

The pdfescape package

Heiko Oberdiek
<heiko.oberdiek at gmail.com>

2010/03/01 v1.9

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 (pdfTeX analogues)	10
2.7	Help macros	11
2.7.1	Characters	11
2.7.2	Switch for ϵ -TeX	12
2.8	Conversions	12
2.8.1	Conversion to hex string	12
2.8.2	Character code to octal number	13
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	18
3.6	Test with LuaTeX	18
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 LuaTeX	18
3.8	Common part	18

4	Installation	24
4.1	Download	24
4.2	Bundle installation	25
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]		27
[2007/09/09 v1.6]		27
[2007/10/27 v1.7]		27
[2007/11/11 v1.8]		27
[2010/03/01 v1.9]		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 `<string>` is expanded, converted to a string of tokens with catcode 12 (other) and space tokens, and stored in macro `<cmd>`.

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   }
```

```

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   [2010/03/01 v1.9 Provides hex, PDF name and string conversions (H0)]

```

2.2 Catcodes

```

63 \begingroup
64 \catcode123 1 % {
65 \catcode125 2 % }
66 \def\x{\endgroup
67   \expandafter\edef\csname PE@AtEnd\endcsname{%
68     \catcode35 \the\catcode35\relax
69     \catcode64 \the\catcode64\relax
70     \catcode123 \the\catcode123\relax
71     \catcode125 \the\catcode125\relax
72   }%
73 }%
74 \x
75 \catcode35 6 % #
76 \catcode64 11 % @
77 \catcode123 1 % {
78 \catcode125 2 % }
79 \def\TMP@EnsureCode#1#2#3{%
80   \edef\PE@AtEnd{%
81     \PE@AtEnd
82     #1#2 \the#1#2\relax
83   }%
84   #1#2 #3\relax
85 }
86 \TMP@EnsureCode\catcode{0}{12}% ^^@
87 \TMP@EnsureCode\catcode{34}{12}% "
88 \TMP@EnsureCode\catcode{39}{12}% '
89 \TMP@EnsureCode\catcode{42}{12}% *
90 \TMP@EnsureCode\catcode{45}{12}% -
91 \TMP@EnsureCode\catcode{46}{12}% .
92 \TMP@EnsureCode\catcode{47}{12}% /
93 \TMP@EnsureCode\catcode{60}{12}% <
94 \TMP@EnsureCode\catcode{61}{12}% =
95 \TMP@EnsureCode\catcode{62}{12}% >
96 \TMP@EnsureCode\catcode{94}{7}% ^
97 \TMP@EnsureCode\catcode{96}{12}% `
98 \TMP@EnsureCode\uccode{34}{0}% "
99 \TMP@EnsureCode\uccode{48}{0}% 0
100 \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.

```

101 \begingroup\expandafter\expandafter\expandafter\endgroup
102 \expandafter\ifx\csname detokenize\endcsname\relax

```

```

103 \long\def\EdefSanitize#1#2{%
104   \begingroup
105     \csname @safe@activetrue\endcsname
106     \edef#1{#2}%
107     \PE@onelevel@sanitize#1%
108   \expandafter\endgroup
109   \expandafter\def\expandafter#1\expandafter{#1}%
110 }%
111 \begingroup\expandafter\expandafter\expandafter\endgroup
112 \expandafter\ifx\csname @onelevel@sanitize\endcsname\relax
113   \def\PE@onelevel@sanitize#1{%
114     \edef#1{\expandafter\PE@strip@prefix\meaning#1}%
115   }%
116   \def\PE@strip@prefix#1>{%}%
117 \else
118   \let\PE@onelevel@sanitize\@onelevel@sanitize
119 \fi
120 \else
121 \long\def\EdefSanitize#1#2{%
122   \begingroup
123     \csname @safe@activetrue\endcsname
124     \edef#1{#2}%
125   \expandafter\endgroup
126   \expandafter\def\expandafter#1\expandafter{%
127     \detokenize\expandafter{#1}%
128   }%
129 }%
130 \def\PE@onelevel@sanitize#1{%
131   \edef#1{\detokenize\expandafter{#1}}%
132 }%
133 \fi

```

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

```
134 \let\PE@sanitize\EdefSanitize
```

2.3.1 Space characters

`\PE@space@other`

```

135 \begingroup
136   \catcode'\ =12\relax%
137 \def\x{\endgroup\def\PE@space@other{ }}\x\relax

```

`\PE@space@space`

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

2.3.2 Space normalization

`\PE@SanitizeSpaceOther`

```

139 \def\PE@SanitizeSpaceOther#1{%
140   \edef#1{\expandafter\PE@SpaceToOther#1 \relax}%
141 }

```

`\PE@SpaceToOther`

```

142 \def\PE@SpaceToOther#1 #2\relax{%
143   #1%
144   \ifx\#2\%
145   \else
146     \PE@space@other
147     \@ReturnAfterFi{%
148       \PE@SpaceToOther#2\relax

