incomplete}{unescapehex, incomplete}
929   \EdefUnescapeHex\x{4}%
930   \Expect*{\x}{@}%

```

```

931 \end{qstest}
932
933 \begin{qstest}{unhex-space}{unescapehex, space}
934   \EdefUnescapeHex\x{20}%
935   \Expect*{\x}{ }%
936   \ExpectVar\x\space
937 \end{qstest}
938
939 \begin{qstest}{unhex-spaces}{unescapehex, spaces}
940   \EdefUnescapeHex\x{204020204120}%
941   \def\y#1{%
942     \edef\z{#1\string @#1#1\string A#1}%
943   }\y{ }%
944   \Expect*{\x}*{\z}%
945   \ExpectVar\x\z
946 \end{qstest}
947
948 \begin{qstest}{unhex-hash}{unescapehex, hash}
949   \catcode'\#=12 %
950   \EdefUnescapeHex\x{#20}%
951   \ExpectVar\x\space
952 \end{qstest}
953
954 \begin{qstest}{unhex-invalid}{unescapehex, invalid}
955   \def\test#1#2{%
956     \EdefUnescapeHex\x{#1}%
957     \edef\y{#2}%
958     \@onelevel@sanitize\y
959     \ExpectVar\x\y
960   }%
961   <*test2>
962   \edef\x{\pdfunescapehex{4X}}%
963   \edef\y{\string @}%
964   \ifx\x\y
965   \else
966     \def~{\space}%
967     \typeout{*****}%
968     \typeout{* Your pdfTeX contains bug 777.~~~~*}%
969     \typeout{* This test is redefined as dummy, *}%
970     \typeout{* because it triggers the bug.~~~~*}%
971     \typeout{*****}%
972     \def\test#1#2{%
973       \fi
974     </test2>
975     \test{X}{}%
976     \test{XY}{}%
977     \test{XYZ}{}%
978     \test{A}{~^a0}%
979     \test{AX}{^^a0}%
980     \test{XA}{^^a0}%
981     \test{XXAXX}{^^a0}%
982 \end{qstest}
983
984 \begin{qstest}{all-name}{\AllBytes, escapename}
985   \EdefEscapeName\x{\AllBytes}%
986   \Expect*{\x}*{\AllBytesName}%
987   \ExpectVar\x\AllBytesName
988 \end{qstest}
989
990 \begin{qstest}{all-string}{\AllBytes, escapestring}
991   \EdefEscapeString\x{\AllBytes}%
992   \Expect*{\x}*{\AllBytesString}%

```

```

993 \ExpectVar\x\AllBytesString
994 \end{qstest}
995
996 \begin{qstest}{uchexdigit}{unescape, uppercase hex digit}
997 \catcode'\@=11 %
998 \catcode0=12 %
999 \def\test#1#2{%
1000 \uccode0=#1\relax
1001 \uppercase{%
1002 \def\x{^^@}%
1003 }%
1004 \Expect*{%
1005 \ifcase\expandafter\PE@TestUcHexDigit\x
1006 true%
1007 \else
1008 false%
1009 \fi
1010 }{#2}%
1011 }%
1012 \def\range#1#2#3{%
1013 \count0=#1\relax
1014 \loop
1015 \ifnum\count0<#2\relax
1016 \test{\count0}{#3}%
1017 \advance\count0 by 1 %
1018 \repeat
1019 }%
1020 \range{0}{47}{false}%
1021 \range{48}{57}{true}%
1022 \range{58}{64}{false}%
1023 \range{65}{70}{true}%
1024 \range{71}{255}{false}%
1025 \end{qstest}
1026
1027 \begin{qstest}{unescape}{unescape}
1028 \def\test#1#2{%
1029 \EdefUnescapeName\x{#1}%
1030 \edef\y{#2}%
1031 \@onelevel@sanitize\y
1032 \ExpectVar\x\y
1033 }%
1034 \catcode'\#=12 %
1035 \catcode0=12 %
1036 \test{}{}%
1037 \test{x}{x}%
1038 \test{xy}{xy}%
1039 \test{#}{#}%
1040 \test{##}{##}%
1041 \test{###}{###}%
1042 \test{####}{####}%
1043 \test{#x}{#x}%
1044 \test{#xy}{#xy}%
1045 \test{#1}{#1}%
1046 \test{#40}{@}%
1047 \test{#400}{@}%
1048 \test{#4x0}{#4x0}%
1049 \test{#ab}{^^ab}%
1050 \test{#00}{^^@}%
1051 \test{x#40y#40z}{x@y@z}%
1052 \test{#40#40#40#40}{@@@}%
1053 \test{a#x}{a#x}%
1054 \test{a#xy}{a#xy}%

```

