

# The pdfescape package

Heiko Oberdiek\*

2019/12/09 v1.15

## Abstract

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

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Additional unescape macros	2
1.2	Sanitizing macro	3
<b>2</b>	<b>Implementation</b>	<b>3</b>
2.1	Reload check and package identification	3
2.2	Catcodes	4
2.3	Load package	5
2.4	Sanitizing	5
2.4.1	Space characters	6
2.4.2	Space normalization	6
2.5	<code>\EdefUnescapeName</code>	7
2.6	<code>\EdefUnescapeString</code>	8
2.7	User macros (pdfTeX analogues)	12
2.8	Help macros	13
2.8.1	Characters	13
2.8.2	Switch for $\epsilon$ -TeX	13
2.9	Conversions	13
2.9.1	Conversion to hex string	13
2.9.2	Character code to octal number	14
2.9.3	Unpack hex string	15
2.9.4	Conversion to PDF name	16
2.9.5	Conversion to PDF string	17
<b>3</b>	<b>Installation</b>	<b>18</b>
3.1	Download	18
3.2	Bundle installation	18
3.3	Package installation	19
3.4	Refresh file name databases	19
3.5	Some details for the interested	19

---

\*Please report any issues at <https://github.com/ho-tex/pdfescape/issues>

<b>4 History</b>	<b>19</b>
[2007/02/21 v1.0]	19
[2007/02/25 v1.1]	20
[2007/03/20 v1.2]	20
[2007/04/11 v1.3]	20
[2007/04/21 v1.4]	20
[2007/08/27 v1.5]	20
[2007/09/09 v1.6]	20
[2007/10/27 v1.7]	20
[2007/11/11 v1.8]	20
[2010/03/01 v1.9]	20
[2010/11/12 v1.10]	20
[2011/01/30 v1.11]	21
[2011/04/04 v1.12]	21
[2011/11/25 v1.13]	21
[2016/05/16 v1.14]	21
[2019/12/09 v1.15]	21
<b>5 Index</b>	<b>21</b>

# 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  $\geq 1.30$  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  $\epsilon$ -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<sup>A</sup>T<sub>E</sub>X.

```
2 \begingroup\catcode61\catcode48\catcode32=10\relax%
3 \catcode13=5 % ^~M
4 \endlinechar=13 %
5 \catcode35=6 % #
6 \catcode39=12 % '
7 \catcode44=12 % ,
8 \catcode45=12 % -
9 \catcode46=12 % .
10 \catcode58=12 % :
11 \catcode64=11 % @
12 \catcode123=1 % {
13 \catcode125=2 % }
14 \expandafter\let\expandafter\x\csname ver@pdfescape.sty\endcsname
15 \ifx\x\relax % plain-TeX, first loading
16 \else
17 \def\empty{}%
18 \ifx\x\empty % LaTeX, first loading,
19 % variable is initialized, but \ProvidesPackage not yet seen
20 \else
21 \expandafter\ifx\csname PackageInfo\endcsname\relax
22 \def\x#1#2{%
23 \immediate\write-1{Package #1 Info: #2.}%
24 }%
25 \else
26 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
27 \fi
28 \x{pdfescape}{The package is already loaded}%
29 \aftergroup\endinput
30 \fi
31 \fi
32 \endgroup%
```

Package identification:

```
33 \begingroup\catcode61\catcode48\catcode32=10\relax%
34 \catcode13=5 % ^~M
35 \endlinechar=13 %
```

```

36 \catcode35=6 % #
37 \catcode39=12 % '
38 \catcode40=12 % (
39 \catcode41=12 % )
40 \catcode44=12 % ,
41 \catcode45=12 % -
42 \catcode46=12 % .
43 \catcode47=12 % /
44 \catcode58=12 % :
45 \catcode64=11 % @
46 \catcode91=12 % [
47 \catcode93=12 % ]
48 \catcode123=1 % {
49 \catcode125=2 % }
50 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
51   \def\x#1#2#3[#4]{\endgroup
52     \immediate\write-1{Package: #3 #4}%
53     \xdef#1{#4}%
54   }%
55 \else
56   \def\x#1#2[#3]{\endgroup
57     #2[#3]%
58     \ifx#1\@undefined
59       \xdef#1{#3}%
60     \fi
61     \ifx#1\relax
62       \xdef#1{#3}%
63     \fi
64   }%
65 \fi
66 \expandafter\x\csname ver@pdfescape.sty\endcsname
67 \ProvidesPackage{pdfescape}%
68 [2019/12/09 v1.15 Implements pdfTeX's escape features (HO)]%

```

## 2.2 Catcodes

```

69 \begingroup\catcode61\catcode48\catcode32=10\relax%
70 \catcode13=5 % ^M
71 \endlinechar=13 %
72 \catcode123 1 % {
73 \catcode125 2 % }
74 \catcode64 11 %
75 \def\x{\endgroup
76   \expandafter\edef\csname PE@AtEnd\endcsname{%
77     \endlinechar=\the\endlinechar\relax
78     \catcode13=\the\catcode13\relax
79     \catcode32=\the\catcode32\relax
80     \catcode35=\the\catcode35\relax
81     \catcode61=\the\catcode61\relax
82     \catcode64=\the\catcode64\relax
83     \catcode123=\the\catcode123\relax
84     \catcode125=\the\catcode125\relax
85   }%
86 }%
87 \x\catcode61\catcode48\catcode32=10\relax%
88 \catcode13=5 % ^M
89 \endlinechar=13 %
90 \catcode35=6 % #

