

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>	6
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\@undefined
54       \xdef#1{#3}%
55     \fi
56     \ifx#1\relax
57       \xdef#1{#3}%
58     \fi
59   }%
60 \fi
61 \expandafter\x\csname ver@pdfescape.sty\endcsname
62 \ProvidesPackage{pdfescape}%
63 [2007/11/11 v1.8 Provides hex, PDF name and string conversions (HO)]

```

2.2 Catcodes

```

64 \expandafter\edef\csname PE@AtEnd\endcsname{%
65   \catcode64 \the\catcode64\relax
66 }
67 \catcode64 11 % @
68 \def\TMP@EnsureCode#1#2#3{%
69   \edef\PE@AtEnd{%
70     \PE@AtEnd
71     #1#2 \the#1#2\relax
72   }%
73   #1#2 #3\relax
74 }
75 \TMP@EnsureCode\catcode{0}{12}% ^^@
76 \TMP@EnsureCode\catcode{34}{12}% "
77 \TMP@EnsureCode\catcode{39}{12}% '
78 \TMP@EnsureCode\catcode{42}{12}% *
79 \TMP@EnsureCode\catcode{45}{12}% -
80 \TMP@EnsureCode\catcode{46}{12}% .
81 \TMP@EnsureCode\catcode{47}{12}% /
82 \TMP@EnsureCode\catcode{60}{12}% <
83 \TMP@EnsureCode\catcode{61}{12}% =
84 \TMP@EnsureCode\catcode{62}{12}% >
85 \TMP@EnsureCode\catcode{94}{7}% ^
86 \TMP@EnsureCode\catcode{96}{12}% `
87 \TMP@EnsureCode\uccode{34}{0}% "
88 \TMP@EnsureCode\uccode{48}{0}% 0
89 \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.

```

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

```

```

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

```

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

```

123 \let\PE@sanitize\EdefSanitize

```

2.3.1 Space characters

`\PE@space@other`

```

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

```

`\PE@space@space`

```

127 \def\PE@space@space{ }

```

2.3.2 Space normalization

`\PE@SanitizeSpaceOther`

```

128 \def\PE@SanitizeSpaceOther#1{%
129   \edef#1{\expandafter\PE@SpaceToOther#1 \relax}%
130 }

```

`\PE@SpaceToOther`

```

131 \def\PE@SpaceToOther#1 #2\relax{%
132   #1%
133   \ifx\#2\%
134   \else
135     \PE@space@other
136     \@ReturnAfterFi{%
137       \PE@SpaceToOther#2\relax
138     }%
139   \fi
140 }

```

`\@ReturnAfterFi`

```

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

```

2.4 \EdefUnescapeName

\EdefUnescapeName

```
142 \def\EdefUnescapeName#1#2{%
143   \EdefSanitize#1{#2}%
144   \PE@SanitizeSpaceOther#1%
145   \PE@UnescapeName#1%
146   \PE@onelevel@sanitize#1%
147 }
```

\PE@UnescapeName

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

\PE@InitUccodeHexDigit

```
199 \def\PE@InitUccodeHexDigit{%
200   \uccode'a='A\relax
201   \uccode'b='B\relax
202   \uccode'c='C\relax
203   \uccode'd='D\relax
204   \uccode'e='E\relax
205   \uccode'f='F\relax
206   \uccode'A=\z@
207   \uccode'B=\z@
208   \uccode'C=\z@
209   \uccode'D=\z@
210   \uccode'E=\z@
211   \uccode'F=\z@
212   \uccode'0=\z@
213   \uccode'1=\z@
214   \uccode'2=\z@
215   \uccode'3=\z@
216   \uccode'4=\z@
217   \uccode'5=\z@
218   \uccode'6=\z@
219   \uccode'7=\z@
220   \uccode'8=\z@
221   \uccode'9=\z@
222 }
```

\PE@TestUcHexDigit

```
223 \def\PE@TestUcHexDigit#1{%
224   \ifnum'#1<48 % 0
225     \@ne
226   \else
227     \ifnum'#1>70 % F
228       \@ne
229     \else
230       \ifnum'#1>57 % 9
231         \ifnum'#1<65 % A
232           \@ne
233         \else
234           \z@
235         \fi
236       \else
237         \z@
238       \fi
239     \fi
240   \fi
241 }
```

2.5 \EdefUnescapeString

\EdefUnescapeString

```
242 \def\EdefUnescapeString#1#2{%
243   \EdefSanitize#1{#2}%
244   \PE@SanitizeSpaceOther#1%
245   \PE@NormalizeLineEnd#1%
246   \PE@UnescapeString#1%
247   \PE@onelevel@sanitize#1%
248 }

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

\PE@NormalizeLineEnd

```
253 \def\PE@NormalizeLineEnd##1{%
254   \def\PE@result{}%
255   \expandafter\PE@@NormalizeLineEnd##1#2\relax
256   \let##1\PE@result
257 }%
```

\PE@@NormalizeLineEnd

