

The pdfescape package

Heiko Oberdiek
<heiko.oberdiek at gmail.com>

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 {<cmd>} {<string>}
```

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 (*package)
```

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 \catcode123 1 % {
9 \catcode125 2 % }
10 \expandafter\let\expandafter\x\csname ver@pdfescape.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{pdfescape}{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@pdfescape.sty\endcsname
61 \ProvidesPackage{pdfescape}%
62 [2007/11/11 v1.8 Provides hex, PDF name and string conversions (H0)]

```

2.2 Catcodes

```

63 \expandafter\edef\csname PE@AtEnd\endcsname{%
64   \catcode64 \the\catcode64\relax
65 }
66 \catcode64 11 % @
67 \def\TMP@EnsureCode#1#2#3{%
68   \edef\PE@AtEnd{%
69     \PE@AtEnd
70     #1#2 \the#1#2\relax
71   }%
72   #1#2 #3\relax
73 }
74 \TMP@EnsureCode\catcode{0}{12}% ^^@
75 \TMP@EnsureCode\catcode{34}{12}% "
76 \TMP@EnsureCode\catcode{39}{12}% '
77 \TMP@EnsureCode\catcode{42}{12}% *
78 \TMP@EnsureCode\catcode{45}{12}% -
79 \TMP@EnsureCode\catcode{46}{12}% .
80 \TMP@EnsureCode\catcode{47}{12}% /
81 \TMP@EnsureCode\catcode{60}{12}% <
82 \TMP@EnsureCode\catcode{61}{12}% =
83 \TMP@EnsureCode\catcode{62}{12}% >
84 \TMP@EnsureCode\catcode{94}{7}% ^
85 \TMP@EnsureCode\catcode{96}{12}% `
86 \TMP@EnsureCode\uccode{34}{0}% "
87 \TMP@EnsureCode\uccode{48}{0}% 0
88 \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.

```

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

```

```

103   }%
104   \def\PE@strip@prefix#1>{%
105   \else
106   \let\PE@onelevel@sanitize\onelevel@sanitize
107   \fi
108 \else
109 \long\def\EdefSanitize#1#2{%
110   \begingroup
111   \csname @safe@activestruel\endcsname
112   \edef#1{#2}%
113   \expandafter\endgroup
114   \expandafter\def\expandafter#1\expandafter{%
115     \detokenize\expandafter{#1}%
116   }%
117 }%
118 \def\PE@onelevel@sanitize#1{%
119   \edef#1{\detokenize\expandafter{#1}}%
120 }%
121 \fi

```

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

```

122 \let\PE@sanitize\EdefSanitize

```

2.3.1 Space characters

`\PE@space@other`

```

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

```

`\PE@space@space`

```

126 \def\PE@space@space{ }

```

2.3.2 Space normalization

`\PE@SanitizeSpaceOther`

```

127 \def\PE@SanitizeSpaceOther#1{%
128   \edef#1{\expandafter\PE@SpaceToOther#1 \relax}%
129 }

```

`\PE@SpaceToOther`

```

130 \def\PE@SpaceToOther#1 #2\relax{%
131   #1%
132   \ifx\#2\%
133   \else
134     \PE@space@other
135     \@ReturnAfterFi{%
136       \PE@SpaceToOther#2\relax
137     }%
138   \fi
139 }

```

`\@ReturnAfterFi`

```

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

```

2.4 `\EdefUnescapeName`

`\EdefUnescapeName`

```

141 \def\EdefUnescapeName#1#2{%

```

```

142 \EdefSanitize#1{#2}%
143 \PE@SanitizeSpaceOther#1%
144 \PE@UnescapeName#1%
145 \PE@onelevel@sanitize#1%
146 }

```

\PE@UnescapeName

```

147 \begingroup
148 \catcode'\$=6 % hash
149 \catcode'\#=12 % other
150 \gdef\PE@UnescapeName$1{%
151   \begingroup
152     \PE@InitUccodeHexDigit
153     \def\PE@result{%}
154     \expandafter\PE@DeName$1#\relax\relax
155   \expandafter\endgroup
156   \expandafter\def\expandafter$1\expandafter{\PE@result}%
157 }%
158 \gdef\PE@DeName$1#$2$3{%
159   \ifx\relax$2%
160     \edef\PE@result{\PE@result$1}%
161     \let\PE@next\relax
162   \else
163     \ifx\relax$3%
164       % wrong escape sequence in input
165       \edef\PE@result{\PE@result$1#}%
166       \let\PE@next\relax
167     \else
168       \uppercase{%
169         \def\PE@testA{$2}%
170         \def\PE@testB{$3}%
171       }%
172       \ifcase\ifcase\expandafter\PE@TestUcHexDigit\PE@testA
173         \ifcase\expandafter\PE@TestUcHexDigit\PE@testB
174           \z@
175         \else
176           \@ne
177         \fi
178       \else
179         \@ne
180       \fi
181       \uccode\z@="\PE@testA\PE@testB\relax
182       \uppercase{%
183         \def\PE@temp{^^@}%
184       }%
185       \uccode\z@=\z@
186       \edef\PE@result{\PE@result$1\PE@temp}%
187       \let\PE@next\PE@DeName
188     \else
189       % wrong escape sequence in input
190       \edef\PE@result{\PE@result$1#}%
191       \def\PE@next{\PE@DeName$2$3}%
192     \fi
193   \fi
194 \PE@next
195 }%
196 }%
197 \endgroup

```

\PE@InitUccodeHexDigit

```

198 \def\PE@InitUccodeHexDigit{%
199   \uccode'a='A\relax

```

```

200 \uccode' b='B\relax
201 \uccode' c='C\relax
202 \uccode' d='D\relax
203 \uccode' e='E\relax
204 \uccode' f='F\relax
205 \uccode' A=\z@
206 \uccode' B=\z@
207 \uccode' C=\z@
208 \uccode' D=\z@
209 \uccode' E=\z@
210 \uccode' F=\z@
211 \uccode' 0=\z@
212 \uccode' 1=\z@
213 \uccode' 2=\z@
214 \uccode' 3=\z@
215 \uccode' 4=\z@
216 \uccode' 5=\z@
217 \uccode' 6=\z@
218 \uccode' 7=\z@
219 \uccode' 8=\z@
220 \uccode' 9=\z@
221 }