```

```

91 \catcode64=11 % @
92 \catcode123=1 % {
93 \catcode125=2 % }
94 \def\TMP@EnsureCode#1#2#3{%
95   \edef\PE@AtEnd{%
96     \PE@AtEnd
97     #1#2=\the#1#2\relax
98   }%
99   #1#2=#3\relax
100 }
101 \TMP@EnsureCode\catcode{0}{12}% ^^@
102 \TMP@EnsureCode\catcode{34}{12}% "
103 \TMP@EnsureCode\catcode{36}{3}% $
104 \TMP@EnsureCode\catcode{38}{4}% &
105 \TMP@EnsureCode\catcode{39}{12}% '
106 \TMP@EnsureCode\catcode{42}{12}% *
107 \TMP@EnsureCode\catcode{45}{12}% -
108 \TMP@EnsureCode\catcode{46}{12}% .
109 \TMP@EnsureCode\catcode{47}{12}% /
110 \TMP@EnsureCode\catcode{60}{12}% <
111 \TMP@EnsureCode\catcode{62}{12}% >
112 \TMP@EnsureCode\catcode{91}{12}% [
113 \TMP@EnsureCode\catcode{93}{12}% ]
114 \TMP@EnsureCode\catcode{94}{7}% ^
115 \TMP@EnsureCode\catcode{96}{12}% `
116 \TMP@EnsureCode\uccode{34}{0}% "
117 \TMP@EnsureCode\uccode{48}{0}% 0
118 \TMP@EnsureCode\uccode{61}{0}% =
119 \edef\PE@AtEnd{\PE@AtEnd\noexpand\endinput}

```

## 2.3 Load package

```

120 \begingroup\expandafter\expandafter\expandafter\endgroup
121 \expandafter\ifx\csname RequirePackage\endcsname\relax
122   \def\TMP@RequirePackage#1[#2]{%
123     \begingroup\expandafter\expandafter\expandafter\endgroup
124     \expandafter\ifx\csname ver@#1.sty\endcsname\relax
125       \input #1.sty\relax
126     \fi
127   }%
128   \TMP@RequirePackage{ltxcms}[2010/04/08]%
129 \else
130   \RequirePackage{ltxcms}[2010/04/08]%
131 \fi

```

## 2.4 Sanitizing

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

```

132 \begingroup\expandafter\expandafter\expandafter\endgroup
133 \expandafter\ifx\csname detokenize\endcsname\relax
134   \long\def\EdefSanitize#1#2{%
135     \begingroup
136       \csname @safe@activestruel\endcsname
137       \edef#1{#2}%
138       \PE@onelevel@sanitize#1%
139     \expandafter\endgroup
140     \expandafter\def\expandafter#1\expandafter{#1}%
141   }%

```

```

142 \begingroup\expandafter\expandafter\expandafter\endgroup
143 \expandafter\ifx\csname @onelevel@sanitize\endcsname\relax
144 \def\PE@onelevel@sanitize#1{%
145 \edef#1{\expandafter\PE@strip@prefix\meaning#1}%
146 }%
147 \def\PE@strip@prefix#1>{}%
148 \else
149 \let\PE@onelevel@sanitize\@onelevel@sanitize
150 \fi
151 \else
152 \long\def\EdefSanitize#1#2{%
153 \begingroup
154 \csname @safe@activestrue\endcsname
155 \edef#1{#2}%
156 \expandafter\endgroup
157 \expandafter\def\expandafter#1\expandafter{%
158 \detokenize\expandafter{#1}%
159 }%
160 }%
161 \def\PE@onelevel@sanitize#1{%
162 \edef#1{\detokenize\expandafter{#1}}%
163 }%
164 \fi

```

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

```
165 \let\PE@sanitize\EdefSanitize
```

### 2.4.1 Space characters

`\PE@space@other`

```

166 \begingroup
167 \catcode'\ =12\relax%
168 \def\x{\endgroup\def\PE@space@other{ }}\x\relax

```

`\PE@space@space`

```
169 \def\PE@space@space{ }
```

### 2.4.2 Space normalization

`\PE@SanitizeSpaceOther`

```

170 \def\PE@SanitizeSpaceOther#1{%
171 \edef#1{\expandafter\PE@SpaceToOther#1 \relax}%
172 }

```

`\PE@SpaceToOther`

```

173 \def\PE@SpaceToOther#1 #2\relax{%
174 #1%
175 \ifx\#2\%
176 \else
177 \PE@space@other
178 \ltx@ReturnAfterFi{%
179 \PE@SpaceToOther#2\relax
180 }%
181 \fi
182 }

```

## 2.5 \EdefUnescapeName

\EdefUnescapeName

```
183 \def\EdefUnescapeName#1#2{%
184   \EdefSanitize#1{#2}%
185   \PE@SanitizeSpaceOther#1%
186   \PE@UnescapeName#1%
187   \PE@onelevel@sanitize#1%
188 }
```