```
258 \def\PE@@NormalizeLineEnd##1#2##2{%
259   \ifx\relax##2%
260     \edef\PE@result{\PE@result##1}%
261     \let\PE@next\relax
262   \else
263     \edef\PE@result{\PE@result##1#1}%
264     \ifx#1##2% \if
265       \let\PE@next\PE@@NormalizeLineEnd
266     \else
267       \def\PE@next{\PE@@NormalizeLineEnd##2}%
268     \fi
269   \fi
270   \PE@next
271 }%
272 }%
273 \uppercase{%
274   \x 89%
275 }

276 \begingroup
277 \catcode'\|=0 %
278 \catcode'\|=12 %
```

\PE@UnescapeString

```
279 |gdef|PE@UnescapeString#1{%
280   |begingroup
281   |def|PE@result{}%
282   |expandafter|PE@DeString#1|\relax
283   |expandafter|endgroup
284   |expandafter|def|expandafter#1|expandafter{|PE@result}%
285 }%
```

\PE@DeString

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



```

307     }%
308     |edef|PE@result{|PE@result#1|PE@temp}%
309     |let|PE@next|PE@DeString
310   |else
311     |if|#2% backslash
312       |edef|PE@result{|PE@result#1}%
313       |let|PE@next|PE@CheckEndBackslash
314     |else
315       |ifnum'#2=10 % linefeed
316         |edef|PE@result{|PE@result#1}%
317         |let|PE@next|PE@DeString
318       |else
319         |ifcase|PE@TestOctDigit#2%
320           |edef|PE@result{|PE@result#1}%
321           |def|PE@next{|PE@OctI#2}%
322         |else
323           |edef|PE@result{|PE@result#1#2}%
324           |let|PE@next|PE@DeString
325         |fi
326       |fi
327     |fi
328   |fi
329 |fi
330 |PE@next
331 }%

```

\PE@CheckEndBackslash

```

332 |gdef|PE@CheckEndBackslash#1{%
333   |ifx|relax#1%
334   |else
335     |edef|PE@result{|PE@result\}%
336     |expandafter|PE@DeString|expandafter#1%
337   |fi
338 }%
339 |endgroup

```

\PE@TestOctDigit

```

340 \def\PE@TestOctDigit#1{%
341   \ifnum'#1<48 % 0
342     \@ne
343   \else
344     \ifnum'#1>55 % 7
345       \@ne
346     \else
347       \z@
348     \fi
349   \fi
350 }

```

\PE@OctI

```

351 \def\PE@OctI#1#2{%
352   \ifcase\PE@TestOctDigit#2%
353     \def\PE@next{\PE@OctII{#1#2}}%
354   \else
355     \def\PE@next{\PE@OctAll#1#2}%
356   \fi
357   \PE@next
358 }

```

\PE@OctII

```

359 \def\PE@OctII#1#2{%

```

```

360 \ifcase\PE@TestOctDigit#2%
361   \def\PE@next{\PE@OctAll{#1#2}}%
362 \else
363   \def\PE@next{\PE@OctAll{#1}#2}%
364 \fi
365 \PE@next
366 }

```

\PE@OctAll

```

367 \def\PE@OctAll#1{%
368   \uccode\z@='#1\relax
369   \uppercase{%
370     \edef\PE@result{\PE@result^^0}%
371   }%
372   \PE@DeString
373 }

```

2.6 User macros (pdfTeX analogues)

```

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

380 \begingroup\expandafter\expandafter\expandafter\endgroup
381 \expandafter\ifx\csname pdf@escapehex\endcsname\relax

```

\EdefEscapeHex

```

382 \long\def\EdefEscapeHex#1#2{%
383   \EdefSanitize#1{#2}%
384   \PE@SanitizeSpaceOther#1%
385   \PE@EscapeHex#1%
386 }%

```

\EdefUnescapeHex

```

387 \def\EdefUnescapeHex#1#2{%
388   \EdefSanitize#1{#2}%
389   \PE@UnescapeHex#1%
390 }%

```

\EdefEscapeName

```

391 \long\def\EdefEscapeName#1#2{%
392   \EdefSanitize#1{#2}%
393   \PE@SanitizeSpaceOther#1%
394   \PE@EscapeName#1%
395 }%

```

\EdefEscapeString

```

396 \long\def\EdefEscapeString#1#2{%
397   \EdefSanitize#1{#2}%
398   \PE@SanitizeSpaceOther#1%
399   \PE@EscapeString#1%
400 }%

```

```

401 \else

```

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

```

402 \long\def\PE@edefbabel#1#2#3{%
403   \begingroup
404     \csname @save@activetrue\endcsname

```

```

405     \edef#1{#2{#3}}}%
406     \expandafter\endgroup
407     \expandafter\def\expandafter#1\expandafter{#1}%
408 }%

```

\EdefEscapeHex

```

409 \long\def\EdefEscapeHex#1#2{%
410     \PE@edefbabel#1\pdf@escapehex{#2}%
411 }%

```

\EdefUnescapeHex

```

412 \def\EdefUnescapeHex#1#2{%
413     \PE@edefbabel#1\pdf@unescapehex{#2}%
414 }%

```

\EdefEscapeName