```

1055 \test{a#1}{a#1}%
1056 \test{a#40}{a@}%
1057 \test{a#400}{a@0}%
1058 \test{#20}{ }%
1059 \test{a#20}{a }%
1060 \test{a#20b}{a b}%
1061 \test{a#20#20#20b}{a \space\space b}%
1062 \end{qstest}
1063
1064 \begin{qstest}{unescapestring}{unescapestring}
1065 \def\test#1#2{%
1066 \EdefUnescapeString\x{#1}%
1067 \edef\y{#2}%
1068 \@onelevel@sanitize\y
1069 \ExpectVar\x\y
1070 }%
1071 \catcode0=12 %
1072 \def\DefChar#1#2{%
1073 \begingroup
1074 \uccode0=#2\relax
1075 \uppercase{\endgroup
1076 \def#1{^~@}%
1077 }%
1078 }%
1079 \DefChar\nul{0}%
1080 \DefChar\one{1}%
1081 \DefChar\bel{8}%
1082 \DefChar\tab{9}%
1083 \DefChar\lf{10}%
1084 \DefChar\ff{12}%
1085 \DefChar\cr{13}%
1086 \DefChar\{92}%
1087 \test{}{}%
1088 \test{a}{a}%
1089 \test{\}{}%
1090 \test{\\}{}%
1091 \test{\\y}{\y}%
1092 \test{\\000}{\nul}%
1093 \test{\\b}{\bel}%
1094 \test{\\t}{\tab}%
1095 \test{\\n}{\lf}%
1096 \test{\\f}{\ff}%
1097 \test{\\r}{\cr}%
1098 \test{\\{(){}%
1099 \test{\\}){)%}%
1100 \test{\\040}{ }%
1101 \test{\\100}{@}%
1102 \test{\\40}{ }%
1103 \test{\\1}{\one}%
1104 \test{\\01}{\one}%
1105 \test{\\001}{\one}%
1106 \test{\\18}{\one8}%
1107 \test{\\018}{\one8}%
1108 \test{\\0018}{\one8}%
1109 \test{x\\}{x}%
1110 \test{x\\\\}{x\\}%
1111 \test{x\\\\y}{x\y}%
1112 \test{x\\000}{x\nul}%
1113 \test{x\\b}{x\bel}%
1114 \test{x\\t}{x\tab}%
1115 \test{x\\n}{x\lf}%
1116 \test{x\\f}{x\ff}%

```

```

1117 \test{x\\r}{x\cr}%
1118 \test{x\\(){x(}%
1119 \test{x\\)}{x})}%
1120 \test{x\\040}{x }%
1121 \test{x\\100}{x@}%
1122 \test{x\\40}{x }%
1123 \test{x\\1}{x\one}%
1124 \test{x\\01}{x\one}%
1125 \test{x\\001}{x\one}%
1126 \test{x\\18}{x\one8}%
1127 \test{x\\018}{x\one8}%
1128 \test{x\\0018}{x\one8}%
1129 \test{\\b\\t\\n\\f\\r\\(\\)\\\\\\\\000\\040}{%
1130 \bel\tab\lf\ff\cr()\\nul\space
1131 }%
1132 \test{\\lf}{}%
1133 \test{x\\lf}{x}%
1134 \test{\cr}{\lf}%
1135 \test{\cr\lf}{\lf}%
1136 \test{\lf}{\lf}%
1137 \test{\lf\cr}{\lf\lf}%
1138 \test{x\cr}{x\lf}%
1139 \test{x\cr\lf}{x\lf}%
1140 \test{x\lf}{x\lf}%
1141 \test{x\lf\cr}{x\lf\lf}%
1142 \test{x\\cr\lf y\cr}{xy\lf}%
1143 \end{qstest}
1144 \stop
1145 </test2 | test3 | test4 | test5>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