```

\PE@TestUcHexDigit

```

222 \def\PE@TestUcHexDigit#1{%
223 \ifnum'#1<48 % 0
224 \@ne
225 \else
226 \ifnum'#1>70 % F
227 \@ne
228 \else
229 \ifnum'#1>57 % 9
230 \ifnum'#1<65 % A
231 \@ne
232 \else
233 \z@
234 \fi
235 \else
236 \z@
237 \fi
238 \fi
239 \fi
240 }

```

2.5 \EdefUnescapeString

\EdefUnescapeString

```

241 \def\EdefUnescapeString#1#2{%
242 \EdefSanitize#1{#2}%
243 \PE@SanitizeSpaceOther#1%
244 \PE@NormalizeLineEnd#1%
245 \PE@UnescapeString#1%
246 \PE@onelevel@sanitize#1%
247 }

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

```

\PE@NormalizeLineEnd

```

252 \def\PE@NormalizeLineEnd##1{%

```

```

253 \def\PE@result{}%
254 \expandafter\PE@@NormalizeLineEnd##1#2\relax
255 \let##1\PE@result
256 }%

```

\PE@@NormalizeLineEnd

```

257 \def\PE@@NormalizeLineEnd##1#2##2{%
258 \ifx\relax##2%
259 \edef\PE@result{\PE@result##1}%
260 \let\PE@next\relax
261 \else
262 \edef\PE@result{\PE@result##1#1}%
263 \ifx#1##2% \f
264 \let\PE@next\PE@@NormalizeLineEnd
265 \else
266 \def\PE@next{\PE@@NormalizeLineEnd##2}%
267 \fi
268 \fi
269 \PE@next
270 }%
271 }%
272 \uppercase{%
273 \x 89%
274 }

275 \begingroup
276 \catcode'\|=0 %
277 \catcode'\|=12 %

```

\PE@UnescapeString

```

278 |gdef|PE@UnescapeString#1{%
279 |begingroup
280 |def|PE@result{}%
281 |expandafter|PE@DeString#1\|relax
282 |expandafter|endgroup
283 |expandafter|def|expandafter#1|expandafter{|PE@result}%
284 }%

```

\PE@DeString

```

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

```



```

308         |let|PE@next|PE@DeString
309     |else
310         |if|#2% backslash
311             |edef|PE@result{|PE@result#1}%
312             |let|PE@next|PE@CheckEndBackslash
313         |else
314             |ifnum'#2=10 % linefeed
315                 |edef|PE@result{|PE@result#1}%
316                 |let|PE@next|PE@DeString
317             |else
318                 |ifcase|PE@TestOctDigit#2%
319                     |edef|PE@result{|PE@result#1}%
320                     |def|PE@next{|PE@OctI#2}%
321                 |else
322                     |edef|PE@result{|PE@result#1#2}%
323                     |let|PE@next|PE@DeString
324                 |fi
325             |fi
326         |fi
327     |fi
328 |fi
329 |PE@next
330 }%

```

\PE@CheckEndBackslash

```

331 |gdef|PE@CheckEndBackslash#1{%
332     |ifx|relax#1%
333     |else
334         |edef|PE@result{|PE@result\}%
335         |expandafter|PE@DeString|expandafter#1%
336     |fi
337 }%

```

338 |endgroup

\PE@TestOctDigit

```

339 \def\PE@TestOctDigit#1{%
340     \ifnum'#1<48 % 0
341         \@ne
342     \else
343         \ifnum'#1>55 % 7
344             \@ne
345         \else
346             \z@
347         \fi
348     \fi
349 }

```

\PE@OctI

```

350 \def\PE@OctI#1#2{%
351     \ifcase\PE@TestOctDigit#2%
352         \def\PE@next{\PE@OctII{#1#2}}%
353     \else
354         \def\PE@next{\PE@OctAll#1#2}%
355     \fi
356     \PE@next
357 }

```

\PE@OctII

```

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

```

```

361 \else
362   \def\PE@next{\PE@OctAll{#1}#2}%
363 \fi
364 \PE@next
365 }

```

\PE@OctAll

```

366 \def\PE@OctAll#1{%
367   \uccode\z@=#1\relax
368   \uppercase{%
369     \edef\PE@result{\PE@result^^@}%
370   }%
371   \PE@DeString
372 }

```

2.6 User macros (pdfTeX analogues)

```

373 \begingroup\expandafter\expandafter\expandafter\endgroup
374 \expandafter\ifx\csname RequirePackage\endcsname\relax
375   \input pdftexcmds.sty\relax
376 \else
377   \RequirePackage{pdftexcmds}[2007/11/11]%
378 \fi
379 \begingroup\expandafter\expandafter\expandafter\endgroup
380 \expandafter\ifx\csname pdf@escapehex\endcsname\relax

```

\EdefEscapeHex

```

381 \long\def\EdefEscapeHex#1#2{%
382   \EdefSanitize#1{#2}%
383   \PE@SanitizeSpaceOther#1%
384   \PE@EscapeHex#1%
385 }%

```

\EdefUnescapeHex

```

386 \def\EdefUnescapeHex#1#2{%
387   \EdefSanitize#1{#2}%
388   \PE@UnescapeHex#1%
389 }%

```

\EdefEscapeName

```

390 \long\def\EdefEscapeName#1#2{%
391   \EdefSanitize#1{#2}%
392   \PE@SanitizeSpaceOther#1%
393   \PE@EscapeName#1%
394 }%

```

\EdefEscapeString

```

395 \long\def\EdefEscapeString#1#2{%
396   \EdefSanitize#1{#2}%
397   \PE@SanitizeSpaceOther#1%
398   \PE@EscapeString#1%
399 }%