```

415 \long\def\EdefEscapeName#1#2{%
416     \PE@edefbabel#1\pdf@escapename{#2}%
417 }%

```

\EdefEscapeString

```

418 \long\def\EdefEscapeString#1#2{%
419     \PE@edefbabel#1\pdf@escapestring{#2}%
420 }%

421 \PE@AtEnd
422 \expandafter\endinput
423 \fi

```

2.7 Help macros

2.7.1 Characters

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

\PE@hash

```

424 \edef\PE@hash{\string#}

```

\PE@backslash

```

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

```

2.7.2 Switch for ε -T_EX

```

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

```

2.8 Conversions

2.8.1 Conversion to hex string

\PE@EscapeHex

```

437 \ifPE@etex
438     \def\PE@EscapeHex#1{%

```

```

439 \edef#1{\expandafter\PE@ToHex#1\relax}%
440 }%
441 \else
442 \def\PE@EscapeHex#1{%
443 \def\PE@result{%
444 \expandafter\PE@ToHex#1\relax
445 \let#1\PE@result
446 }%
447 \fi

\PE@ToHex
448 \def\PE@ToHex#1{%
449 \ifx\relax#1%
450 \else
451 \PE@HexChar{#1}%
452 \expandafter\PE@ToHex
453 \fi
454 }%

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

\PE@HexDigit
474 \def\PE@HexDigit#1{%
475 \expandafter\string
476 \ifcase#1%
477 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or 9\or
478 A\or B\or C\or D\or E\or F%
479 \fi
480 }

```

2.8.2 Character code to octal number

```

\PE@OctChar
481 \ifPE@etex
482 \def\PE@OctChar#1{%
483 \expandafter\PE@@OctChar
484 \the\numexpr\dimexpr.015625\dimexpr'#1sp\relax\relax
485 \expandafter\relax
486 \expandafter\relax
487 \the\numexpr\dimexpr.125\dimexpr'#1sp\relax\relax\relax
488 \relax
489 #1%

```

```

490 }%
491 \def\PE@OctChar#1\relax#2\relax#3{%
492   \PE@backslash
493   #1%
494   \the\numexpr#2-8*#1\relax
495   \the\numexpr\dimexpr'#3sp\relax-8*#2\relax
496 }%
497 \else
498   \def\PE@OctChar#1{%
499     \dimen0='#1sp%
500     \dimen2=.125\dimen0 %
501     \dimen4=.125\dimen2 %
502     \advance\dimen0-8\dimen2 %
503     \advance\dimen2-8\dimen4 %
504     \edef\PE@result{%
505       \PE@result
506       \PE@backslash
507       \number\dimen4 %
508       \number\dimen2 %
509       \number\dimen0 %
510     }%
511   }%
512 \fi

```

2.8.3 Unpack hex string

\PE@UnescapeHex

```

513 \def\PE@UnescapeHex#1{%
514   \begingroup
515   \PE@InitUccodeHexDigit
516   \def\PE@result{}%
517   \expandafter\PE@DeHex#1\relax\relax
518   \expandafter\endgroup
519   \expandafter\def\expandafter#1\expandafter{\PE@result}%
520 }

```

\PE@DeHex

```

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

```

```

546         \let\PE@temp\PE@space@space
547     \else
548         \uppercase{%
549             \def\PE@temp{^^@}%
550         }%
551     \fi
552     \edef\PE@result{\PE@result\PE@temp}%
553     \let\PE@next\PE@DeHex
554 \else
555     % invalid input sequence
556     \def\PE@next{%
557         \PE@DeHex#1%
558     }%
559 \fi
560 \else
561     % invalid input sequence
562     \def\PE@next{\PE@DeHex#2}%
563 \fi
564 \fi
565 \PE@next
566 }

```

2.8.4 Conversion to PDF name

\PE@EscapeName

```

567 \ifPE@etex
568   \def\PE@EscapeName#1{%
569     \edef#1{\expandafter\PE@EscapeNameTokens#1\relax}%
570   }%
571 \else
572   \def\PE@EscapeName#1{%
573     \def\PE@result{%
574       \expandafter\PE@EscapeNameTokens#1\relax
575       \let#1\PE@result
576     }%
577   \fi

```

\PE@EscapeNameTokens

```

578 \def\PE@EscapeNameTokens#1{%
579   \ifx\relax#1%
580   \else
581     \ifnum'#1<33 %
582       \ifcase'#1 %
583         % drop illegal zero
584       \else
585         \PE@EscapeNameAdd\PE@hash
586         \PE@HexChar#1%
587       \fi
588     \else
589       \ifnum'#1>126 %
590         \PE@EscapeNameAdd\PE@hash
591         \PE@HexChar#1%
592       \else \ifnum'#1=35 \PE@EscapeNameHashChar 23% #
593         \else\ifnum'#1=37 \PE@EscapeNameHashChar 25% %
594         \else\ifnum'#1=40 \PE@EscapeNameHashChar 28% (
595         \else\ifnum'#1=41 \PE@EscapeNameHashChar 29% )
596         \else\ifnum'#1=47 \PE@EscapeNameHashChar 2F% /
597         \else\ifnum'#1=60 \PE@EscapeNameHashChar 3C% <
598         \else\ifnum'#1=62 \PE@EscapeNameHashChar 3E% >
599         \else\ifnum'#1=91 \PE@EscapeNameHashChar 5B% [
600         \else\ifnum'#1=93 \PE@EscapeNameHashChar 5D% ]
601         \else\ifnum'#1=123 \PE@EscapeNameHashChar 7B% {

```