```

```

149    }%
150    \fi
151 }

\@ReturnAfterFi

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

```

2.4 \EdefUnescapeName

\EdefUnescapeName

```

153 \def\EdefUnescapeName#1#2{%
154   \EdefSanitize#1{#2}%
155   \PE@SanitizeSpaceOther#1%
156   \PE@UnescapeName#1%
157   \PE@onelevel@sanitize#1%
158 }

\PE@UnescapeName

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

```

```

203         \def\PE@next{\PE@DeName$2$3}%
204     \fi
205 \fi
206 \fi
207 \PE@next
208 }%
209 \endgroup

```

\PE@InitUccodeHexDigit

```

210 \def\PE@InitUccodeHexDigit{%
211     \uccode'a='A\relax
212     \uccode'b='B\relax
213     \uccode'c='C\relax
214     \uccode'd='D\relax
215     \uccode'e='E\relax
216     \uccode'f='F\relax
217     \uccode'A=\z@
218     \uccode'B=\z@
219     \uccode'C=\z@
220     \uccode'D=\z@
221     \uccode'E=\z@
222     \uccode'F=\z@
223     \uccode'0=\z@
224     \uccode'1=\z@
225     \uccode'2=\z@
226     \uccode'3=\z@
227     \uccode'4=\z@
228     \uccode'5=\z@
229     \uccode'6=\z@
230     \uccode'7=\z@
231     \uccode'8=\z@
232     \uccode'9=\z@
233 }

```

\PE@TestUcHexDigit

```

234 \def\PE@TestUcHexDigit#1{%
235     \ifnum'#1<48 % 0
236         \@ne
237     \else
238         \ifnum'#1>70 % F
239             \@ne
240         \else
241             \ifnum'#1>57 % 9
242                 \ifnum'#1<65 % A
243                     \@ne
244                 \else
245                     \z@
246                 \fi
247             \else
248                 \z@
249             \fi
250         \fi
251     \fi
252 }

```

2.5 \EdefUnescapeString

\EdefUnescapeString

```

253 \def\EdefUnescapeString#1#2{%
254     \EdefSanitize#1{#2}%
255     \PE@SanitizeSpaceOther#1%
256     \PE@NormalizeLineEnd#1%

```

```

257 \PE@UnescapeString#1%
258 \PE@onelevel@sanitize#1%
259 }

260 \begingroup
261 \uccode'\8=10 % lf
262 \uccode'\9=13 % cr
263 \def\x#1#2{\endgroup

\PE@NormalizeLineEnd
264 \def\PE@NormalizeLineEnd##1{%
265 \def\PE@result{}%
266 \expandafter\PE@@NormalizeLineEnd##1#2\relax
267 \let##1\PE@result
268 }%

\PE@@NormalizeLineEnd
269 \def\PE@@NormalizeLineEnd##1#2##2{%
270 \ifx\relax##2%
271 \edef\PE@result{\PE@result##1}%
272 \let\PE@next\relax
273 \else
274 \edef\PE@result{\PE@result##1#1}%
275 \ifx#1##2% lf
276 \let\PE@next\PE@@NormalizeLineEnd
277 \else
278 \def\PE@next{\PE@@NormalizeLineEnd##2}%
279 \fi
280 \fi
281 \PE@next
282 }%
283 }%
284 \uppercase{%
285 \x 89%
286 }

287 \begingroup
288 \catcode'\|=0 %
289 \catcode'\|=12 %

\PE@UnescapeString
290 |gdef|PE@UnescapeString#1{%
291 |begingroup
292 |def|PE@result{}%
293 |expandafter|PE@DeString#1\|relax
294 |expandafter|endgroup
295 |expandafter|def|expandafter#1|expandafter{|PE@result}%
296 }%

\PE@DeString
297 |gdef|PE@DeString#1\#2{%
298 |ifx|relax#2%
299 |edef|PE@result{|PE@result#1}%
300 |let|PE@next|relax
301 |else
302 |if n#2%
303 |uccode|z@=10 %
304 |else|if r#2%
305 |uccode|z@=13 %
306 |else|if t#2%
307 |uccode|z@=9 %
308 |else|if b#2%

```

```

309         |uccode|z@=8 %
310     |else|if f#2%
311         |uccode|z@=12 %
312     |else
313         |uccode|z@=|z@
314     |fi|fi|fi|fi|fi
315     |ifnum|uccode|z@>|z@
316         |uppercase{%
317             |edef|PE@temp{^^@}%
318         }%
319         |edef|PE@result{|PE@result#1|PE@temp}%
320         |let|PE@next|PE@DeString
321     |else
322         |if|#2% backslash
323             |edef|PE@result{|PE@result#1}%
324             |let|PE@next|PE@CheckEndBackslash
325         |else
326             |ifnum'#2=10 % linefeed
327                 |edef|PE@result{|PE@result#1}%
328                 |let|PE@next|PE@DeString
329             |else
330                 |ifcase|PE@TestOctDigit#2%
331                     |edef|PE@result{|PE@result#1}%
332                     |def|PE@next{|PE@OctI#2}%
333                 |else
334                     |edef|PE@result{|PE@result#1#2}%
335                     |let|PE@next|PE@DeString
336                 |fi
337             |fi
338         |fi
339     |fi
340 |fi
341 |PE@next
342 }%