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

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

4.2 Bundle installation

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

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

¹<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-`TEX`:

```
tex pdfescape.dtx
```

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

```
pdfescape.sty      → tex/generic/oberdiek/pdfescape.sty
pdfescape.pdf      → doc/latex/oberdiek/pdfescape.pdf
test/pdfescape-test1.tex → doc/latex/oberdiek/test/pdfescape-test1.tex
test/pdfescape-test2.tex → doc/latex/oberdiek/test/pdfescape-test2.tex
test/pdfescape-test3.tex → doc/latex/oberdiek/test/pdfescape-test3.tex
test/pdfescape-test4.tex → doc/latex/oberdiek/test/pdfescape-test4.tex
test/pdfescape-test5.tex → doc/latex/oberdiek/test/pdfescape-test5.tex
pdfescape.dtx      → source/latex/oberdiek/pdfescape.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`, mik`TEX`, ...) relies on file name databases, you must refresh these. For example, te`TEX` users run `texhash` or `mktextlsr`.

4.5 Some details for the interested

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

```
pdftk pdfescape.pdf unpack_files output .
```

Unpacking with L^AT_EX. The `.dtx` chooses its action depending on the format:

plain-`TEX`: Run `docstrip` and extract the files.

L^AT_EX: Generate the documentation.

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

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

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

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

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

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