```

602         \else\ifnum'#1=125 \PE@EscapeNameHashChar 7D% }
603         \else
604             \PE@EscapeNameAdd{#1}%
605         \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
606     \fi
607 \fi
608 \expandafter\PE@EscapeNameTokens
609 \fi
610 }%
611 \def\PE@EscapeNameHashChar#1#2{%
612     \PE@EscapeNameAdd{\PE@hash\string#1\string#2}%
613 }%

```

\PE@EscapeNameAdd

```

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

```

2.8.5 Conversion to PDF string

\PE@EscapeString

```

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

```

\PE@EscapeStringTokens

```

637 \def\PE@EscapeStringTokens#1{%
638     \ifx\relax#1%
639     \else
640         \ifnum'#1<33 %
641             \PE@OctChar#1%
642         \else
643             \ifnum'#1>126 %
644                 \PE@OctChar#1%
645             \else \ifnum'#1=40 \PE@EscapeStringAdd{\string\}% (
646                 \else\ifnum'#1=41 \PE@EscapeStringAdd{\string\)}% )
647                 \else\ifnum'#1=92 \PE@EscapeStringAdd{\string\\}% \
648                 \else
649                     \PE@EscapeStringAdd{#1}%
650                 \fi\fi\fi
651             \fi
652         \fi
653     \expandafter\PE@EscapeStringTokens
654 \fi
655 }%

```

\PE@EscapeStringAdd

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

666 \PE@AtEnd
667 \</package>
```

3 Test

3.1 Catcode checks for loading

```
668 \<test1>

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



```

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

```

3.2 Macro tests

```

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

```

3.3 Test with \pdfescape... commands

```

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

```

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

```

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

```

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

```

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

```

3.6 Test with LuaTeX

```

761 <*test5>
762 \ProvidesFile{pdfescape-test5.tex}%
763     [2007/11/11 v1.8 Test with LuaTeX]%
764 </test5>

```

3.7 Check/ensure test preconditions

3.7.1 Check \pdfescape...

```

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

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

```

3.7.2 Check ε -TeX

```

779 <*test3>
780 \@ifundefined{numexpr}{%
781     \PackageError{pdfescape-test3}{%
782         Missing \eTeX
783     }{Test aborted.}%
784     \stop
785 }{}
786 </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

```

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

```

3.8 Common part

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

```

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

```

```

808 \Expect*{\ifx#1#2true\else false\fi}{true}%
809 }
810
811 \makeatletter
812 \begingroup
813 \gdef\AllBytes{%
814 \count@=0 %
815 \catcode@=12 %
816 \@whilenum\count@<256 \do{%
817 \lccode@=\count@
818 \ifnum\count@=32 %
819 \xdef\AllBytes{\AllBytes\space}%
820 \else
821 \lowercase{%
822 \xdef\AllBytes{\AllBytes^^@}%
823 }%
824 \fi
825 \advance\count@ by 1 %
826 }%
827 \endgroup
828 \newcommand*{\AllBytesHex}{%
829 000102030405060708090A0B0C0D0E0F%
830 101112131415161718191A1B1C1D1E1F%
831 202122232425262728292A2B2C2D2E2F%
832 303132333435363738393A3B3C3D3E3F%
833 404142434445464748494A4B4C4D4E4F%
834 505152535455565758595A5B5C5D5E5F%
835 606162636465666768696A6B6C6D6E6F%
836 707172737475767778797A7B7C7D7E7F%
837 808182838485868788898A8B8C8D8E8F%
838 909192939495969798999A9B9C9D9E9F%
839 A0A1A2A3A4A5A6A7A8A9AABACADAFAF%
840 B0B1B2B3B4B5B6B7B8B9BABBBBCDBEBF%
841 C0C1C2C3C4C5C6C7C8C9CACBCCDCECF%
842 D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
843 E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
844 F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF%
845 }
846 \@onelevel@sanitize\AllBytesHex
847 \expandafter\lowercase\expandafter{%
848 \expandafter\newcommand\expandafter*\expandafter\AllBytesHexLC
849 \expandafter{\AllBytesHex}%
850 }
851 \newcommand*{\AllBytesName}{%
852 \begingroup
853 \catcode'\# =12 %
854 \xdef\AllBytesName{%
855 #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
856 #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
857 #20!"#23$#25&'#28#29*+,-.#2F%
858 0123456789:;#3C=#3E?%
859 @ABCDEFGHIJKLMNO%
860 PQRSTUVWXYZ#5B\@backslashchar#5D^_%
861 'abcdefghijklmnopqrstuvwxyz%
862 pqrstuvwxyz#7B|#7D\string~#7F%
863 #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
864 #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
865 #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
866 #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
867 #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
868 #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
869 #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%