```

\PE@CheckEndBackslash

```

343 |gdef|PE@CheckEndBackslash#1{%
344     |ifx|relax#1%
345     |else
346         |edef|PE@result{|PE@result\}%
347         |expandafter|PE@DeString|expandafter#1%
348     |fi
349 }%

```

350 |endgroup

\PE@TestOctDigit

```

351 \def\PE@TestOctDigit#1{%
352     \ifnum'#1<48 % 0
353         \@ne
354     \else
355         \ifnum'#1>55 % 7
356             \@ne
357         \else
358             \z@
359         \fi
360     \fi
361 }

```

\PE@OctI

```

362 \def\PE@OctI#1#2{%
363     \ifcase\PE@TestOctDigit#2%

```

```

364     \def\PE@next{\PE@OctII{#1#2}}%
365 \else
366     \def\PE@next{\PE@OctAll{#1#2}}%
367 \fi
368 \PE@next
369 }

\PE@OctII

370 \def\PE@OctII#1#2{%
371     \ifcase\PE@TestOctDigit#2%
372         \def\PE@next{\PE@OctAll{#1#2}}%
373     \else
374         \def\PE@next{\PE@OctAll{#1}#2}%
375     \fi
376     \PE@next
377 }

\PE@OctAll

378 \def\PE@OctAll#1{%
379     \uccode\z@='#1\relax
380     \uppercase{%
381         \edef\PE@result{\PE@result^^@}%
382     }%
383     \PE@DeString
384 }

```

2.6 User macros (pdfTeX analogues)

```

385 \begingroup\expandafter\expandafter\expandafter\endgroup
386 \expandafter\ifx\csname RequirePackage\endcsname\relax
387     \input pdftexcmds.sty\relax
388 \else
389     \RequirePackage{pdftexcmds}[2007/11/11]%
390 \fi

391 \begingroup\expandafter\expandafter\expandafter\endgroup
392 \expandafter\ifx\csname pdf@escapehex\endcsname\relax

\EdefEscapeHex

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

\EdefUnescapeHex

398     \def\EdefUnescapeHex#1#2{%
399         \EdefSanitize#1{#2}%
400         \PE@UnescapeHex#1%
401     }%

\EdefEscapeName

402     \long\def\EdefEscapeName#1#2{%
403         \EdefSanitize#1{#2}%
404         \PE@SanitizeSpaceOther#1%
405         \PE@EscapeName#1%
406     }%

\EdefEscapeString

407     \long\def\EdefEscapeString#1#2{%
408         \EdefSanitize#1{#2}%
409         \PE@SanitizeSpaceOther#1%

```

```

410     \PE@EscapeString#1%
411 }%

412 \else

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

413 \long\def\PE@edefbabel#1#2#3{%
414     \begingroup
415     \csname @save@activetrue\endcsname
416     \edef#1{#2{#3}}%
417     \expandafter\endgroup
418     \expandafter\def\expandafter#1\expandafter{#1}%
419 }%

\EdefEscapeHex

420 \long\def\EdefEscapeHex#1#2{%
421     \PE@edefbabel#1\pdf@escapehex{#2}%
422 }%

\EdefUnescapeHex

423 \def\EdefUnescapeHex#1#2{%
424     \PE@edefbabel#1\pdf@unescapehex{#2}%
425 }%

\EdefEscapeName

426 \long\def\EdefEscapeName#1#2{%
427     \PE@edefbabel#1\pdf@escapename{#2}%
428 }%

\EdefEscapeString

429 \long\def\EdefEscapeString#1#2{%
430     \PE@edefbabel#1\pdf@escapestring{#2}%
431 }%

432 \PE@AtEnd
433 \expandafter\endinput
434 \fi

```

2.7 Help macros

2.7.1 Characters

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

```

\PE@hash

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

\PE@backslash

436 \begingroup
437 \escapechar=-1 %
438 \edef\x{\endgroup
439     \def\noexpand\PE@backslash{\string\\}%
440 }
441 \x

```

2.7.2 Switch for ϵ -T_EX

```
442 \newif\ifPE@etex
443 \begingroup\expandafter\expandafter\expandafter\endgroup
444 \expandafter\ifx\csname numexpr\endcsname\relax
445 \else
446   \PE@etextrue
447 \fi
```

2.8 Conversions

2.8.1 Conversion to hex string

\PE@EscapeHex

```
448 \ifPE@etex
449   \def\PE@EscapeHex#1{%
450     \edef#1{\expandafter\PE@ToHex#1\relax}%
451   }%
452 \else
453   \def\PE@EscapeHex#1{%
454     \def\PE@result{%
455       \expandafter\PE@ToHex#1\relax
456     }\let#1\PE@result
457   }%
458 \fi
```