```
pdflatex pdfescape.dtx
makeindex -s gind.ist pdfescape.idx
pdflatex pdfescape.dtx
makeindex -s gind.ist pdfescape.idx
pdflatex pdfescape.dtx
```

5 History

[2007/02/21 v1.0]

- First version.

[2007/02/25 v1.1]

- Test files added.
- `\EdefUnescapeHex` supports lowercase letters.
- Fix: `\EdefEscapeName{^^@}`
- Fix: `\EdefEscapeName{\string#}`
- Fix for `\EdefUnescapeHex` in case of incomplete hex string.
- Fix: `\EdefUnescapeHex` generates space tokens with catcode 10 (space) in all cases.
- Fix: `\EdefEscapeHex` and `\EdefEscapeName` now generate tokens with catcode 12 (other) only.

[2007/03/20 v1.2]

- Fix: Wrong year in `\ProvidesPackage`.

[2007/04/11 v1.3]

- Line ends sanitized.

[2007/04/21 v1.4]

- `\EdefUnescapeName` and `\EdefUnescapeString` added.

[2007/08/27 v1.5]

- `\EdefSanitize` added (replaces `\PE@sanitize`).

[2007/09/09 v1.6]

- Fix in catcode setup.

[2007/10/27 v1.7]

- More efficient `\EdefSanitize`.

- Use of package `pdftexcmds` for L^AT_EX support.

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.

Symbols			
<code>\#</code>	147, 283, 308, 668, 727, 847, 874, 949, 1034	<code>\AllBytesHex</code>	822, 840, 843, 912, 913, 916, 917
<code>\\$</code>	146	<code>\AllBytesHexLC</code>	842, 922, 923
<code>\%</code>	730, 872	<code>\AllBytesName</code>	845, 848, 867, 986, 987
<code>\(</code>	642, 880	<code>\AllBytesString</code>	869, 896, 992, 993
<code>\)</code>	643, 880	B	
<code>\@</code>	669, 723, 997	<code>\begin</code>	910, 916, 922, 928, 933, 939, 948, 954, 984, 990, 996, 1027, 1064
<code>\@ReturnAfterFi</code>	133, <u>138</u>	<code>\bel</code>	1081, 1093, 1113, 1130
<code>\@backslashchar</code>	854	<code>\body</code>	686, 690
<code>\@firstofone</code>	677, 680	C	
<code>\@gobble</code>	674, 682	<code>\catcode</code>	3, 4, 5, 6, 7, 18, 19, 20, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 62, 64, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 122, 146, 147, 274, 275, 666, 667, 668, 669, 704, 713, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 809, 847, 873, 874, 875, 876, 949, 997, 998, 1034, 1035, 1071
<code>\@ifundefined</code>	760, 774, 782	<code>\count</code>	1013, 1015, 1016, 1017
<code>\@ne</code>	174, 177, 222, 225, 229, 339, 342	<code>\count@</code>	671, 700, 704, 706, 707, 711, 713, 714, 715, 808, 810, 811, 812, 819
<code>\@onelevel@sanitize</code>	104, 797, 798, 840, 867, 896, 958, 1031, 1068	<code>\countdef</code>	671
<code>\@percentchar</code>	872	<code>\cr</code>	1085, 1097, 1117, 1130, 1134, 1135, 1137, 1138, 1139, 1141, 1142
<code>\@undefined</code>	768, 769, 770, 771, 900, 902	<code>\csname</code>	8, 21, 45, 58, 61, 88, 91, 98, 109, 372, 378, 401, 430, 670, 673, 676, 679, 737
<code>\@whilenum</code>	810	D	
<code>\[</code>	728	<code>\DefChar</code>	1072, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086
<code>\[</code>	130, 275, 425, 644, 724, 876, 883, 1086, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1132, 1133, 1142	<code>\detokenize</code>	113, 117, 899, 900, 906
<code>\{</code>	666, 725	<code>\dimen</code>	461, 462, 463, 466, 467, 496, 497, 498, 499, 500, 504, 505, 506
<code>\}</code>	332, 667, 726	<code>\dimexpr</code>	454, 456, 481, 484, 492
<code>\]</code>	729	<code>\do</code>	810
<code>\ </code>	274, 279, 871, 873	E	
<code>\~</code>	875	<code>\EdefEscapeHex</code>	2, <u>379</u> , <u>406</u> , 911
Numbers		<code>\EdefEscapeName</code>	388, <u>412</u> , 985
<code>\0</code>	878, 879, 880	<code>\EdefEscapeString</code>	<u>393</u> , <u>415</u> , 991