```

```

870      #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
871    }%
872 \endgroup
873 \@onelevel@sanitize\AllBytesName
874
875 \newcommand*{\AllBytesString}{%
876 \begin{group}
877   \def\{|}%
878   \edef\%{\@percentchar}%
879   \catcode'\|=0 %
880   \catcode'\#=12 %
881   \catcode'\~=12 %
882   \catcode'\|=12 %
883   \xdef\AllBytesString{%
884     \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
885     \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
886     \040!"#$%&'(\)*+,-./%
887     0123456789:;<=>?%
888     @ABCDEFGHIJKLMNO%
889     PQRSTUVWXYZ[\]^_%
890     'abcdefghijklmno%
891     pqrstuvwxyz{|}~\177%
892     \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
893     \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
894     \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
895     \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
896     \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
897     \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
898     \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
899     \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
900   }%
901 \endgroup
902 \@onelevel@sanitize\AllBytesString
903
904 <*test4>
905 \let\org@detokenize\detokenize
906 \let\detokenize\@undefined
907 \let\org@numexpr\numexpr
908 \let\numexpr\@undefined
909 </test4>
910 \RequirePackage{pdfescape}
911 <*test4>
912 \let\detokenize\org@detokenize
913 \let\numexpr\org@numexpr
914 </test4>
915
916 \begin{qstest}{all-hex}{\AllBytes, escapehex}
917   \EdefEscapeHex\x{\AllBytes}%
918   \Expect*{\x}*{\AllBytesHex}%
919   \ExpectVar\x\AllBytesHex
920 \end{qstest}
921
922 \begin{qstest}{all-unhex}{\AllBytesHex, unescapehex}
923   \EdefUnescapeHex\x{\AllBytesHex}%
924   \Expect*{\x}*{\AllBytes}%
925   \ExpectVar\x\AllBytes
926 \end{qstest}
927
928 \begin{qstest}{all-unhex-lc}{\AllBytesHexLC, unescapehex, lowercase}
929   \EdefUnescapeHex\x{\AllBytesHexLC}%
930   \Expect*{\x}*{\AllBytes}%
931   \ExpectVar\x\AllBytes

```

```

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

```

```

994 \end{qstest}
995
996 \begin{qstest}{all-string}{\AllBytes, escapestring}
997   \EdefEscapeString\x{\AllBytes}%
998   \Expect*{\x}*{\AllBytesString}%
999   \ExpectVar\x\AllBytesString
1000 \end{qstest}
1001
1002 \begin{qstest}{uchexdigit}{unescape, uppercase hex digit}
1003   \catcode'\@=11 %
1004   \catcode0=12 %
1005   \def\test#1#2{%
1006     \uccode0=#1\relax
1007     \uppercase{%
1008       \def\x{^^@}%
1009     }%
1010     \Expect*{%
1011       \ifcase\expandafter\PE@TestUcHexDigit\x
1012         true%
1013       \else
1014         false%
1015       \fi
1016     }{#2}%
1017   }%
1018   \def\range#1#2#3{%
1019     \count0=#1\relax
1020     \loop
1021     \ifnum\count0<#2\relax
1022       \test{\count0}{#3}%
1023       \advance\count0 by 1 %
1024     \repeat
1025   }%
1026   \range{0}{47}{false}%
1027   \range{48}{57}{true}%
1028   \range{58}{64}{false}%
1029   \range{65}{70}{true}%
1030   \range{71}{255}{false}%
1031 \end{qstest}
1032
1033 \begin{qstest}{unescapename}{unescapename}
1034   \def\test#1#2{%
1035     \EdefUnescapeName\x{#1}%
1036     \edef\y{#2}%
1037     \@onelevel@sanitize\y
1038     \ExpectVar\x\y
1039   }%
1040   \catcode'\#=12 %
1041   \catcode0=12 %
1042   \test{}{}%
1043   \test{x}{x}%
1044   \test{xy}{xy}%
1045   \test{#}{#}%
1046   \test{##}{##}%
1047   \test{###}{###}%
1048   \test{####}{####}%
1049   \test{#x}{#x}%
1050   \test{#xy}{#xy}%
1051   \test{#1}{#1}%
1052   \test{#40}{@}%
1053   \test{#400}{@}%
1054   \test{#4x0}{#4x0}%
1055   \test{#ab}{^^ab}%