\PE@ToHex

```
459 \def\PE@ToHex#1{%
460   \ifx\relax#1%
461   \else
462     \PE@HexChar{#1}%
463     \expandafter\PE@ToHex
464   \fi
465 }%
```

\PE@HexChar

```
466 \ifPE@etex
467   \def\PE@HexChar#1{%
468     \PE@HexDigit{\numexpr\dimexpr.0625\dimexpr'#1sp\relax\relax\relax}%
469     \PE@HexDigit{%
470       \numexpr'#1-16*\dimexpr.0625\dimexpr'#1sp\relax\relax\relax
471     }%
472   }%
473 \else
474   \def\PE@HexChar#1{%
475     \dimen0='#1sp%
476     \dimen2=.0625\dimen0 %
477     \advance\dimen0-16\dimen2 %
478     \edef\PE@result{%
479       \PE@result
480       \PE@HexDigit{\dimen2 }%
481       \PE@HexDigit{\dimen0 }%
482     }%
483   }%
484 \fi
```

\PE@HexDigit

```
485 \def\PE@HexDigit#1{%
486   \expandafter\string
487   \ifcase#1%
488     0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or 9\or
489     A\or B\or C\or D\or E\or F%
490   \fi
491 }
```

2.8.2 Character code to octal number

\PE@OctChar

```

492 \ifPE@etex
493   \def\PE@OctChar#1{%
494     \expandafter\PE@@OctChar
495       \the\numexpr\dimexpr.015625\dimexpr'#1sp\relax\relax
496       \expandafter\relax
497       \expandafter\relax
498       \the\numexpr\dimexpr.125\dimexpr'#1sp\relax\relax\relax
499       \relax
500       #1%
501   }%
502   \def\PE@@OctChar#1\relax#2\relax#3{%
503     \PE@backslash
504     #1%
505     \the\numexpr#2-8*#1\relax
506     \the\numexpr\dimexpr'#3sp\relax-8*#2\relax
507   }%
508 \else
509   \def\PE@OctChar#1{%
510     \dimen0='#1sp%
511     \dimen2=.125\dimen0 %
512     \dimen4=.125\dimen2 %
513     \advance\dimen0-8\dimen2 %
514     \advance\dimen2-8\dimen4 %
515     \edef\PE@result{%
516       \PE@result
517       \PE@backslash
518       \number\dimen4 %
519       \number\dimen2 %
520       \number\dimen0 %
521     }%
522   }%
523 \fi

```

2.8.3 Unpack hex string

\PE@UnescapeHex

```

524 \def\PE@UnescapeHex#1{%
525   \begingroup
526     \PE@InitUccodeHexDigit
527     \def\PE@result{}%
528     \expandafter\PE@DeHex#1\relax\relax
529   \expandafter\endgroup
530   \expandafter\def\expandafter#1\expandafter{\PE@result}%
531 }

```

\PE@DeHex

```

532 \def\PE@DeHex#1#2{%
533   \ifx#2\relax
534     \ifx#1\relax
535       \let\PE@next\relax
536     \else
537       \uppercase{%
538         \def\PE@testA{#1}%
539       }%
540       \ifcase\expandafter\PE@TestUcHexDigit\PE@testA
541         \def\PE@next{%
542           \PE@DeHex#10\relax\relax
543         }%
544       \else

```

```

545         \let\PE@next\relax
546     \fi
547 \fi
548 \else
549     \uppercase{%
550         \def\PE@testA{#1}%
551         \def\PE@testB{#2}%
552     }%
553     \ifcase\expandafter\PE@TestUcHexDigit\PE@testA
554     \ifcase\expandafter\PE@TestUcHexDigit\PE@testB
555         \uccode\z@="\PE@testA\PE@testB\relax
556         \ifnum\uccode\z@=32 %
557             \let\PE@temp\PE@space@space
558         \else
559             \uppercase{%
560                 \def\PE@temp{^^@}%
561             }%
562         \fi
563         \edef\PE@result{\PE@result\PE@temp}%
564         \let\PE@next\PE@DeHex
565     \else
566         % invalid input sequence
567         \def\PE@next{%
568             \PE@DeHex#1%
569         }%
570     \fi
571 \else
572     % invalid input sequence
573     \def\PE@next{\PE@DeHex#2}%
574 \fi
575 \fi
576 \PE@next
577 }

```

2.8.4 Conversion to PDF name

\PE@EscapeName

```

578 \ifPE@etex
579     \def\PE@EscapeName#1{%
580         \edef#1{\expandafter\PE@EscapeNameTokens#1\relax}%
581     }%
582 \else
583     \def\PE@EscapeName#1{%
584         \def\PE@result{%
585             \expandafter\PE@EscapeNameTokens#1\relax
586             \let#1\PE@result
587         }%
588     \fi