\PE@UnescapeName

```
189 \begingroup
190   \catcode'\$=6 % hash
191   \catcode'\#=12 % other
192   \gdef\PE@UnescapeName$1{%
193     \begingroup
194       \PE@InitUccodeHexDigit
195       \def\PE@result{}%
196       \expandafter\PE@DeName$1#\relax\relax
197       \expandafter\endgroup
198       \expandafter\def\expandafter$1\expandafter{\PE@result}%
199     }%
200   \gdef\PE@DeName$1#$2$3{%
201     \ifx\relax$2%
202       \edef\PE@result{\PE@result$1}%
203       \let\PE@next\relax
204     \else
205       \ifx\relax$3%
206         % wrong escape sequence in input
207         \edef\PE@result{\PE@result$1#}%
208         \let\PE@next\relax
209       \else
210         \uppercase{%
211           \def\PE@testA{$2}%
212           \def\PE@testB{$3}%
213         }%
214         \ifcase\ifcase\expandafter\PE@TestUcHexDigit\PE@testA
215           \ifcase\expandafter\PE@TestUcHexDigit\PE@testB
216             \ltx@zero
217           \else
218             \ltx@one
219           \fi
220         \else
221           \ltx@one
222         \fi
223         \uccode\ltx@zero="\PE@testA\PE@testB\relax
224         \uppercase{%
225           \def\PE@temp{^^@}%
226         }%
227         \uccode\ltx@zero=\ltx@zero
228         \edef\PE@result{\PE@result$1\PE@temp}%
229         \let\PE@next\PE@DeName
230       \else
231         % wrong escape sequence in input
232         \edef\PE@result{\PE@result$1#}%
233         \def\PE@next{\PE@DeName$2$3}%
234       \fi
235     \fi
```

```

236     \fi
237     \PE@next
238 }%
239 \endgroup

```

`\PE@InitUccodeHexDigit`

```

240 \def\PE@InitUccodeHexDigit{%
241   \uccode'a='A\relax
242   \uccode'b='B\relax
243   \uccode'c='C\relax
244   \uccode'd='D\relax
245   \uccode'e='E\relax
246   \uccode'f='F\relax
247   \uccode'A=\ltx@zero
248   \uccode'B=\ltx@zero
249   \uccode'C=\ltx@zero
250   \uccode'D=\ltx@zero
251   \uccode'E=\ltx@zero
252   \uccode'F=\ltx@zero
253   \uccode'0=\ltx@zero
254   \uccode'1=\ltx@zero
255   \uccode'2=\ltx@zero
256   \uccode'3=\ltx@zero
257   \uccode'4=\ltx@zero
258   \uccode'5=\ltx@zero
259   \uccode'6=\ltx@zero
260   \uccode'7=\ltx@zero
261   \uccode'8=\ltx@zero
262   \uccode'9=\ltx@zero
263 }

```

`\PE@TestUcHexDigit`

```

264 \def\PE@TestUcHexDigit#1{%
265   \ifnum'#1<48 % 0
266     \ltx@one
267   \else
268     \ifnum'#1>70 % F
269       \ltx@one
270     \else
271       \ifnum'#1>57 % 9
272         \ifnum'#1<65 % A
273           \ltx@one
274         \else
275           \ltx@zero
276         \fi
277       \else
278         \ltx@zero
279       \fi
280     \fi
281   \fi
282 }

```

## 2.6 `\EdefUnescapeString`

`\EdefUnescapeString`

```

283 \def\EdefUnescapeString#1#2{%
284   \EdefSanitize#1{#2}%
285   \PE@SanitizeSpaceOther#1%

```

```

286 \PE@NormalizeLineEnd#1%
287 \PE@UnescapeString#1%
288 \PE@onelevel@sanitize#1%
289 }

```

```

290 \begingroup
291 \uccode'\8=10 % lf
292 \uccode'\9=13 % cr
293 \def\x#1#2{\endgroup

```

\PE@NormalizeLineEnd

```

294 \def\PE@NormalizeLineEnd##1{%
295 \def\PE@result{}%
296 \expandafter\PE@@NormalizeLineEnd##1#2\relax
297 \let##1\PE@result
298 }%

```

\PE@@NormalizeLineEnd

```

299 \def\PE@@NormalizeLineEnd##1#2##2{%
300 \ifx\relax##2%
301 \edef\PE@result{\PE@result##1}%
302 \let\PE@next\relax
303 \else
304 \edef\PE@result{\PE@result##1#1}%
305 \ifx#1##2% lf
306 \let\PE@next\PE@@NormalizeLineEnd
307 \else
308 \def\PE@next{\PE@@NormalizeLineEnd##2}%
309 \fi
310 \fi
311 \PE@next
312 }%
313 }%
314 \uppercase{%
315 \x 89%
316 }

```

```

317 \begingroup
318 \catcode'\|=0 %
319 \catcode'\|=12 %

```

\PE@UnescapeString

```

320 |gdef|PE@UnescapeString#1{%
321 |begingroup
322 |def|PE@result{}%
323 |expandafter|PE@DeString#1\|relax
324 |expandafter|endgroup
325 |expandafter|def|expandafter#1|expandafter{|PE@result}%
326 }%

```

\PE@DeString

```

327 |gdef|PE@DeString#1\#2{%
328 |ifx|relax#2%
329 |edef|PE@result{|PE@result#1}%
330 |let|PE@next|relax
331 |else
332 |if n#2%
333 |uccode|ltx@zero=10 %