```

```

1056 \test{#00}{^~@}%
1057 \test{x#40y#40z}{x@y@z}%
1058 \test{#40#40#40#40}{@@@}%
1059 \test{a#x}{a#x}%
1060 \test{a#xy}{a#xy}%
1061 \test{a#1}{a#1}%
1062 \test{a#40}{a@}%
1063 \test{a#400}{a@0}%
1064 \test{#20}{ }%
1065 \test{a#20}{a }%
1066 \test{a#20b}{a b}%
1067 \test{a#20#20#20b}{a \space\space b}%
1068 \end{qstest}
1069
1070 \begin{qstest}{unescapestring}{unescapestring}
1071 \def\test#1#2{%
1072   \EdefUnescapeString\x{#1}%
1073   \edef\y{#2}%
1074   \@onelevel@sanitize\y
1075   \ExpectVar\x\y
1076 }%
1077 \catcode0=12 %
1078 \def\DefChar#1#2{%
1079   \begingroup
1080     \uccode0=#2\relax
1081     \uppercase{\endgroup
1082       \def#1{^~@}%
1083     }%
1084 }%
1085 \DefChar\nul{0}%
1086 \DefChar\one{1}%
1087 \DefChar\bel{8}%
1088 \DefChar\tab{9}%
1089 \DefChar\lf{10}%
1090 \DefChar\ff{12}%
1091 \DefChar\cr{13}%
1092 \DefChar\{92}%
1093 \test{}{}%
1094 \test{a}{a}%
1095 \test{\}{}%
1096 \test{\\\\}{\\}%
1097 \test{\\\\y}{\\y}%
1098 \test{\\000}{\nul}%
1099 \test{\\b}{\bel}%
1100 \test{\\t}{\tab}%
1101 \test{\\n}{\lf}%
1102 \test{\\f}{\ff}%
1103 \test{\\r}{\cr}%
1104 \test{\\({}{)%
1105 \test{\\)}{)%
1106 \test{\\040}{ }%
1107 \test{\\100}{@}%
1108 \test{\\40}{ }%
1109 \test{\\1}{\one}%
1110 \test{\\01}{\one}%
1111 \test{\\001}{\one}%
1112 \test{\\18}{\one8}%
1113 \test{\\018}{\one8}%
1114 \test{\\0018}{\one8}%
1115 \test{x\\}{x}%
1116 \test{x\\\\}{x\\}%
1117 \test{x\\\\y}{x\\y}%