```

\PE@EscapeNameTokens

```

589 \def\PE@EscapeNameTokens#1{%
590     \ifx\relax#1%
591     \else
592         \ifnum'#1<33 %
593             \ifcase'#1 %
594                 % drop illegal zero
595             \else
596                 \PE@EscapeNameAdd\PE@hash
597                 \PE@HexChar#1%
598             \fi
599         \else
600             \ifnum'#1>126 %

```

```

601      \PE@EscapeNameAdd\PE@hash
602      \PE@HexChar#1%
603      \else \ifnum'#1=35 \PE@EscapeNameHashChar 23% #
604      \else\ifnum'#1=37 \PE@EscapeNameHashChar 25% %
605      \else\ifnum'#1=40 \PE@EscapeNameHashChar 28% (
606      \else\ifnum'#1=41 \PE@EscapeNameHashChar 29% )
607      \else\ifnum'#1=47 \PE@EscapeNameHashChar 2F% /
608      \else\ifnum'#1=60 \PE@EscapeNameHashChar 3C% <
609      \else\ifnum'#1=62 \PE@EscapeNameHashChar 3E% >
610      \else\ifnum'#1=91 \PE@EscapeNameHashChar 5B% [
611      \else\ifnum'#1=93 \PE@EscapeNameHashChar 5D% ]
612      \else\ifnum'#1=123 \PE@EscapeNameHashChar 7B% {
613      \else\ifnum'#1=125 \PE@EscapeNameHashChar 7D% }
614      \else
615      \PE@EscapeNameAdd{#1}%
616      \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
617      \fi
618      \fi
619      \expandafter\PE@EscapeNameTokens
620      \fi
621      }%
622 \def\PE@EscapeNameHashChar#1#2{%
623 \PE@EscapeNameAdd{\PE@hash\string#1\string#2}%
624 }%

```

\PE@EscapeNameAdd

```

625 \ifPE@etex
626 \def\PE@EscapeNameAdd#1{#1}%
627 \else
628 \def\PE@EscapeNameAdd#1{%
629 \edef\PE@result{%
630 \PE@result
631 #1%
632 }%
633 }%
634 \fi

```

2.8.5 Conversion to PDF string

\PE@EscapeString

```

635 \ifPE@etex
636 \def\PE@EscapeString#1{%
637 \edef#1{\expandafter\PE@EscapeStringTokens#1\relax}%
638 }%
639 \else
640 \def\PE@EscapeString#1{%
641 \begingroup
642 \def\PE@result{}%
643 \expandafter\PE@EscapeStringTokens#1\relax
644 \expandafter\endgroup
645 \expandafter\def\expandafter#1\expandafter{\PE@result}%
646 }%
647 \fi

```

\PE@EscapeStringTokens

```

648 \def\PE@EscapeStringTokens#1{%
649 \ifx\relax#1%
650 \else
651 \ifnum'#1<33 %
652 \PE@OctChar#1%
653 \else
654 \ifnum'#1>126 %

```

```

655      \PE@OctChar#1%
656      \else \ifnum'#1=40 \PE@EscapeStringAdd{\string\}% (
657      \else\ifnum'#1=41 \PE@EscapeStringAdd{\string\)}% )
658      \else\ifnum'#1=92 \PE@EscapeStringAdd{\string\\}% \
659      \else
660      \PE@EscapeStringAdd{#1}%
661      \fi\fi\fi
662      \fi
663      \fi
664      \expandafter\PE@EscapeStringTokens
665      \fi
666 }%

```

\PE@EscapeStringAdd

```

667 \ifPE@etex
668 \def\PE@EscapeStringAdd#1{#1}%
669 \else
670 \def\PE@EscapeStringAdd#1{%
671 \edef\PE@result{%
672 \PE@result
673 #1%
674 }%
675 }%
676 \fi

677 \PE@AtEnd
678 </package>

```

3 Test

3.1 Catcode checks for loading

```

679 <*test1>

680 \catcode'\{=1 %
681 \catcode'\}=2 %
682 \catcode'\#=6 %
683 \catcode'\@=11 %
684 \expandafter\ifx\csname count@\endcsname\relax
685 \countdef\count@=255 %
686 \fi
687 \expandafter\ifx\csname @gobble\endcsname\relax
688 \long\def@gobble#1{}%
689 \fi
690 \expandafter\ifx\csname @firstofone\endcsname\relax
691 \long\def@firstofone#1{#1}%
692 \fi
693 \expandafter\ifx\csname loop\endcsname\relax
694 \expandafter\@firstofone
695 \else
696 \expandafter\@gobble
697 \fi
698 {%
699 \def\loop#1\repeat{%
700 \def\body{#1}%
701 \iterate
702 }%
703 \def\iterate{%
704 \body
705 \let\next\iterate
706 \else
707 \let\next\relax