```

```

334     |elseif r#2%
335         |uccode|ltx@zero=13 %
336     |elseif t#2%
337         |uccode|ltx@zero=9 %
338     |elseif b#2%
339         |uccode|ltx@zero=8 %
340     |elseif f#2%
341         |uccode|ltx@zero=12 %
342     |else
343         |uccode|ltx@zero=|ltx@zero
344     |fi|fi|fi|fi|fi
345     |ifnum|uccode|ltx@zero>|ltx@zero
346         |uppercase{%
347             |edef|PE@temp{^^@}%
348         }%
349         |edef|PE@result{|PE@result#1|PE@temp}%
350         |let|PE@next|PE@DeString
351     |else
352         |if|#2% backslash
353             |edef|PE@result{|PE@result#1}%
354             |let|PE@next|PE@CheckEndBackslash
355         |else
356             |ifnum'#2=10 % linefeed
357                 |edef|PE@result{|PE@result#1}%
358                 |let|PE@next|PE@DeString
359             |else
360                 |ifcase|PE@TestOctDigit#2%
361                     |edef|PE@result{|PE@result#1}%
362                     |def|PE@next{|PE@OctI#2}%
363                 |else
364                     |edef|PE@result{|PE@result#1#2}%
365                     |let|PE@next|PE@DeString
366                 |fi
367             |fi
368         |fi
369     |fi
370     |fi
371     |PE@next
372 }%

```

\PE@CheckEndBackslash

```

373 |gdef|PE@CheckEndBackslash#1{%
374     |ifx|relax#1%
375     |else
376         |edef|PE@result{|PE@result\}%
377         |expandafter|PE@DeString|expandafter#1%
378     |fi
379 }%

```

380 |endgroup

\PE@TestOctDigit

```

381 \def\PE@TestOctDigit#1{%
382     \ifnum'#1<48 % 0
383         \ltx@one
384     \else
385         \ifnum'#1>55 % 7
386         \ltx@one

```

```

387     \else
388         \ltx@zero
389     \fi
390 \fi
391 }

\PE@OctI
392 \def\PE@OctI#1#2{%
393     \ifcase\PE@TestOctDigit#2%
394         \def\PE@next{\PE@OctII{#1#2}}%
395     \else
396         \def\PE@next{\PE@OctAll{#1#2}}%
397     \fi
398     \PE@next
399 }

\PE@OctII
400 \def\PE@OctII#1#2{%
401     \ifcase\PE@TestOctDigit#2%
402         \def\PE@next{\PE@OctIII{#1#2}}%
403     \else
404         \def\PE@next{\PE@OctAll{#1}#2}%
405     \fi
406     \PE@next
407 }

408 \ltx@ifundefined{numexpr}{%
409     \catcode'\$=9 %
410     \catcode'\&=14 %
411 }{%
412     \catcode'\$=14 %
413     \catcode'\&=9 %
414 }

\PE@OctIII
415 \def\PE@OctIII#1#2#3{%
416     \ifnum#1<4 %
417         \def\PE@next{\PE@OctAll{#1#2#3}}%
418     \else
419     $ \count\ltx@ccclv#1 %
420     $ \advance\count\ltx@ccclv -4 %
421     \edef\PE@next{%
422         \noexpand\PE@OctAll{%
423     $ \the\count\ltx@ccclv
424     & \the\numexpr#1-4\relax
425         #2#3%
426     }%
427     }%
428     \fi
429     \PE@next
430 }

\PE@OctAll
431 \def\PE@OctAll#1{%
432     \uccode\ltx@zero=#1\relax
433     \uppercase{%
434         \edef\PE@result{\PE@result^^@}%
435     }%

```

```

436 \PE@DeString
437 }

```

## 2.7 User macros (pdfTeX analogues)

```

438 \begingroup\expandafter\expandafter\expandafter\endgroup
439 \expandafter\ifx\csname RequirePackage\endcsname\relax
440 \def\TMP@RequirePackage#1[#2]{%
441 \begingroup\expandafter\expandafter\expandafter\endgroup
442 \expandafter\ifx\csname ver@#1.sty\endcsname\relax
443 \input #1.sty\relax
444 \fi
445 }%
446 \TMP@RequirePackage{pdftexcmds}[2007/11/11]%
447 \else
448 \RequirePackage{pdftexcmds}[2007/11/11]%
449 \fi
450 \begingroup\expandafter\expandafter\expandafter\endgroup
451 \expandafter\ifx\csname pdf@escapehex\endcsname\relax

```

\EdefEscapeHex

```

452 \long\def\EdefEscapeHex#1#2{%
453 \EdefSanitize#1{#2}%
454 \PE@SanitizeSpaceOther#1%
455 \PE@EscapeHex#1%
456 }%

```

\EdefUnescapeHex

```

457 \def\EdefUnescapeHex#1#2{%
458 \EdefSanitize#1{#2}%
459 \PE@UnescapeHex#1%
460 }%

```

\EdefEscapeName

```

461 \long\def\EdefEscapeName#1#2{%
462 \EdefSanitize#1{#2}%
463 \PE@SanitizeSpaceOther#1%
464 \PE@EscapeName#1%
465 }%

```

\EdefEscapeString

```

466 \long\def\EdefEscapeString#1#2{%
467 \EdefSanitize#1{#2}%
468 \PE@SanitizeSpaceOther#1%
469 \PE@EscapeString#1%
470 }%