```

```

1118 \test{x\\000}{x\nul}%
1119 \test{x\\b}{x\bel}%
1120 \test{x\\t}{x\tab}%
1121 \test{x\\n}{x\lf}%
1122 \test{x\\f}{x\ff}%
1123 \test{x\\r}{x\cr}%
1124 \test{x\\()}{x{}}%
1125 \test{x\\})}{x}%
1126 \test{x\\040}{x }%
1127 \test{x\\100}{x@}%
1128 \test{x\\40}{x }%
1129 \test{x\\1}{x\one}%
1130 \test{x\\01}{x\one}%
1131 \test{x\\001}{x\one}%
1132 \test{x\\18}{x\one8}%
1133 \test{x\\018}{x\one8}%
1134 \test{x\\0018}{x\one8}%
1135 \test{\\b\\t\\n\\f\\r\\(\\)\\\\\\\\000\\040}{%
1136 \bel\tab\lf\ff\cr()\\nul\space
1137 }%
1138 \test{\\\\lf}{}%
1139 \test{x\\\\lf}{x}%
1140 \test{\cr}{\lf}%
1141 \test{\cr\lf}{\lf}%
1142 \test{\lf}{\lf}%
1143 \test{\lf\cr}{\lf\lf}%
1144 \test{x\cr}{x\lf}%
1145 \test{x\cr\lf}{x\lf}%
1146 \test{x\lf}{x\lf}%
1147 \test{x\lf\cr}{x\lf\lf}%
1148 \test{x\\cr\lf y\cr}{xy\lf}%
1149 \end{qstest}
1150 \stop
1151 </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 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 \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>\#</code>	150, 286, 311, 671, 733, 853, 880, 955, 1040
<code>\\$</code>	149
<code>\%</code>	736, 878
<code>\(</code>	645, 886
<code>\)</code>	646, 886
<code>\@</code>	672, 729, 1003
<code>\@ReturnAfterFi</code>	136, <u>141</u>
<code>\@backslashchar</code>	860
<code>\@firstofone</code>	680, 683
<code>\@gobble</code>	677, 685
<code>\@ifundefined</code>	766, 780, 788
<code>\@one</code> ..	177, 180, 225, 228, 232, 342, 345
<code>\@onelevel@sanitize</code>	107, 803, 804, 846, 873, 902, 964, 1037, 1074
<code>\@percentchar</code>	878
<code>\@undefined</code> 53, 774, 775, 776, 777, 906, 908
<code>\@whilenum</code>	816
<code>\[</code>	734
<code>\]</code> 133, 278, 428, 647, 730, 882, 889, 1092, 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, 1136, 1138, 1139, 1148	
<code>\{</code>	669, 731
<code>\}</code>	335, 670, 732
<code>\]</code>	735
<code>\ </code>	277, 282, 877, 879
<code>\~</code>	881
Numbers	
<code>\0</code>	884, 885, 886
<code>\1</code>	891
<code>\2</code>	892, 893, 894, 895
<code>\3</code>	896, 897, 898, 899
<code>\8</code>	250
<code>\9</code>	251
A	
<code>\advance</code>	466, 502, 503, 710, 718, 825, 1023
B	
<code>\AllBytes</code> 813, 819, 822, 916, 917, 924, 925, 930, 931, 990, 991, 996, 997	
<code>\AllBytesHex</code>	828, 846, 849, 918, 919, 922, 923
<code>\AllBytesHexLC</code>	848, 928, 929
<code>\AllBytesName</code> .	851, 854, 873, 992, 993
<code>\AllBytesString</code> ...	875, 902, 998, 999
C	
<code>\catcode</code>	3, 4, 5, 6, 7, 18, 19, 20, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 65, 67, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 125, 149, 150, 277, 278, 669, 670, 671, 672, 707, 716, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 815, 853, 879, 880, 881, 882, 955, 1003, 1004, 1040, 1041, 1077
<code>\count</code>	1019, 1021, 1022, 1023
<code>\count@</code>	674, 703, 707, 709, 710, 714, 716, 717, 718, 814, 816, 817, 818, 825
<code>\countdef</code>	674
<code>\cr</code> ..	1091, 1103, 1123, 1136, 1140, 1141, 1143, 1144, 1145, 1147, 1148
<code>\csname</code>	8, 21, 45, 61, 64, 91, 94, 101, 112, 375, 381, 404, 433, 673, 676, 679, 682, 721, 743
D	
<code>\DefChar</code>	1078, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092
<code>\detokenize</code> ...	116, 120, 905, 906, 912
<code>\dimen</code> .	464, 465, 466, 469, 470, 499, 500, 501, 502, 503, 507, 508, 509
<code>\dimexpr</code>	457, 459, 484, 487, 495
<code>\do</code>	816
E	
<code>\EdefEscapeHex</code>	2, <u>382</u> , <u>409</u> , 917
<code>\EdefEscapeName</code>	<u>391</u> , <u>415</u> , 991
<code>\EdefEscapeString</code>	<u>396</u> , <u>418</u> , 997
<code>\EdefSanitize</code>	<u>3</u> , <u>90</u> , 123, 143, 243, 383, 388, 392, 397

<code>\EdefUnescapeHex</code>	387, 412, 923, 929, 935, 940, 946, 956, 962	<code>\next</code>	694, 696, 698
<code>\EdefUnescapeName</code>	2, 142, 1035	<code>\nul</code>	1085, 1098, 1118, 1136
<code>\EdefUnescapeString</code>	2, 242, 1072	<code>\number</code>	507, 508, 509
<code>\empty</code>	12	<code>\numexpr</code>	457, 459, 484, 487, 494, 495, 907, 908, 913
<code>\end</code>	744, 920, 926, 932, 937, 943, 952, 958, 988, 994, 1000, 1031, 1068, 1149	O	
<code>\endcsname</code>	8, 21, 45, 61, 64, 91, 94, 101, 112, 375, 381, 404, 433, 673, 676, 679, 682, 721, 743	<code>\one</code>	1086, 1109, 1110, 1111, 1112, 1113, 1114, 1129, 1130, 1131, 1132, 1133, 1134
<code>\endinginput</code>	30, 422	<code>\org@dtokenize</code>	905, 912
<code>\escapechar</code>	426	<code>\org@numexpr</code>	907, 913
<code>\eTeX</code>	782	P	
<code>\Expect</code>	808, 918, 924, 930, 936, 941, 950, 992, 998, 1010	<code>\PackageError</code>	767, 781, 789
<code>\ExpectVar</code>	799, 919, 925, 931, 942, 951, 957, 965, 993, 999, 1038, 1075	<code>\PackageInfo</code>	26
F		<code>\pdf@escapehex</code>	410
<code>\ff</code>	1090, 1102, 1122, 1136	<code>\pdf@escapename</code>	416
G		<code>\pdf@escapestring</code>	419