```

```

708 \fi
709 \next
710 }%
711 \let\repeat=\fi
712 }%
713 \def\RestoreCatcodes{}
714 \count@=0 %
715 \loop
716 \edef\RestoreCatcodes{%
717 \RestoreCatcodes
718 \catcode\the\count@=\the\catcode\count@\relax
719 }%
720 \ifnum\count@<255 %
721 \advance\count@ 1 %
722 \repeat
723
724 \def\RangeCatcodeInvalid#1#2{%
725 \count@=#1\relax
726 \loop
727 \catcode\count@=15 %
728 \ifnum\count@<#2\relax
729 \advance\count@ 1 %
730 \repeat
731 }
732 \expandafter\ifx\csname LoadCommand\endcsname\relax
733 \def\LoadCommand{\input pdfescape.sty\relax}%
734 \fi
735 \def\Test{%
736 \RangeCatcodeInvalid{0}{47}%
737 \RangeCatcodeInvalid{58}{64}%
738 \RangeCatcodeInvalid{91}{96}%
739 \RangeCatcodeInvalid{123}{255}%
740 \catcode'\@=12 %
741 \catcode'\=0 %
742 \catcode'\{=1 %
743 \catcode'\}=2 %
744 \catcode'\#=6 %
745 \catcode'\[=12 %
746 \catcode'\]=12 %
747 \catcode'\%=14 %
748 \catcode'\ =10 %
749 \catcode13=5 %
750 \LoadCommand
751 \RestoreCatcodes
752 }
753 \Test
754 \csname @@end\endcsname
755 \end
756 </test1>

```

3.2 Macro tests

```

757 <*test2 | test3 | test4 | test5>
758 \NeedsTeXFormat{LaTeX2e}
759 \makeatletter

```

3.3 Test with \pdfescape... commands

```

760 <*test2>
761 \ProvidesFile{pdfescape-test2.tex}%
762 [2010/03/01 v1.9 Test with \string\pdfescape... commands]%
763 </test2>

```

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

```

764 <*test3>
765 \ProvidesFile{pdfescape-test3.tex}%
766     [2010/03/01 v1.9 Test without \string\pdfescape..., with e-TeX]%
767 </test3>

```

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

```

768 <*test4>
769 \ProvidesFile{pdfescape-test4.tex}%
770     [2010/03/01 v1.9 Test without \string\pdfescape... and e-TeX]%
771 </test4>

```

3.6 Test with LuaTeX

```

772 <*test5>
773 \ProvidesFile{pdfescape-test5.tex}%
774     [2010/03/01 v1.9 Test with LuaTeX]%
775 </test5>

```

3.7 Check/ensure test preconditions

3.7.1 Check \pdfescape...

```

776 <*test2>
777 \@ifundefined{pdfescapehex}{%
778     \PackageError{pdfescape-test2}{%
779         Missing \string\pdfescape... commands%
780     }{Test aborted.}%
781     \stop
782 }{}
783 </test2>

784 <*test3 | test4>
785 \let\pdfescapehex\@undefined
786 \let\pdfunescapehex\@undefined
787 \let\pdfescapename\@undefined
788 \let\pdfescapestring\@undefined
789 </test3 | test4>

```

3.7.2 Check ε -TeX

```

790 <*test3>
791 \@ifundefined{numexpr}{%
792     \PackageError{pdfescape-test3}{%
793         Missing \eTeX
794     }{Test aborted.}%
795     \stop
796 }{}
797 </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

```

798 <*test5>
799 \@ifundefined{directlua}{%
800     \PackageError{pdfescape-test5}{%
801         Missing LuaTeX%
802     }{Test aborted.}%
803     \stop
804 }{}
805 </test5>

```

3.8 Common part

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

```

806 \RequirePackage{qstest}

```

```

807 \IncludeTests{*}
808 \LogTests{log}{*}{*}
809
810 \newcommand*{\ExpectVar}[2]{%
811   \ifx#1#2%
812   \else
813     \begingroup
814       \@onelevel@sanitize#1%
815       \@onelevel@sanitize#2%
816       \typeout{[#1] <> [#2]}% hash-ok
817     \endgroup
818   \fi
819   \Expect*{\ifx#1#2true\else false\fi}{true}%
820 }
821
822 \makeatletter
823 \begingroup
824 \gdef\AllBytes{}%
825 \count@=0 %
826 \catcode@=12 %
827 \@whilenum\count@<256 \do{%
828   \lccode@=\count@
829   \ifnum\count@=32 %
830     \xdef\AllBytes{\AllBytes\space}%
831   \else
832     \lowercase{%
833       \xdef\AllBytes{\AllBytes^^@}%
834     }%
835   \fi
836   \advance\count@ by 1 %
837 }%
838 \endgroup
839 \newcommand*{\AllBytesHex}{%
840   000102030405060708090A0B0C0D0E0F%
841   101112131415161718191A1B1C1D1E1F%
842   202122232425262728292A2B2C2D2E2F%
843   303132333435363738393A3B3C3D3E3F%
844   404142434445464748494A4B4C4D4E4F%
845   505152535455565758595A5B5C5D5E5F%
846   606162636465666768696A6B6C6D6E6F%
847   707172737475767778797A7B7C7D7E7F%
848   808182838485868788898A8B8C8D8E8F%
849   909192939495969798999A9B9C9D9E9F%
850   A0A1A2A3A4A5A6A7A8A9AABACADA EAF%
851   B0B1B2B3B4B5B6B7B8B9BABBBBCDBEBF%
852   C0C1C2C3C4C5C6C7C8C9CACBCCDCECF%
853   D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
854   E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
855   F0F1F2F3F4F5F6F7F8F9FABFBCFDFEFF%
856 }
857 \@onelevel@sanitize\AllBytesHex
858 \expandafter\lowercase\expandafter{%
859   \expandafter\newcommand\expandafter*\expandafter\AllBytesHexLC
860   \expandafter{\AllBytesHex}%
861 }
862 \newcommand*{\AllBytesName}{%
863 \begingroup
864   \catcode'\# =12 %
865   \xdef\AllBytesName{%
866     #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
867     #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
868     #20!"#23$#25&'#28#29*+,-.#2F%

```