```

400 \else

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

```

401 \long\def\PE@edefbabel#1#2#3{%
402   \begingroup
403     \csname @save@activetrue\endcsname
404     \edef#1{#2{#3}}%
405   \expandafter\endgroup
406   \expandafter\def\expandafter#1\expandafter{#1}%
407 }%

```

`\EdefEscapeHex`

```
408 \long\def\EdefEscapeHex#1#2{%
409   \PE@edefbabel#1\pdf@escapehex{#2}%
410 }%
```

`\EdefUnescapeHex`

```
411 \def\EdefUnescapeHex#1#2{%
412   \PE@edefbabel#1\pdf@unescapehex{#2}%
413 }%
```

`\EdefEscapeName`

```
414 \long\def\EdefEscapeName#1#2{%
415   \PE@edefbabel#1\pdf@escapename{#2}%
416 }%
```

`\EdefEscapeString`

```
417 \long\def\EdefEscapeString#1#2{%
418   \PE@edefbabel#1\pdf@escapestring{#2}%
419 }%

420 \PE@AtEnd
421 \expandafter\endinput
422 \fi
```

2.7 Help macros

2.7.1 Characters

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

`\PE@hash`

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

`\PE@backslash`

```
424 \begingroup
425   \escapechar=-1 %
426 \edef\x{\endgroup
427   \def\noexpand\PE@backslash{\string\\}}%
428 }
429 \x
```

2.7.2 Switch for ε -T_EX

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

2.8 Conversions

2.8.1 Conversion to hex string

`\PE@EscapeHex`

```
436 \ifPE@etex
437   \def\PE@EscapeHex#1{%
438     \edef#1{\expandafter\PE@ToHex#1\relax}%
439   }%
440 \else
441   \def\PE@EscapeHex#1{%
442     \def\PE@result{#1}%
443   }%
444 \fi
```

```

443 \expandafter\PE@ToHex#1\relax
444 \let#1\PE@result
445 }%
446 \fi

```

\PE@ToHex

```

447 \def\PE@ToHex#1{%
448 \ifx\relax#1%
449 \else
450 \PE@HexChar{#1}%
451 \expandafter\PE@ToHex
452 \fi
453 }%

```

\PE@HexChar

```

454 \ifPE@etex
455 \def\PE@HexChar#1{%
456 \PE@HexDigit{\numexpr\dimexpr.0625\dimexpr'#1sp\relax\relax\relax}%
457 \PE@HexDigit{%
458 \numexpr'#1-16*\dimexpr.0625\dimexpr'#1sp\relax\relax\relax
459 }%
460 }%
461 \else
462 \def\PE@HexChar#1{%
463 \dimen0='#1sp%
464 \dimen2=.0625\dimen0 %
465 \advance\dimen0-16\dimen2 %
466 \edef\PE@result{%
467 \PE@result
468 \PE@HexDigit{\dimen2 }%
469 \PE@HexDigit{\dimen0 }%
470 }%
471 }%
472 \fi

```

\PE@HexDigit

```

473 \def\PE@HexDigit#1{%
474 \expandafter\string
475 \ifcase#1%
476 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or 9\or
477 A\or B\or C\or D\or E\or F%
478 \fi
479 }

```

2.8.2 Character code to octal number

\PE@OctChar

```

480 \ifPE@etex
481 \def\PE@OctChar#1{%
482 \expandafter\PE@@OctChar
483 \the\numexpr\dimexpr.015625\dimexpr'#1sp\relax\relax
484 \expandafter\relax
485 \expandafter\relax
486 \the\numexpr\dimexpr.125\dimexpr'#1sp\relax\relax\relax
487 \relax
488 #1%
489 }%
490 \def\PE@@OctChar#1\relax#2\relax#3{%
491 \PE@backslash
492 #1%
493 \the\numexpr#2-8*#1\relax

```

```

494 \the\numexpr\dimexpr'#3sp\relax-8*#2\relax
495 }%
496 \else
497 \def\PE@OctChar#1{%
498 \dimen0='#1sp%
499 \dimen2=.125\dimen0 %
500 \dimen4=.125\dimen2 %
501 \advance\dimen0-8\dimen2 %
502 \advance\dimen2-8\dimen4 %
503 \edef\PE@result{%
504 \PE@result
505 \PE@backslash
506 \number\dimen4 %
507 \number\dimen2 %
508 \number\dimen0 %
509 }%
510 }%
511 \fi

```

2.8.3 Unpack hex string

\PE@UnescapeHex

```

512 \def\PE@UnescapeHex#1{%
513 \begingroup
514 \PE@InitUccodeHexDigit
515 \def\PE@result{%
516 \expandafter\PE@DeHex#1\relax\relax
517 \expandafter\endgroup
518 \expandafter\def\expandafter#1\expandafter{\PE@result}%
519 }

```

\PE@DeHex

```

520 \def\PE@DeHex#1#2{%
521 \ifx#2\relax
522 \ifx#1\relax
523 \let\PE@next\relax
524 \else
525 \uppercase{%
526 \def\PE@testA{#1}%
527 }%
528 \ifcase\expandafter\PE@TestUcHexDigit\PE@testA
529 \def\PE@next{%
530 \PE@DeHex#10\relax\relax
531 }%
532 \else
533 \let\PE@next\relax
534 \fi
535 \fi
536 \else
537 \uppercase{%
538 \def\PE@testA{#1}%
539 \def\PE@testB{#2}%
540 }%
541 \ifcase\expandafter\PE@TestUcHexDigit\PE@testA
542 \ifcase\expandafter\PE@TestUcHexDigit\PE@testB
543 \uccode\z@="\PE@testA\PE@testB\relax
544 \ifnum\uccode\z@=32 %
545 \let\PE@temp\PE@space@space
546 \else
547 \uppercase{%
548 \def\PE@temp{^^@}%
549 }%