```

471 \else

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

```

472 \long\def\PE@edefbabel#1#2#3{%
473 \begingroup
474 \csname @save@activetrue\endcsname
475 \edef#1{#2{#3}}%
476 \expandafter\endgroup
477 \expandafter\def\expandafter#1\expandafter{#1}%
478 }%

```

`\EdefEscapeHex`

```
479 \long\def\EdefEscapeHex#1#2{%
480   \PE@edefbabel#1\pdf@escapehex{#2}%
481 }%
```

`\EdefUnescapeHex`

```
482 \def\EdefUnescapeHex#1#2{%
483   \PE@edefbabel#1\pdf@unescapehex{#2}%
484 }%
```

`\EdefEscapeName`

```
485 \long\def\EdefEscapeName#1#2{%
486   \PE@edefbabel#1\pdf@escapename{#2}%
487 }%
```

`\EdefEscapeString`

```
488 \long\def\EdefEscapeString#1#2{%
489   \PE@edefbabel#1\pdf@escapestring{#2}%
490 }%

491 \expandafter\PE@AtEnd
492 \fi%
```

## 2.8 Help macros

### 2.8.1 Characters

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

`\PE@hash`

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

`\PE@backslash`

```
494 \begingroup
495   \escapechar=-1 %
496 \edef\x{\endgroup
497   \def\noexpand\PE@backslash{\string\}}%
498 }
499 \x
```

### 2.8.2 Switch for $\varepsilon$ -TeX

```
500 \ltx@newif\ifPE@etex
501 \begingroup\expandafter\expandafter\expandafter\endgroup
502 \expandafter\ifx\csname numexpr\endcsname\relax
503 \else
504   \PE@etexttrue
505 \fi
```

## 2.9 Conversions

### 2.9.1 Conversion to hex string

`\PE@EscapeHex`

```
506 \ifPE@etex
507   \def\PE@EscapeHex#1{%
508     \edef#1{\expandafter\PE@ToHex#1\relax}%
509   }%
510 \else
```

```

511 \def\PE@EscapeHex#1{%
512 \def\PE@result{}%
513 \expandafter\PE@ToHex#1\relax
514 \let#1\PE@result
515 }%
516 \fi

```

\PE@ToHex

```

517 \def\PE@ToHex#1{%
518 \ifx\relax#1%
519 \else
520 \PE@HexChar{#1}%
521 \expandafter\PE@ToHex
522 \fi
523 }%

```

\PE@HexChar

```

524 \ifPE@etex
525 \def\PE@HexChar#1{%
526 \PE@HexDigit{\numexpr\dimexpr.0625\dimexpr'#1sp\relax\relax\relax}%
527 \PE@HexDigit{%
528 \numexpr'#1-16*\dimexpr.0625\dimexpr'#1sp\relax\relax\relax
529 }%
530 }%
531 \else
532 \def\PE@HexChar#1{%
533 \dimen0='#1sp%
534 \dimen2=.0625\dimen0 %
535 \advance\dimen0-16\dimen2 %
536 \edef\PE@result{%
537 \PE@result
538 \PE@HexDigit{\dimen2 }%
539 \PE@HexDigit{\dimen0 }%
540 }%
541 }%
542 \fi

```

\PE@HexDigit

```

543 \def\PE@HexDigit#1{%
544 \expandafter\string
545 \ifcase#1%
546 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or 9\or
547 A\or B\or C\or D\or E\or F%
548 \fi
549 }

```

## 2.9.2 Character code to octal number

\PE@OctChar

```

550 \ifPE@etex
551 \def\PE@OctChar#1{%
552 \expandafter\PE@@OctChar
553 \the\numexpr\dimexpr.015625\dimexpr'#1sp\relax\relax
554 \expandafter\relax
555 \expandafter\relax
556 \the\numexpr\dimexpr.125\dimexpr'#1sp\relax\relax\relax
557 \relax
558 #1%

```

```

559 }%
560 \def\PE@OctChar#1\relax#2\relax#3{%
561   \PE@backslash
562   #1%
563   \the\numexpr#2-8*#1\relax
564   \the\numexpr\dimexpr'#3sp\relax-8*#2\relax
565 }%
566 \else
567   \def\PE@OctChar#1{%
568     \dimen0='#1sp%
569     \dimen2=.125\dimen0 %
570     \dimen4=.125\dimen2 %
571     \advance\dimen0-8\dimen2 %
572     \advance\dimen2-8\dimen4 %
573     \edef\PE@result{%
574       \PE@result
575       \PE@backslash
576       \number\dimen4 %
577       \number\dimen2 %
578       \number\dimen0 %
579     }%
580 }%
581 \fi

```

### 2.9.3 Unpack hex string

\PE@UnescapeHex

```

582 \def\PE@UnescapeHex#1{%
583   \begingroup
584   \PE@InitUccodeHexDigit
585   \def\PE@result{}%
586   \expandafter\PE@DeHex#1\relax\relax
587   \expandafter\endgroup
588   \expandafter\def\expandafter#1\expandafter{\PE@result}%
589 }

```

\PE@DeHex

```

590 \def\PE@DeHex#1#2{%
591   \ifx#2\relax
592     \ifx#1\relax
593       \let\PE@next\relax
594     \else
595       \uppercase{%
596         \def\PE@testA{#1}%
597       }%
598       \ifcase\expandafter\PE@TestUcHexDigit\PE@testA
599         \def\PE@next{%
600           \PE@DeHex#10\relax\relax
601         }%
602       \else
603         \let\PE@next\relax
604       \fi
605     \fi
606   \else
607     \uppercase{%
608       \def\PE@testA{#1}%
609       \def\PE@testB{#2}%
610     }%