```

869 0123456789:;#3C=#3E?%
870 @ABCDEFGHIJKLMNO%
871 PQRSTUVWXYZ#5B\@backslashchar#5D^_%
872 'abcdefghijklmno%
873 pqrstuvwxyz#7B|#7D\string~#7F%
874 #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
875 #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
876 #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
877 #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
878 #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
879 #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
880 #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
881 #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
882 }%
883 \endgroup
884 \@onelevel@sanitize\AllBytesName
885
886 \newcommand*{\AllBytesString}{%
887 \begin{group}
888 \def\|{|}%
889 \edef\%{\@percentchar}%
890 \catcode'\|=0 %
891 \catcode'\#=12 %
892 \catcode'\~=12 %
893 \catcode'\|=12 %
894 |xdef|AllBytesString{%
895 \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
896 \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
897 \040!"#$%&'(\)*+,-./%
898 0123456789:;<=>?%
899 @ABCDEFGHIJKLMNO%
900 PQRSTUVWXYZ[\]^_%
901 'abcdefghijklmno%
902 pqrstuvwxyz{|}~\177%
903 \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
904 \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
905 \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
906 \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
907 \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
908 \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
909 \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
910 \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
911 }%
912 \endgroup
913 \@onelevel@sanitize\AllBytesString
914
915 <*test4>
916 \let\org@detokenize\detokenize
917 \let\detokenize\@undefined
918 \let\org@numexpr\numexpr
919 \let\numexpr\@undefined
920 </test4>
921 \RequirePackage{pdfescape}
922 <*test4>
923 \let\detokenize\org@detokenize
924 \let\numexpr\org@numexpr
925 </test4>
926
927 \begin{qstest}{all-hex}{\AllBytes, escapehex}
928 \EdefEscapeHex\x{\AllBytes}%
929 \Expect*{\x}*{\AllBytesHex}%
930 \ExpectVar\x\AllBytesHex

```

```

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

```

```

993 \test{XY}{}%
994 \test{XYZ}{}%
995 \test{A}{^~a0}%
996 \test{AX}{^^a0}%
997 \test{XA}{^^a0}%
998 \test{XXAXX}{^^a0}%
999 \end{qstest}
1000
1001 \begin{qstest}{all-name}{\AllBytes, escapename}
1002 \EdefEscapeName\x{\AllBytes}%
1003 \Expect*{\x}*{\AllBytesName}%
1004 \ExpectVar\x\AllBytesName
1005 \end{qstest}
1006
1007 \begin{qstest}{all-string}{\AllBytes, escapestring}
1008 \EdefEscapeString\x{\AllBytes}%
1009 \Expect*{\x}*{\AllBytesString}%
1010 \ExpectVar\x\AllBytesString
1011 \end{qstest}
1012
1013 \begin{qstest}{uchexdigit}{unescape, uppercase hex digit}
1014 \catcode'\@=11 %
1015 \catcode0=12 %
1016 \def\test#1#2{%
1017 \uccode0=#1\relax
1018 \uppercase{%
1019 \def\x{^~@}%
1020 }%
1021 \Expect*{%
1022 \ifcase\expandafter\PE@TestUcHexDigit\x
1023 true%
1024 \else
1025 false%
1026 \fi
1027 }{#2}%
1028 }%
1029 \def\range#1#2#3{%
1030 \count0=#1\relax
1031 \loop
1032 \ifnum\count0<#2\relax
1033 \test{\count0}{#3}%
1034 \advance\count0 by 1 %
1035 \repeat
1036 }%
1037 \range{0}{47}{false}%
1038 \range{48}{57}{true}%
1039 \range{58}{64}{false}%
1040 \range{65}{70}{true}%
1041 \range{71}{255}{false}%
1042 \end{qstest}
1043
1044 \begin{qstest}{unescapename}{unescapename}
1045 \def\test#1#2{%
1046 \EdefUnescapeName\x{#1}%
1047 \edef\y{#2}%
1048 \@onelevel@sanitize\y
1049 \ExpectVar\x\y
1050 }%
1051 \catcode'\#=12 %
1052 \catcode0=12 %
1053 \test{}{}%
1054 \test{x}{x}%

```