```



```

606     \fi
607     \expandafter\PE@EscapeNameTokens
608     \fi
609 }%
610 \def\PE@EscapeNameHashChar#1#2{%
611     \PE@EscapeNameAdd{\PE@hash\string#1\string#2}%
612 }%

```

\PE@EscapeNameAdd

```

613 \ifPE@etex
614     \def\PE@EscapeNameAdd#1{#1}%
615 \else
616     \def\PE@EscapeNameAdd#1{%
617         \edef\PE@result{%
618             \PE@result
619             #1%
620         }%
621     }%
622 \fi

```

2.8.5 Conversion to PDF string

\PE@EscapeString

```

623 \ifPE@etex
624     \def\PE@EscapeString#1{%
625         \edef#1{\expandafter\PE@EscapeStringTokens#1\relax}%
626     }%
627 \else
628     \def\PE@EscapeString#1{%
629         \begingroup
630             \def\PE@result{}%
631             \expandafter\PE@EscapeStringTokens#1\relax
632         \expandafter\endgroup
633         \expandafter\def\expandafter#1\expandafter{\PE@result}%
634     }%
635 \fi

```

\PE@EscapeStringTokens

```

636 \def\PE@EscapeStringTokens#1{%
637     \ifx\relax#1%
638     \else
639         \ifnum'#1<33 %
640             \PE@OctChar#1%
641         \else
642             \ifnum'#1>126 %
643                 \PE@OctChar#1%
644             \else \ifnum'#1=40 \PE@EscapeStringAdd{\string\}% (
645                 \else\ifnum'#1=41 \PE@EscapeStringAdd{\string\)}% )
646                 \else\ifnum'#1=92 \PE@EscapeStringAdd{\string\\}% \
647                 \else
648                     \PE@EscapeStringAdd{#1}%
649                 \fi\fi\fi
650             \fi
651         \fi
652         \expandafter\PE@EscapeStringTokens
653     \fi
654 }%

```

\PE@EscapeStringAdd

```

655 \ifPE@etex
656     \def\PE@EscapeStringAdd#1{#1}%

```

```

657 \else
658   \def\PE@EscapeStringAdd#1{%
659     \edef\PE@result{%
660       \PE@result
661       #1%
662     }%
663   }%
664 \fi

665 \PE@AtEnd
666 \endpackage

```

3 Test

3.1 Catcode checks for loading

```

667 \begin{test1}

668 \catcode'\{=1 %
669 \catcode'\}=2 %
670 \catcode'\#=6 %
671 \catcode'\@=11 %
672 \expandafter\ifx\csname count@\endcsname\relax
673   \countdef\count@=255 %
674 \fi
675 \expandafter\ifx\csname @gobble\endcsname\relax
676   \long\def\@gobble#1{}%
677 \fi
678 \expandafter\ifx\csname @firstofone\endcsname\relax
679   \long\def\@firstofone#1{#1}%
680 \fi
681 \expandafter\ifx\csname loop\endcsname\relax
682   \expandafter\@firstofone
683 \else
684   \expandafter\@gobble
685 \fi
686 {%
687   \def\loop#1\repeat{%
688     \def\body{#1}%
689     \iterate
690   }%
691   \def\iterate{%
692     \body
693     \let\next\iterate
694   \else
695     \let\next\relax
696   \fi
697   \next
698 }%
699 \let\repeat=\fi
700 }%
701 \def\RestoreCatcodes{}
702 \count@=0 %
703 \loop
704   \edef\RestoreCatcodes{%
705     \RestoreCatcodes
706     \catcode\the\count@=\the\catcode\count@\relax
707   }%
708 \ifnum\count@<255 %
709   \advance\count@ 1 %
710 \repeat
711

```



```

712 \def\RangeCatcodeInvalid#1#2{%
713   \count@=#1\relax
714   \loop
715     \catcode\count@=15 %
716   \ifnum\count@<#2\relax
717     \advance\count@ 1 %
718   \repeat
719 }
720 \expandafter\ifx\csname LoadCommand\endcsname\relax
721 \def\LoadCommand{\input pdfescape.sty\relax}%
722 \fi
723 \def\Test{%
724   \RangeCatcodeInvalid{0}{47}%
725   \RangeCatcodeInvalid{58}{64}%
726   \RangeCatcodeInvalid{91}{96}%
727   \RangeCatcodeInvalid{123}{255}%
728   \catcode'\@=12 %
729   \catcode'\=0 %
730   \catcode'\{=1 %
731   \catcode'\}=2 %
732   \catcode'\#=6 %
733   \catcode'\[=12 %
734   \catcode'\]=12 %
735   \catcode'\%=14 %
736   \catcode'\ =10 %
737   \catcode13=5 %
738   \LoadCommand
739   \RestoreCatcodes
740 }
741 \Test
742 \csname @end\endcsname
743 \end
744 \</test1>

```

3.2 Macro tests

```

745 <*test2 | test3 | test4 | test5>
746 \NeedsTeXFormat{LaTeX2e}
747 \makeatletter

```

3.3 Test with \pdfescape... commands

```

748 <*test2>
749 \ProvidesFile{pdfescape-test2.tex}%
750   [2007/11/11 v1.8 Test with \string\pdfescape... commands]%
751 \</test2>

```

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

```

752 <*test3>
753 \ProvidesFile{pdfescape-test3.tex}%
754   [2007/11/11 v1.8 Test without \string\pdfescape..., with e-TeX]%
755 \</test3>

```

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

```

756 <*test4>
757 \ProvidesFile{pdfescape-test4.tex}%
758   [2007/11/11 v1.8 Test without \string\pdfescape... and e-TeX]%
759 \</test4>

```

3.6 Test with LuaTeX

```

760 <*test5>
761 \ProvidesFile{pdfescape-test5.tex}%
762   [2007/11/11 v1.8 Test with LuaTeX]%