```

```

611 \ifcase\expandafter\PE@TestUcHexDigit\PE@testA
612 \ifcase\expandafter\PE@TestUcHexDigit\PE@testB
613 \uccode\ltx@zero="\PE@testA\PE@testB\relax
614 \ifnum\uccode\ltx@zero=32 %
615 \let\PE@temp\PE@space@space
616 \else
617 \uppercase{%
618 \def\PE@temp{^^@}%
619 }%
620 \fi
621 \edef\PE@result{\PE@result\PE@temp}%
622 \let\PE@next\PE@DeHex
623 \else
624 % invalid input sequence
625 \def\PE@next{%
626 \PE@DeHex#1%
627 }%
628 \fi
629 \else
630 % invalid input sequence
631 \def\PE@next{\PE@DeHex#2}%
632 \fi
633 \fi
634 \PE@next
635 }

```

#### 2.9.4 Conversion to PDF name

\PE@EscapeName

```

636 \ifPE@etex
637 \def\PE@EscapeName#1{%
638 \edef#1{\expandafter\PE@EscapeNameTokens#1\relax}%
639 }%
640 \else
641 \def\PE@EscapeName#1{%
642 \def\PE@result{}%
643 \expandafter\PE@EscapeNameTokens#1\relax
644 \let#1\PE@result
645 }%
646 \fi

```

\PE@EscapeNameTokens

```

647 \def\PE@EscapeNameTokens#1{%
648 \ifx\relax#1%
649 \else
650 \ifnum'#1<33 %
651 \ifcase'#1 %
652 % drop illegal zero
653 \else
654 \PE@EscapeNameAdd\PE@hash
655 \PE@HexChar#1%
656 \fi
657 \else
658 \ifnum'#1>126 %
659 \PE@EscapeNameAdd\PE@hash
660 \PE@HexChar#1%
661 \else \ifnum'#1=35 \PE@EscapeNameHashChar 23% #
662 \else\ifnum'#1=37 \PE@EscapeNameHashChar 25% %

```

```

663     \else\ifnum'#1=40 \PE@EscapeNameHashChar 28% (
664     \else\ifnum'#1=41 \PE@EscapeNameHashChar 29% )
665     \else\ifnum'#1=47 \PE@EscapeNameHashChar 2F% /
666     \else\ifnum'#1=60 \PE@EscapeNameHashChar 3C% <
667     \else\ifnum'#1=62 \PE@EscapeNameHashChar 3E% >
668     \else\ifnum'#1=91 \PE@EscapeNameHashChar 5B% [
669     \else\ifnum'#1=93 \PE@EscapeNameHashChar 5D% ]
670     \else\ifnum'#1=123 \PE@EscapeNameHashChar 7B% {
671     \else\ifnum'#1=125 \PE@EscapeNameHashChar 7D% }
672     \else
673         \PE@EscapeNameAdd{#1}%
674         \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
675     \fi
676     \fi
677     \expandafter\PE@EscapeNameTokens
678 \fi
679 }%
680 \def\PE@EscapeNameHashChar#1#2{%
681     \PE@EscapeNameAdd{\PE@hash\string#1\string#2}%
682 }%

```

\PE@EscapeNameAdd

```

683 \ifPE@etex
684     \def\PE@EscapeNameAdd#1{#1}%
685 \else
686     \def\PE@EscapeNameAdd#1{%
687         \edef\PE@result{%
688             \PE@result
689             #1%
690         }%
691     }%
692 \fi

```

## 2.9.5 Conversion to PDF string

\PE@EscapeString

```

693 \ifPE@etex
694     \def\PE@EscapeString#1{%
695         \edef#1{\expandafter\PE@EscapeStringTokens#1\relax}%
696     }%
697 \else
698     \def\PE@EscapeString#1{%
699         \begingroup
700             \def\PE@result{%
701                 \expandafter\PE@EscapeStringTokens#1\relax
702             }%
703         \expandafter\def\expandafter#1\expandafter{\PE@result}%
704     }%
705 \fi

```

\PE@EscapeStringTokens

```

706 \def\PE@EscapeStringTokens#1{%
707     \ifx\relax#1%
708     \else
709         \ifnum'#1<33 %
710             \PE@OctChar#1%
711         \else
712             \ifnum'#1>126 %

```

```

713     \PE@OctChar#1%
714     \else \ifnum'#1=40 \PE@EscapeStringAdd{\string\}% (
715     \else\ifnum'#1=41 \PE@EscapeStringAdd{\string\)}% )
716     \else\ifnum'#1=92 \PE@EscapeStringAdd{\string\\}% \
717     \else
718     \PE@EscapeStringAdd{#1}%
719     \fi\fi\fi
720     \fi
721     \fi
722     \expandafter\PE@EscapeStringTokens
723     \fi
724 }%

```

\PE@EscapeStringAdd

```

725 \ifPE@etex
726   \def\PE@EscapeStringAdd#1{#1}%
727 \else
728   \def\PE@EscapeStringAdd#1{%
729     \edef\PE@result{%
730       \PE@result
731       #1%
732     }%
733   }%
734 \fi

735 \PE@AtEnd%
736 </package>

```

## 3 Installation

### 3.1 Download

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

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

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

**Bundle.** All the packages of the bundle ‘pdfescape’ 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/pdfescape.tds.zip](#)

*TDS* refers to the standard “A Directory Structure for T<sub>E</sub>X Files” ([CTAN:pkg/tds](#)). Directories with `texmf` in their name are usually organized this way.