<code>\gdef</code>	151, 159, 813	<code>\pdf@unescapehex</code>	413
I		<code>\pdfescape</code>	751, 755, 759, 768
<code>\ifcase</code>	9, 173, 174, 352, 360, 476, 529, 542, 543, 582, 1011	<code>\pdfescapehex</code>	774
<code>\ifnum</code>	224, 227, 230, 231, 341, 344, 545, 581, 589, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 640, 643, 645, 646, 647, 709, 717, 818, 1021	<code>\pdfescapename</code>	776
<code>\ifPE@etex</code>	431, 437, 455, 481, 567, 614, 624, 656	<code>\pdfescapestring</code>	777
<code>\ifx</code>	10, 12, 21, 45, 53, 56, 91, 101, 133, 160, 164, 259, 264, 375, 381, 433, 449, 522, 523, 579, 638, 673, 676, 679, 682, 721, 800, 808, 970	<code>\pdfunescapehex</code>	775, 968
<code>\immediate</code>	23, 47	<code>\PE@@NormalizeLineEnd</code>	255, 258
<code>\IncludeTests</code>	796	<code>\PE@@OctChar</code>	483, 491
<code>\input</code>	376, 722	<code>\PE@AtEnd</code>	69, 70, 421, 666
<code>\iterate</code>	690, 692, 694	<code>\PE@backslash</code>	425, 492, 506
L		<code>\PE@CheckEndBackslash</code>	332
<code>\lccode</code>	817	<code>\PE@DeHex</code>	517, 521
<code>\lf</code>	1089, 1101, 1121, 1136, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148	<code>\PE@DeName</code>	155, 159, 188, 192
<code>\LoadCommand</code>	722, 739	<code>\PE@DeString</code>	286, 372
<code>\LogTests</code>	797	<code>\PE@edefbabel</code>	402, 410, 413, 416, 419
<code>\loop</code>	688, 704, 715, 1020	<code>\PE@EscapeHex</code>	385, 437
<code>\lowercase</code>	821, 847	<code>\PE@EscapeName</code>	394, 567
M		<code>\PE@EscapeNameAdd</code>	585, 590, 604, 612, 614
<code>\makeatletter</code>	748, 811	<code>\PE@EscapeNameHashChar</code>	592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 611
<code>\meaning</code>	103	<code>\PE@EscapeNameTokens</code>	569, 574, 578
N		<code>\PE@EscapeString</code>	399, 624
<code>\NeedsTeXFormat</code>	747	<code>\PE@EscapeStringAdd</code>	645, 646, 647, 649, 656
<code>\newcommand</code>	799, 828, 848, 851, 875	<code>\PE@EscapeStringTokens</code>	626, 632, 637
<code>\newif</code>	431	<code>\PE@etextrue</code>	435
		<code>\PE@hash</code>	424, 585, 590, 612
		<code>\PE@HexChar</code>	451, 455, 586, 591
		<code>\PE@HexDigit</code>	457, 458, 469, 470, 474
		<code>\PE@InitUccodeHexDigit</code>	153, 199, 515
		<code>\PE@next</code>	162, 167, 188, 192, 196, 261, 265, 267, 270, 353, 355, 357, 361, 363, 365, 524, 530, 534, 553, 556, 562, 565
		<code>\PE@NormalizeLineEnd</code>	245, 253
		<code>\PE@OctAll</code>	355, 361, 363, 367
		<code>\PE@OctChar</code>	481, 641, 644
		<code>\PE@OctI</code>	351
		<code>\PE@OctII</code>	353, 359
		<code>\PE@onelevel@sanitize</code>	96, 102, 107, 119, 146, 247

<code>\PE@result</code>	154, 157, 161, 166, 187, 191, 254, 256, 260, 263, 370, 443, 445, 467, 468, 504, 505, 516, 519, 552, 573, 575, 618, 619, 631, 634, 660, 661	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, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148
<code>\PE@sanitize</code>	<u>123</u>	
<code>\PE@SanitizeSpaceOther</code>		
.	<u>128</u> , 144, 244, 384, 393, 398	
<code>\PE@space@other</code>	<u>124</u> , 135	
<code>\PE@Space@space</code>	<u>127</u> , 546	
<code>\PE@SpaceToOther</code>	<u>129</u> , <u>131</u>	<code>\the</code> 65, 71, 484, 487, 494, 495, 707
<code>\PE@strip@prefix</code>	103, <u>105</u>	<code>\TMP@EnsureCode</code> 68, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89
<code>\PE@temp</code>	184, 187, 546, 549, 552	<code>\typeout</code> . . 805, 973, 974, 975, 976, 977
<code>\PE@testA</code>	170, 173, 182, 527, 529, 539, 542, 544	
<code>\PE@testB</code>	171, 174, 182, 540, 543, 544	
<code>\PE@TestOctDigit</code>	<u>340</u> , 352, 360	
<code>\PE@TestUcHexDigit</code>		
.	173, 174, <u>223</u> , 529, 542, 543, 1011	
<code>\PE@ToHex</code>	439, 444, <u>448</u>	
<code>\PE@UnescapeHex</code>	389, <u>513</u>	
<code>\PE@UnescapeName</code>	145, <u>148</u>	
<code>\PE@UnescapeString</code>	246, <u>279</u>	
<code>\ProvidesFile</code>	750, 754, 758, 762	<code>\uppercase</code> 169, 183, 273, 369, 526, 538, 548, 1007, 1081
<code>\ProvidesPackage</code>	62	
R		
<code>\range</code> 1018, 1026, 1027, 1028, 1029, 1030		
<code>\RangeCatcodeInvalid</code>		
.	713, 725, 726, 727, 728	
<code>\repeat</code>	688, 700, 711, 719, 1024	
<code>\RequirePackage</code>	378, 795, 910	
<code>\RestoreCatcodes</code>	702, 705, 706, 740	
S		
<code>\space</code>	819, 942, 957, 972, 1067, 1136	
<code>\stop</code>	770, 784, 792, 1150	
T		
<code>\tab</code>	1088, 1100, 1120, 1136	
<code>\Test</code>	724, 742	
<code>\test</code>	961, 978, 981, 982, 983, 984, 985, 986, 987, 1005, 1022, 1034, 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, 1067, 1071, 1093, 1094, 1095,	
		<code>\y</code> 947, 949, 963, 964, 965, 969, 970, 1036, 1037, 1038, 1073, 1074, 1075
		Z
		<code>\z</code> 948, 950, 951
		<code>\z@</code> 175, 182, 186, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 234, 237, 347, 368, 544, 545