```

```
763 </test5>
```

3.7 Check/ensure test preconditions

3.7.1 Check `\pdfescape...`

```
764 <*test2>
765 \ifundefined{pdfescapehex}{%
766   \PackageError{pdfescape-test2}{%
767     Missing \string\pdfescape... commands%
768   }{Test aborted.}%
769   \stop
770 }{}
771 </test2>

772 <*test3 | test4>
773 \let\pdfescapehex\undefined
774 \let\pdfunescapehex\undefined
775 \let\pdfescapename\undefined
776 \let\pdfescapestring\undefined
777 </test3 | test4>
```

3.7.2 Check ε -`\TeX`

```
778 <*test3>
779 \ifundefined{numexpr}{%
780   \PackageError{pdfescape-test3}{%
781     Missing \eTeX
782   }{Test aborted.}%
783   \stop
784 }{}
785 </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`

```
786 <*test5>
787 \ifundefined{directlua}{%
788   \PackageError{pdfescape-test5}{%
789     Missing LuaTeX%
790   }{Test aborted.}%
791   \stop
792 }{}
793 </test5>
```

3.8 Common part

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

```
794 \RequirePackage{qstest}
795 \IncludeTests{*}
796 \LogTests{log}{*}{*}
797
798 \newcommand*\ExpectVar}[2]{%
799   \ifx#1#2%
800   \else
801     \begingroup
802       \@onelevel@sanitize#1%
803       \@onelevel@sanitize#2%
804       \typeout{[#1] <> [#2]}% hash-ok
805     \endgroup
806   \fi
807   \Expect*{\ifx#1#2true\else false\fi}{true}%
808 }
809
```

```

810 \makeatletter
811 \begingroup
812 \gdef\AllBytes{}%
813 \count@=0 %
814 \catcode@=12 %
815 \@whilenum\count@<256 \do{%
816   \lccode@=\count@
817   \ifnum\count@=32 %
818     \xdef\AllBytes{\AllBytes\space}%
819   \else
820     \lowercase{%
821       \xdef\AllBytes{\AllBytes^^@}%
822     }%
823   \fi
824   \advance\count@ by 1 %
825 }%
826 \endgroup
827 \newcommand*{\AllBytesHex}{%
828 000102030405060708090A0B0C0D0E0F%
829 101112131415161718191A1B1C1D1E1F%
830 202122232425262728292A2B2C2D2E2F%
831 303132333435363738393A3B3C3D3E3F%
832 404142434445464748494A4B4C4D4E4F%
833 505152535455565758595A5B5C5D5E5F%
834 606162636465666768696A6B6C6D6E6F%
835 707172737475767778797A7B7C7D7E7F%
836 808182838485868788898A8B8C8D8E8F%
837 909192939495969798999A9B9C9D9E9F%
838 A0A1A2A3A4A5A6A7A8A9AAABACADAFAF%
839 B0B1B2B3B4B5B6B7B8B9BABBCBDBEBBF%
840 C0C1C2C3C4C5C6C7C8C9CACCCDCECF%
841 D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
842 E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
843 F0F1F2F3F4F5F6F7F8F9FAFBFCFDFF%
844 }
845 \@onelevel@sanitize\AllBytesHex
846 \expandafter\lowercase\expandafter{%
847 \expandafter\newcommand\expandafter*\expandafter\AllBytesHexLC
848 \expandafter{\AllBytesHex}%
849 }
850 \newcommand*{\AllBytesName}{%
851 \begingroup
852 \catcode'\#=12 %
853 \xdef\AllBytesName{%
854 #0#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
855 #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
856 #20!"#23$#25&'#28#29*+,-.#2F%
857 0123456789:;#3C=#3E?%
858 @ABCDEFGHIJKLMNO%
859 PQRSTUVWXYZ#5B\@backslashchar#5D^_%
860 'abcdefghijklmnop%
861 pqrstuvwxyz#7B|#7D|string~#7F%
862 #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
863 #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
864 #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
865 #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
866 #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
867 #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
868 #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
869 #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
870 }%
871 \endgroup

```

```

872 \@onelevel@sanitize\AllBytesName
873
874 \newcommand*{\AllBytesString}{}
875 \begin{group}
876 \def\|{|}%
877 \edef\%{\@percentchar}%
878 \catcode'\|=0 %
879 \catcode'\#=12 %
880 \catcode'\~=12 %
881 \catcode'\|=12 %
882 |xdef|AllBytesStringf%
883 \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
884 \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
885 \040!"#$%&'(\)*+,-./%
886 0123456789:;<=>?%
887 @ABCDEFGHIJKLMNO%
888 PQRSTUVWXYZ[\]^_%
889 'abcdefghijklmno%
890 pqrstuvwxyz{|}~\177%
891 \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
892 \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
893 \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
894 \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
895 \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
896 \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
897 \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
898 \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
899 }%
900 \end{group}
901 \@onelevel@sanitize\AllBytesString
902
903 <*test4>
904 \let\org@detokenize\detokenize
905 \let\detokenize\@undefined
906 \let\org@numexpr\numexpr
907 \let\numexpr\@undefined
908 </test4>
909 \RequirePackage{pdfescape}
910 <*test4>
911 \let\detokenize\org@detokenize
912 \let\numexpr\org@numexpr
913 </test4>
914
915 \begin{qstest}{all-hex}{\AllBytes, escapehex}
916 \EdefEscapeHex\x{\AllBytes}%
917 \Expect*{\x}*{\AllBytesHex}%
918 \ExpectVar\x\AllBytesHex
919 \end{qstest}
920
921 \begin{qstest}{all-unhex}{\AllBytesHex, unescapehex}
922 \EdefUnescapeHex\x{\AllBytesHex}%
923 \Expect*{\x}*{\AllBytes}%
924 \ExpectVar\x\AllBytes
925 \end{qstest}
926
927 \begin{qstest}{all-unhex-lc}{\AllBytesHexLC, unescapehex, lowercase}
928 \EdefUnescapeHex\x{\AllBytesHexLC}%
929 \Expect*{\x}*{\AllBytes}%
930 \ExpectVar\x\AllBytes
931 \end{qstest}
932
933 \begin{qstest}{unhex-incomplete}{unescapehex, incomplete}

```