### 3.2 Bundle installation

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

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

<sup>1</sup>[CTAN:pkg/pdfescape](#)

### 3.3 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/pdfescape/pdfescape.sty
pdfescape.pdf → doc/latex/pdfescape/pdfescape.pdf
pdfescape.dtx → source/latex/pdfescape/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`.

### 3.4 Refresh file name databases

If your `TEX` distribution (`TEX Live`, `mikTEX`, ...) relies on file name databases, you must refresh these. For example, `TEX Live` users run `texhash` or `mktextlsr`.

### 3.5 Some details for the interested

**Unpacking with L<sup>A</sup>T<sub>E</sub>X.** The `.dtx` chooses its action depending on the format:

**plain T<sub>E</sub>X:** Run `docstrip` and extract the files.

**L<sup>A</sup>T<sub>E</sub>X:** Generate the documentation.

If you insist on using L<sup>A</sup>T<sub>E</sub>X for `docstrip` (really, `docstrip` does not need L<sup>A</sup>T<sub>E</sub>X), 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 `pdfLATEX`:

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

## 4 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`.

**[2007/11/11 v1.8]**

- Use of package `pdftexcmds` for Lua $\TeX$  support.

**[2010/03/01 v1.9]**

- Compatibility with `ini $\TeX$` .

**[2010/11/12 v1.10]**

- Use of package `ltxcmds`.
- Fix for compatibility with `ini $\TeX$` .

[2011/01/30 v1.11]

- Already loaded package files are not input in plain  $\text{\TeX}$ .

[2011/04/04 v1.12]

- Further fixes for compatibility for  $\text{\iniTeX}$ .
- Test file for  $\text{\iniTeX}$  added.

[2011/11/25 v1.13]

- Higher order bit of octal sequences in  $\text{\EdefUnescapeString}$  ignored according to the PDF specification (Bug found by Bruno Le Floch).

[2016/05/16 v1.14]

- Documentation updates.

[2019/12/09 v1.15]

- Documentation updates.

## 5 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.

	<b>Symbols</b>		80, 81, 82, 83, 84, 87, 88, 90, 91,
$\#$	191, 327, 352		92, 93, 101, 102, 103, 104, 105,
$\$$	190, 409, 412		106, 107, 108, 109, 110, 111,
$\&$	410, 413		112, 113, 114, 115, 167, 190,
$\langle$	714		191, 318, 319, 409, 410, 412, 413
$\rangle$	715	$\backslash\text{count}$	419, 420, 423
$\@onelevel@sanitize$	149	$\backslash\text{csname}$	14, 21,
$\@undefined$	58		50, 66, 76, 121, 124, 133, 136,
$\backslash\backslash$	175, 319, 497, 716		143, 154, 439, 442, 451, 474, 502
$\backslash}$	376		
$\backslash $	318, 323		
	<b>Numbers</b>		<b>D</b>
$\backslash 8$	291	$\backslash\text{detokenize}$	158, 162
$\backslash 9$	292	$\backslash\text{dimen}$	533, 534, 535, 538, 539, 568,
			569, 570, 571, 572, 576, 577, 578
$\backslash \square$	167, 716	$\backslash\text{dimexpr}$	526, 528, 553, 556, 564
	<b>A</b>		<b>E</b>
$\backslash\text{advance}$	420, 535, 571, 572	$\backslash\text{EdefEscapeHex}$	2, 452, 479
$\backslash\text{aftergroup}$	29	$\backslash\text{EdefEscapeName}$	461, 485
	<b>C</b>	$\backslash\text{EdefEscapeString}$	466, 488
$\backslash\text{catcode}$	2, 3, 5, 6, 7, 8, 9,	$\backslash\text{EdefSanitize}$	3, 132,
	10, 11, 12, 13, 33, 34, 36, 37, 38,		165, 184, 284, 453, 458, 462, 467
	39, 40, 41, 42, 43, 44, 45, 46, 47,	$\backslash\text{EdefUnescapeHex}$	457, 482
	48, 49, 69, 70, 72, 73, 74, 78, 79,	$\backslash\text{EdefUnescapeName}$	2, 183
		$\backslash\text{EdefUnescapeString}$	3, 283
		$\backslash\text{empty}$	17, 18