<code>\1</code>	885	<code>\EdefSanitize</code>	3, <u>87</u> , 120, 140, 240, 380, 385, 389, 394
<code>\2</code>	886, 887, 888, 889	<code>\EdefUnescapeHex</code>	<u>384</u> , <u>409</u> , 917, 923, 929, 934, 940, 950, 956
<code>\3</code>	890, 891, 892, 893	<code>\EdefUnescapeName</code>	2, <u>139</u> , 1029
<code>\8</code>	247	<code>\EdefUnescapeString</code>	2, <u>239</u> , 1066
<code>\9</code>	248		
<code>_</code>	122, 644, 731		
A			
<code>\advance</code>	463, 499, 500, 707, 715, 819, 1017		
<code>\AllBytes</code>	807, 813, 816, 910, 911, 918, 919, 924, 925, 984, 985, 990, 991		

<code>\empty</code>	12		O	
<code>\end</code> 738, 914, 920, 926, 931, 937, 946, 952, 982, 988, 994, 1025, 1062, 1143			<code>\one</code>	1080, 1103, 1104, 1105, 1106, 1107, 1108, 1123, 1124, 1125, 1126, 1127, 1128
<code>\endcsname</code>	8, 21, 45, 58, 61, 88, 91, 98, 109, 372, 378, 401, 430, 670, 673, 676, 679, 737		<code>\org@detokenize</code>	899, 906
<code>\endinput</code>	30, 419		<code>\org@numexpr</code>	901, 907
<code>\escapechar</code>	423		P	
<code>\eTeX</code>	776		<code>\PackageError</code>	761, 775, 783
<code>\Expect</code>	802, 912, 918, 924, 930, 935, 944, 986, 992, 1004		<code>\PackageInfo</code>	26
<code>\ExpectVar</code> 793, 913, 919, 925, 936, 945, 951, 959, 987, 993, 1032, 1069			<code>\pdf@escapehex</code>	407
F			<code>\pdf@escapename</code>	413
<code>\ff</code>	1084, 1096, 1116, 1130		<code>\pdf@escapestring</code>	416
G			<code>\pdf@unescapehex</code>	410
<code>\gdef</code>	148, 156, 807		<code>\pdfescape</code>	745, 749, 753, 762
I			<code>\pdfescapehex</code>	768
<code>\ifcase</code>	9, 170, 171, 349, 357, 473, 526, 539, 540, 579, 1005		<code>\pdfescapename</code>	770
<code>\ifnum</code>	221, 224, 227, 228, 338, 341, 542, 578, 586, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 637, 640, 642, 643, 644, 706, 714, 812, 1015		<code>\pdfescapestring</code>	771
<code>\ifPE@etex</code>	428, 434, 452, 478, 564, 611, 621, 653		<code>\pdfunescapehex</code>	769, 962
<code>\ifx</code> .. 10, 12, 21, 45, 53, 88, 98, 130, 157, 161, 256, 261, 372, 378, 430, 446, 519, 520, 576, 635, 670, 673, 676, 679, 794, 802, 964			<code>\PE@@NormalizeLineEnd</code>	252, 255
<code>\immediate</code>	23, 47		<code>\PE@@OctChar</code>	480, 488
<code>\IncludeTests</code>	790		<code>\PE@AtEnd</code>	66, 67, 418, 663
<code>\input</code>	373, 733		<code>\PE@backslash</code>	422, 489, 503
<code>\iterate</code>	687, 689, 691		<code>\PE@CheckEndBackslash</code>	329
L			<code>\PE@DeHex</code>	514, 518
<code>\lccode</code>	811		<code>\PE@DeName</code>	152, 156, 185, 189
<code>\lf</code>	1083, 1095, 1115, 1130, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142		<code>\PE@DeString</code>	283, 369
<code>\LogTests</code>	791		<code>\PE@edefbabel</code> .. 399, 407, 410, 413, 416	
<code>\loop</code>	685, 701, 712, 1014		<code>\PE@EscapeHex</code>	382, 434
<code>\lowercase</code>	815, 841		<code>\PE@EscapeName</code>	391, 564
M			<code>\PE@EscapeNameAdd</code>	582, 587, 601, 609, 611
<code>\makeatletter</code>	742, 805		<code>\PE@EscapeNameHashChar</code>	589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 608
<code>\meaning</code>	100		<code>\PE@EscapeNameTokens</code> ..	566, 571, 575
N			<code>\PE@EscapeString</code>	396, 621
<code>\NeedsTeXFormat</code>	741		<code>\PE@EscapeStringAdd</code>	642, 643, 644, 646, 653
<code>\newcommand</code> ... 793, 822, 842, 845, 869			<code>\PE@EscapeStringTokens</code>	623, 629, 634
<code>\newif</code>	428		<code>\PE@etextrue</code>	432