```

934 \EdefUnescapeHex\x{4}%
935 \Expect*{\x}{@}%
936 \end{qstest}
937
938 \begin{qstest}{unhex-space}{unescapehex, space}
939 \EdefUnescapeHex\x{20}%
940 \Expect*{\x}{ }%
941 \ExpectVar\x\space
942 \end{qstest}
943
944 \begin{qstest}{unhex-spaces}{unescapehex, spaces}
945 \EdefUnescapeHex\x{204020204120}%
946 \def\y#1{%
947   \edef\z{#1\string @#1#1\string A#1}%
948 } \y{ }%
949 \Expect*{\x}*{\z}%
950 \ExpectVar\x\z
951 \end{qstest}
952
953 \begin{qstest}{unhex-hash}{unescapehex, hash}
954 \catcode'\#=12 %
955 \EdefUnescapeHex\x{#20}%
956 \ExpectVar\x\space
957 \end{qstest}
958
959 \begin{qstest}{unhex-invalid}{unescapehex, invalid}
960 \def\test#1#2{%
961   \EdefUnescapeHex\x{#1}%
962   \edef\y{#2}%
963   \@onelevel@sanitize\y
964   \ExpectVar\x\y
965 }%
966 \test{
967   \edef\x{\pdfunescapehex{4X}}%
968   \edef\y{\string @}%
969   \ifx\x\y
970   \else
971     \def~{\space}%
972     \typeout{*****}%
973     \typeout{* Your pdfTeX contains bug 777.~~~*}%
974     \typeout{* This test is redefined as dummy, *}%
975     \typeout{* because it triggers the bug.~~~~*}%
976     \typeout{*****}%
977   \def\test#1#2{%
978     \fi
979 } \test{
980 \test{X}{}%
981 \test{XY}{}%
982 \test{XYZ}{}%
983 \test{A}{^~a0}%
984 \test{AX}{^~a0}%
985 \test{XA}{^~a0}%
986 \test{XXAXX}{^~a0}%
987 \end{qstest}
988
989 \begin{qstest}{all-name}{\AllBytes, escapename}
990 \EdefEscapeName\x{\AllBytes}%
991 \Expect*{\x}*{\AllBytesName}%
992 \ExpectVar\x\AllBytesName
993 \end{qstest}
994
995 \begin{qstest}{all-string}{\AllBytes, escapestring}

```

```

996 \EdefEscapeString\x{\AllBytes}%
997 \Expect*{\x}*{\AllBytesString}%
998 \ExpectVar\x\AllBytesString
999 \end{qstest}
1000
1001 \begin{qstest}{uchexdigit}{unescape, uppercase hex digit}
1002 \catcode'\@=11 %
1003 \catcode0=12 %
1004 \def\test#1#2{%
1005   \uccode0=#1\relax
1006   \uppercase{%
1007     \def\x{^^@}%
1008   }%
1009   \Expect*{%
1010     \ifcase\expandafter\PE@TestUcHexDigit\x
1011     true%
1012     \else
1013     false%
1014     \fi
1015   }{#2}%
1016 }%
1017 \def\range#1#2#3{%
1018   \count0=#1\relax
1019   \loop
1020   \ifnum\count0<#2\relax
1021     \test{\count0}{#3}%
1022     \advance\count0 by 1 %
1023   \repeat
1024 }%
1025 \range{0}{47}{false}%
1026 \range{48}{57}{true}%
1027 \range{58}{64}{false}%
1028 \range{65}{70}{true}%
1029 \range{71}{255}{false}%
1030 \end{qstest}
1031
1032 \begin{qstest}{unescapename}{unescapename}
1033 \def\test#1#2{%
1034   \EdefUnescapeName\x{#1}%
1035   \edef\y{#2}%
1036   \@onelevel@sanitize\y
1037   \ExpectVar\x\y
1038 }%
1039 \catcode'\#=12 %
1040 \catcode0=12 %
1041 \test{}{}%
1042 \test{x}{x}%
1043 \test{xy}{xy}%
1044 \test{#}{#}%
1045 \test{##}{##}%
1046 \test{###}{###}%
1047 \test{####}{####}%
1048 \test{#x}{#x}%
1049 \test{#xy}{#xy}%
1050 \test{#1}{#1}%
1051 \test{#40}{@}%
1052 \test{#400}{@0}%
1053 \test{#4x0}{#4x0}%
1054 \test{#ab}{^^ab}%
1055 \test{#00}{^^@}%
1056 \test{x#40y#40z}{x@y@z}%
1057 \test{#40#40#40#40}{@@@}%

```

```

1058 \test{a#x}{a#x}%
1059 \test{a#xy}{a#xy}%
1060 \test{a#1}{a#1}%
1061 \test{a#40}{a@}%
1062 \test{a#400}{a@0}%
1063 \test{#20}{ }%
1064 \test{a#20}{a }%
1065 \test{a#20b}{a b}%
1066 \test{a#20#20#20b}{a \space\space b}%
1067 \end{qstest}
1068
1069 \begin{qstest}{unescapestring}{unescapestring}
1070 \def\test#1#2{%
1071   \EdefUnescapeString\x{#1}%
1072   \edef\y{#2}%
1073   \@onelevel@sanitize\y
1074   \ExpectVar\x\y
1075 }%
1076 \catcode0=12 %
1077 \def\DefChar#1#2{%
1078   \begingroup
1079   \uccode0=#2\relax
1080   \uppercase{\endgroup
1081   \def#1{^@}%
1082   }%
1083 }%
1084 \DefChar\nul{0}%
1085 \DefChar\one{1}%
1086 \DefChar\bel{8}%
1087 \DefChar\tab{9}%
1088 \DefChar\lf{10}%
1089 \DefChar\ff{12}%
1090 \DefChar\cr{13}%
1091 \DefChar\{92}%
1092 \test{}{}%
1093 \test{a}{a}%
1094 \test{\}{}%
1095 \test{\\}{}%
1096 \test{\\y}{\y}%
1097 \test{\\000}{\nul}%
1098 \test{\\b}{\bel}%
1099 \test{\\t}{\tab}%
1100 \test{\\n}{\lf}%
1101 \test{\\f}{\ff}%
1102 \test{\\r}{\cr}%
1103 \test{\\(}{(}%
1104 \test{\\)}{)}%
1105 \test{\\040}{ }%
1106 \test{\\100}{@}%
1107 \test{\\40}{ }%
1108 \test{\\1}{\one}%
1109 \test{\\01}{\one}%
1110 \test{\\001}{\one}%
1111 \test{\\18}{\one8}%
1112 \test{\\018}{\one8}%
1113 \test{\\0018}{\one8}%
1114 \test{x\\}{x}%
1115 \test{x\\\\}{x\\}%
1116 \test{x\\\\y}{x\y}%
1117 \test{x\\000}{x\nul}%
1118 \test{x\\b}{x\bel}%
1119 \test{x\\t}{x\tab}%

```