<code>\endcsname</code>	14, 21, 50, 66, 76, 121, 124, 133, 136, 143, 154, 439, 442, 451, 474, 502	<code>\PE@edefbabel</code>	472, 480, 483, 486, 489
<code>\endinput</code>	29, 119	<code>\PE@EscapeHex</code>	455, 506
<code>\endlinechar</code>	4, 35, 71, 77, 89	<code>\PE@EscapeName</code>	464, 636
<code>\escapechar</code>	495	<code>\PE@EscapeNameAdd</code>	654, 659, 673, 681, 683
<b>G</b>			
<code>\gdef</code>	192, 200	<code>\PE@EscapeNameHashChar</code>	661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 680
<b>I</b>			
<code>\ifcase</code>	214, 215, 393, 401, 545, 598, 611, 612, 651	<code>\PE@EscapeNameTokens</code>	638, 643, 647
<code>\ifnum</code>	265, 268, 271, 272, 382, 385, 416, 614, 650, 658, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 709, 712, 714, 715, 716	<code>\PE@EscapeString</code>	469, 693
<code>\ifPE@etex</code>	500, 506, 524, 550, 636, 683, 693, 725	<code>\PE@EscapeStringAdd</code>	714, 715, 716, 718, 725
<code>\ifx</code>	15, 18, 21, 50, 58, 61, 121, 124, 133, 143, 175, 201, 205, 300, 305, 439, 442, 451, 502, 518, 591, 592, 648, 707	<code>\PE@EscapeStringTokens</code>	695, 701, 706
<code>\immediate</code>	23, 52	<code>\PE@etexttrue</code>	504
<code>\input</code>	125, 443	<code>\PE@hash</code>	493, 654, 659, 681
<b>L</b>			
<code>\ltx@ccclv</code>	419, 420, 423	<code>\PE@HexChar</code>	520, 524, 655, 660
<code>\ltx@ifUndefined</code>	408	<code>\PE@HexDigit</code>	526, 527, 538, 539, 543
<code>\ltx@newif</code>	500	<code>\PE@InitUccodeHexDigit</code>	194, 240, 584
<code>\ltx@one</code>	218, 221, 266, 269, 273, 383, 386	<code>\PE@next</code>	203, 208, 229, 233, 237, 302, 306, 308, 311, 394, 396, 398, 402, 404, 406, 417, 421, 429, 593, 599, 603, 622, 625, 631, 634
<code>\ltx@ReturnAfterFi</code>	178	<code>\PE@NormalizeLineEnd</code>	286, 294
<code>\ltx@zero</code>	216, 223, 227, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 275, 278, 388, 432, 613, 614	<code>\PE@OctAll</code>	396, 404, 417, 422, 431
<b>M</b>			
<code>\meaning</code>	145	<code>\PE@OctChar</code>	550, 710, 713
<b>N</b>			
<code>\number</code>	576, 577, 578	<code>\PE@OctI</code>	392
<code>\numexpr</code>	424, 526, 528, 553, 556, 563, 564	<code>\PE@OctII</code>	394, 400
<b>P</b>			
<code>\PackageInfo</code>	26	<code>\PE@OctIII</code>	402, 415
<code>\pdf@escapehex</code>	480	<code>\PE@onelevel@sanitize</code>	138, 144, 149, 161, 187, 288
<code>\pdf@escapename</code>	486	<code>\PE@result</code>	195, 198, 202, 207, 228, 232, 295, 297, 301, 304, 434, 512, 514, 536, 537, 573, 574, 585, 588, 621, 642, 644, 687, 688, 700, 703, 729, 730
<code>\pdf@escapestring</code>	489	<code>\PE@sanitize</code>	165
<code>\pdf@unescapehex</code>	483	<code>\PE@SanitizeSpaceOther</code>	170, 185, 285, 454, 463, 468
<code>\PE@@NormalizeLineEnd</code>	296, 299	<code>\PE@space@other</code>	166, 177
<code>\PE@@OctChar</code>	552, 560	<code>\PE@space@space</code>	169, 615
<code>\PE@AtEnd</code>	95, 96, 119, 491, 735	<code>\PE@SpaceToOther</code>	171, 173
<code>\PE@backslash</code>	494, 561, 575	<code>\PE@strip@prefix</code>	145, 147
<code>\PE@CheckEndBackslash</code>	373	<code>\PE@temp</code>	225, 228, 615, 618, 621
<code>\PE@DeHex</code>	586, 590	<code>\PE@testA</code>	211, 214, 223, 596, 598, 608, 611, 613
<code>\PE@DeName</code>	196, 200, 229, 233	<code>\PE@testB</code>	212, 215, 223, 609, 612, 613
<code>\PE@DeString</code>	327, 436	<code>\PE@TestOctDigit</code>	381, 393, 401
		<code>\PE@TestUcHexDigit</code>	214, 215, 264, 598, 611, 612
		<code>\PE@ToHex</code>	508, 513, 517
		<code>\PE@UnescapeHex</code>	459, 582
		<code>\PE@UnescapeName</code>	186, 189
		<code>\PE@UnescapeString</code>	287, 320
		<code>\ProvidesPackage</code>	19, 67
<b>R</b>			
<code>\RequirePackage</code>	130, 448		

	<b>T</b>		
<code>\the</code> . . . . .	77, 78, 79, 80, 81, 82, 83, 84, 97, 423, 424, 553, 556, 563, 564		249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 291, 292, 432, 613, 614
<code>\TMP@EnsureCode</code> . . . . .	. . . 94, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118	<code>\uppercase</code> . . . . .	. . . 210, 224, 314, 433, 595, 607, 617
<code>\TMP@RequirePackage</code>	122, 128, 440, 446	<b>W</b>	
		<code>\write</code> . . . . .	23, 52
	<b>U</b>		
<code>\ucode</code> . . . . .	116, 117, 118, 223, 227, 241, 242, 243, 244, 245, 246, 247, 248,	<b>X</b>	
		<code>\x</code> . . . . .	14, 15, 18, 22, 26, 28, 51, 56, 66, 75, 87, 168, 293, 315, 496, 499