<code>\next</code>	691, 693, 695		<code>\PE@hash</code>	421, 582, 587, 609
<code>\nul</code>	1079, 1092, 1112, 1130		<code>\PE@HexChar</code>	448, 452, 583, 588
<code>\number</code>	504, 505, 506		<code>\PE@HexDigit</code> .. 454, 455, 466, 467, 471	
<code>\numexpr</code>	454, 456, 481, 484, 491, 492, 901, 902, 907		<code>\PE@InitUccodeHexDigit</code>	150, 196, 512
			<code>\PE@next</code>	159, 164, 185, 189, 193, 258, 262, 264, 267, 350, 352, 354, 358, 360, 362, 521, 527, 531, 550, 553, 559, 562
			<code>\PE@NormalizeLineEnd</code>	242, 250
			<code>\PE@OctAll</code>	352, 358, 360, 364
			<code>\PE@OctChar</code>	478, 638, 641
			<code>\PE@OctI</code>	348
			<code>\PE@OctII</code>	350, 356
			<code>\PE@onelevel@sanitize</code>	93, 99, 104, 116, 143, 244
			<code>\PE@result</code>	151, 154, 158, 163, 184, 188, 251, 253, 257, 260, 367, 440, 442, 464, 465, 501, 502, 513, 516, 549, 570, 572, 615, 616, 628, 631, 657, 658
			<code>\PE@sanitize</code>	120

\PE@SanitizeSpaceOther	1105, 1106, 1107, 1108, 1109,
..... 125 , 141 , 241 , 381 , 390 , 395	1110, 1111, 1112, 1113, 1114,
\PE@space@other	121 , 132 1115, 1116, 1117, 1118, 1119,
\PE@space@space	124 , 543 1120, 1121, 1122, 1123, 1124,
\PE@SpaceToOther	126 , 128 1125, 1126, 1127, 1128, 1129,
\PE@strip@prefix	100 , 102 1132, 1133, 1134, 1135, 1136,
\PE@temp	181 , 184 , 543 , 546 , 549 1137, 1138, 1139, 1140, 1141, 1142
\PE@testA	167 , \the 62 , 68 , 481 , 484 , 491 , 492 , 704
..... 170 , 179 , 524 , 526 , 536 , 539 , 541	\TMP@EnsureCode 65 , 72 , 73 , 74 , 75 , 76 ,
\PE@testB .	168 , 171 , 179 , 537 , 540 , 541 77 , 78 , 79 , 80 , 81 , 82 , 83 , 84 , 85 , 86
\PE@TestOctDigit	337 , 349 , 357 \typeout .. 799 , 967 , 968 , 969 , 970 , 971
\PE@TestUcHexDigit	
..... 170 , 171 , 220 , 526 , 539 , 540 , 1005	
\PE@ToHex	436 , 441 , 445 U
\PE@UnescapeHex	386 , 510 \uccode
\PE@UnescapeName	142 , 145 84 , 85 ,
\PE@UnescapeString	243 , 276 86 , 179 , 183 , 197 , 198 , 199 , 200 ,
\ProvidesFile	744 , 748 , 752 , 756 201 , 202 , 203 , 204 , 205 , 206 ,
\ProvidesPackage	59 207 , 208 , 209 , 210 , 211 , 212 ,
	213 , 214 , 215 , 216 , 217 , 218 ,
	247 , 248 , 365 , 541 , 542 , 1000 , 1074
	\uppercase
R	166 , 180 ,
\range 1012 , 1020 , 1021 , 1022 , 1023 , 1024	270 , 366 , 523 , 535 , 545 , 1001 , 1075
\RangeCatcodeInvalid	
..... 710 , 719 , 720 , 721 , 722	W
\repeat	685 , 697 , 708 , 716 , 1018 \write
\RequirePackage	375 , 789 , 904 23 , 47
\RestoreCatcodes ..	699 , 702 , 703 , 734
S	X
\space ..	8 , 10 , 12 , 22 , 26 , 28 ,
\stop	46 , 51 , 58 , 123 , 249 , 271 , 424 ,
	427 , 911 , 912 , 913 , 917 , 918 ,
	919 , 923 , 924 , 925 , 929 , 930 ,
	934 , 935 , 936 , 940 , 944 , 945 ,
	950 , 951 , 956 , 959 , 962 , 964 ,
	985 , 986 , 987 , 991 , 992 , 993 ,
T	1002 , 1005 , 1029 , 1032 , 1066 , 1069
\tab	1082 , 1094 , 1114 , 1130
\Test	718 , 736
\test ..	955 , 972 , 975 , 976 , 977 , 978 ,
	979 , 980 , 981 , 999 , 1016 , 1028 ,
	1036 , 1037 , 1038 , 1039 , 1040 ,
	1041 , 1042 , 1043 , 1044 , 1045 ,
	1046 , 1047 , 1048 , 1049 , 1050 ,
	1051 , 1052 , 1053 , 1054 , 1055 ,
	1056 , 1057 , 1058 , 1059 , 1060 ,
	1061 , 1065 , 1087 , 1088 , 1089 ,
	1090 , 1091 , 1092 , 1093 , 1094 ,
	1095 , 1096 , 1097 , 1098 , 1099 ,
	1100 , 1101 , 1102 , 1103 , 1104 ,
	Y
	\y 941 , 943 , 957 , 958 , 959 , 963 , 964 ,
	1030 , 1031 , 1032 , 1067 , 1068 , 1069
	Z
	\z
	172 , 179 , 183 , 203 , 204 , 205 ,
	206 , 207 , 208 , 209 , 210 , 211 ,
	212 , 213 , 214 , 215 , 216 , 217 ,
	218 , 231 , 234 , 344 , 365 , 541 , 542