```

1120 \test{x\n}{x\lf}%
1121 \test{x\ff}{x\ff}%
1122 \test{x\r}{x\cr}%
1123 \test{x\(){x}%
1124 \test{x\)}{x}%
1125 \test{x\040}{x }%
1126 \test{x\100}{x@}%
1127 \test{x\40}{x }%
1128 \test{x\1}{x\one}%
1129 \test{x\01}{x\one}%
1130 \test{x\001}{x\one}%
1131 \test{x\18}{x\one8}%
1132 \test{x\018}{x\one8}%
1133 \test{x\0018}{x\one8}%
1134 \test{\b\t\n\f\r\(\)\000\040}{%
1135 \bel\tab\lf\ff\cr()\nul\space
1136 }%
1137 \test{\lf}{}%
1138 \test{x\lf}{x}%
1139 \test{\cr}{\lf}%
1140 \test{\cr\lf}{\lf}%
1141 \test{\lf}{\lf}%
1142 \test{\lf\cr}{\lf\lf}%
1143 \test{x\cr}{x\lf}%
1144 \test{x\cr\lf}{x\lf}%
1145 \test{x\lf}{x\lf}%
1146 \test{x\lf\cr}{x\lf\lf}%
1147 \test{x\cr\lf y\cr}{xy\lf}%
1148 \end{qstest}
1149 \stop
1150 </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: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
```

¹[ftp://ftp.ctan.org/tex-archive/](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 (`teTeX`, `mikTeX`, ...) relies on file name databases, you must refresh these. For example, `teTeX` 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 \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{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>\AllBytes</code> 812, 818, 821, 915, 916, 923, 924, 929, 930, 989, 990, 995, 996	
<code>\#</code>	149, 285, 310, 670, 732, 852, 879, 954, 1039	<code>\AllBytesHex</code>	
<code>\\$</code>	148	. 827, 845, 848, 917, 918, 921, 922	
<code>\%</code>	735, 877	<code>\AllBytesHexLC</code>	
<code>\(</code>	644, 885	847, 927, 928	
<code>\)</code>	645, 885	<code>\AllBytesName</code> .	
<code>\@</code>	671, 728, 1002	850, 853, 872, 991, 992	
<code>\@ReturnAfterFi</code>	135, <u>140</u>	<code>\AllBytesString</code> ...	
<code>\@backslashchar</code>	859	874, 901, 997, 998	
<code>\@firstofone</code>	679, 682	B	
<code>\@gobble</code>	676, 684	<code>\begin</code> .	915, 921, 927, 933, 938, 944, 953, 959, 989, 995, 1001, 1032, 1069
<code>\@ifundefined</code>	765, 779, 787	<code>\bel</code>	1086, 1098, 1118, 1135
<code>\@one</code> ..	176, 179, 224, 227, 231, 341, 344	<code>\body</code>	688, 692
<code>\@onelevel@sanitize</code>	106, 802, 803, 845, 872, 901, 963, 1036, 1073	C	
<code>\@percentchar</code>	877	<code>\catcode</code>	3, 4, 5, 6, 7, 8, 9, 17, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 64, 66, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 124, 148, 149, 276, 277, 668, 669, 670, 671, 706, 715, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 814, 852, 878, 879, 880, 881, 954, 1002, 1003, 1039, 1040, 1076
<code>\@undefined</code> 52, 773, 774, 775, 776, 905, 907	<code>\count</code>	1018, 1020, 1021, 1022
<code>\@whilenum</code>	815	<code>\count@</code>	673, 702, 706, 708, 709, 713, 715, 716, 717, 813, 815, 816, 817, 824
<code>\[</code>	733	<code>\countdef</code>	673
<code>\</code> ..	132, 277, 427, 646, 729, 881, 888, 1091, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1137, 1138, 1147	<code>\cr</code> ..	1090, 1102, 1122, 1135, 1139, 1140, 1142, 1143, 1144, 1146, 1147
<code>\{</code>	668, 730	<code>\csname</code>	10, 18, 44, 60, 63, 90, 93, 100, 111, 374, 380, 403, 432, 672, 675, 678, 681, 720, 742
<code>\}</code>	334, 669, 731	D	
<code>\]</code>	734	<code>\DefChar</code>	1077, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091
<code>\ </code>	276, 281, 876, 878	<code>\detokenize</code> ...	115, 119, 904, 905, 911
<code>\~</code>	880	<code>\dimen</code> .	463, 464, 465, 468, 469, 498, 499, 500, 501, 502, 506, 507, 508
Numbers		<code>\dimexpr</code>	456, 458, 483, 486, 494
<code>\0</code>	883, 884, 885	<code>\do</code>	815
<code>\1</code>	890	E	
<code>\2</code>	891, 892, 893, 894	<code>\EdefEscapeHex</code>	2, <u>381</u> , <u>408</u> , 916
<code>\3</code>	895, 896, 897, 898	<code>\EdefEscapeName</code>	<u>390</u> , <u>414</u> , 990
<code>\8</code>	249	<code>\EdefEscapeString</code>	<u>395</u> , <u>417</u> , 996
<code>\9</code>	250	<code>\EdefSanitize</code>	3, <u>89</u> , 122, 142, 242, 382, 387, 391, 396
<code>_</code>	124, 646, 736	A	
A		<code>\advance</code>	465, 501, 502, 709, 717, 824, 1022
<code>\advance</code>	465, 501, 502, 709, 717, 824, 1022	<code>\aftergroup</code>	26
<code>\aftergroup</code>	26		

262, 369, 442, 444, 466, 467,
 503, 504, 515, 518, 551, 572,
 574, 617, 618, 630, 633, 659, 660
 \PE@sanitize [122](#)
 \PE@SanitizeSpaceOther
 [127](#), [143](#), [243](#), [383](#), [392](#), [397](#)
 \PE@space@other [123](#), [134](#)
 \PE@space@space [126](#), [545](#)
 \PE@SpaceToOther [128](#), [130](#)
 \PE@strip@prefix [102](#), [104](#)
 \PE@temp [183](#), [186](#), [545](#), [548](#), [551](#)
 \PE@testA [169](#),
 [172](#), [181](#), [526](#), [528](#), [538](#), [541](#), [543](#)
 \PE@testB . [170](#), [173](#), [181](#), [539](#), [542](#), [543](#)
 \PE@TestOctDigit [339](#), [351](#), [359](#)
 \PE@TestUcHexDigit
 [172](#), [173](#), [222](#), [528](#), [541](#), [542](#), [1010](#)
 \PE@ToHex [438](#), [443](#), [447](#)
 \PE@UnescapeHex [388](#), [512](#)
 \PE@UnescapeName [144](#), [147](#)
 \PE@UnescapeString [245](#), [278](#)
 \ProvidesFile [749](#), [753](#), [757](#), [761](#)
 \ProvidesPackage [15](#), [61](#)

R

\range [1017](#), [1025](#), [1026](#), [1027](#), [1028](#), [1029](#)
 \RangeCatcodeInvalid
 [712](#), [724](#), [725](#), [726](#), [727](#)
 \repeat [687](#), [699](#), [710](#), [718](#), [1023](#)
 \RequirePackage [377](#), [794](#), [909](#)
 \RestoreCatcodes .. [701](#), [704](#), [705](#), [739](#)

S

\space .. [818](#), [941](#), [956](#), [971](#), [1066](#), [1135](#)
 \stop [769](#), [783](#), [791](#), [1149](#)

T

\tab [1087](#), [1099](#), [1119](#), [1135](#)
 \Test [723](#), [741](#)
 \test .. [960](#), [977](#), [980](#), [981](#), [982](#), [983](#),
 [984](#), [985](#), [986](#), [1004](#), [1021](#), [1033](#),
 [1041](#), [1042](#), [1043](#), [1044](#), [1045](#),
 [1046](#), [1047](#), [1048](#), [1049](#), [1050](#),
 [1051](#), [1052](#), [1053](#), [1054](#), [1055](#),
 [1056](#), [1057](#), [1058](#), [1059](#), [1060](#),
 [1061](#), [1062](#), [1063](#), [1064](#), [1065](#),
 [1066](#), [1070](#), [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, 1131, 1132, 1133, 1134,
 1137, 1138, 1139, 1140, 1141,
 1142, 1143, 1144, 1145, 1146, 1147
 \the [64](#), [70](#), [483](#), [486](#), [493](#), [494](#), [706](#)
 \TMP@EnsureCode [67](#), [74](#), [75](#), [76](#), [77](#), [78](#),
 [79](#), [80](#), [81](#), [82](#), [83](#), [84](#), [85](#), [86](#), [87](#), [88](#)
 \typeout .. [804](#), [972](#), [973](#), [974](#), [975](#), [976](#)

U

\uccode [86](#), [87](#),
 [88](#), [181](#), [185](#), [199](#), [200](#), [201](#), [202](#),
 [203](#), [204](#), [205](#), [206](#), [207](#), [208](#),
 [209](#), [210](#), [211](#), [212](#), [213](#), [214](#),
 [215](#), [216](#), [217](#), [218](#), [219](#), [220](#),
 [249](#), [250](#), [367](#), [543](#), [544](#), [1005](#), [1079](#)
 \uppercase [168](#), [182](#),
 [272](#), [368](#), [525](#), [537](#), [547](#), [1006](#), [1080](#)

W

\write [20](#), [46](#)

X

\x [10](#), [11](#), [14](#), [19](#), [23](#), [25](#),
 [45](#), [50](#), [60](#), [125](#), [251](#), [273](#), [426](#),
 [429](#), [916](#), [917](#), [918](#), [922](#), [923](#),
 [924](#), [928](#), [929](#), [930](#), [934](#), [935](#),
 [939](#), [940](#), [941](#), [945](#), [949](#), [950](#),
 [955](#), [956](#), [961](#), [964](#), [967](#), [969](#),
 [990](#), [991](#), [992](#), [996](#), [997](#), [998](#),
 [1007](#), [1010](#), [1034](#), [1037](#), [1071](#), [1074](#)

Y

\y [946](#), [948](#), [962](#), [963](#), [964](#), [968](#), [969](#),
 [1035](#), [1036](#), [1037](#), [1072](#), [1073](#), [1074](#)

Z

\z [947](#), [949](#), [950](#)
 \z@ ... [174](#), [181](#), [185](#), [205](#), [206](#), [207](#),
 [208](#), [209](#), [210](#), [211](#), [212](#), [213](#),
 [214](#), [215](#), [216](#), [217](#), [218](#), [219](#),
 [220](#), [233](#), [236](#), [346](#), [367](#), [543](#), [544](#)