```

1055 \test{xy}{xy}%
1056 \test{#}{#}%
1057 \test{##}{##}%
1058 \test{###}{###}%
1059 \test{####}{####}%
1060 \test{#x}{#x}%
1061 \test{#xy}{#xy}%
1062 \test{#1}{#1}%
1063 \test{#40}{@}%
1064 \test{#400}{@0}%
1065 \test{#4x0}{#4x0}%
1066 \test{#ab}{~^ab}%
1067 \test{#00}{~^@}%
1068 \test{x#40y#40z}{x@y@z}%
1069 \test{#40#40#40#40}{@@@}%
1070 \test{a#x}{a#x}%
1071 \test{a#xy}{a#xy}%
1072 \test{a#1}{a#1}%
1073 \test{a#40}{a@}%
1074 \test{a#400}{a@0}%
1075 \test{#20}{ }%
1076 \test{a#20}{a }%
1077 \test{a#20b}{a b}%
1078 \test{a#20#20#20b}{a \space\space b}%
1079 \end{qstest}
1080
1081 \begin{qstest}{unescapestring}{unescapestring}
1082 \def\test#1#2{%
1083   \EdefUnescapeString\x{#1}%
1084   \edef\y{#2}%
1085   \@onelevel@sanitize\y
1086   \ExpectVar\x\y
1087 }%
1088 \catcode0=12 %
1089 \def\DefChar#1#2{%
1090   \begingroup
1091     \uccode0=#2\relax
1092     \uppercase{\endgroup
1093       \def#1{~^@}%
1094     }%
1095 }%
1096 \DefChar\nul{0}%
1097 \DefChar\one{1}%
1098 \DefChar\bel{8}%
1099 \DefChar\tab{9}%
1100 \DefChar\lf{10}%
1101 \DefChar\ff{12}%
1102 \DefChar\cr{13}%
1103 \DefChar\{92}%
1104 \test{}{}%
1105 \test{a}{a}%
1106 \test{\}{}%
1107 \test{\\\\}{\\}%
1108 \test{\\y}{\y}%
1109 \test{\\000}{\nul}%
1110 \test{\\b}{\bel}%
1111 \test{\\t}{\tab}%
1112 \test{\\n}{\lf}%
1113 \test{\\f}{\ff}%
1114 \test{\\r}{\cr}%
1115 \test{\\}{}%
1116 \test{\\}{}%

```

```

1117 \test{\040}{ }%
1118 \test{\100}{@}%
1119 \test{\40}{ }%
1120 \test{\1}{\one}%
1121 \test{\01}{\one}%
1122 \test{\001}{\one}%
1123 \test{\18}{\one8}%
1124 \test{\018}{\one8}%
1125 \test{\0018}{\one8}%
1126 \test{x\}{x}%
1127 \test{x\\\}{x\}%
1128 \test{x\\y}{x\y}%
1129 \test{x\000}{x\nul}%
1130 \test{x\b}{x\bel}%
1131 \test{x\t}{x\tab}%
1132 \test{x\n}{x\lf}%
1133 \test{x\f}{x\ff}%
1134 \test{x\r}{x\cr}%
1135 \test{x\(){x}%
1136 \test{x\)}{x}%
1137 \test{x\040}{x }%
1138 \test{x\100}{x@}%
1139 \test{x\40}{x }%
1140 \test{x\1}{x\one}%
1141 \test{x\01}{x\one}%
1142 \test{x\001}{x\one}%
1143 \test{x\18}{x\one8}%
1144 \test{x\018}{x\one8}%
1145 \test{x\0018}{x\one8}%
1146 \test{\b\t\n\f\r\(\)\\\\\\000\040}{%
1147 \bel\tab\lf\ff\cr()\nul\space
1148 }%
1149 \test{\lf}{}%
1150 \test{x\lf}{x}%
1151 \test{\cr}{\lf}%
1152 \test{\cr\lf}{\lf}%
1153 \test{\lf}{\lf}%
1154 \test{\lf\cr}{\lf\lf}%
1155 \test{x\cr}{x\lf}%
1156 \test{x\cr\lf}{x\lf}%
1157 \test{x\lf}{x\lf}%
1158 \test{x\lf\cr}{x\lf\lf}%
1159 \test{x\cr\lf y\cr}{xy\lf}%
1160 \end{qstest}
1161 \stop
1162 </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.

¹<http://ftp.ctan.org/tex-archive/>

`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
```

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 T_EX:

```
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 T_EX distribution (teT_EX, miK_T_EX, ...) relies on file name databases, you must refresh these. For example, teT_EX users run `texhash` or `mktextlsr`.

4.5 Some details for the interested

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

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

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

plain T_EX: Run docstrip and extract the files.

L^AT_EX: Generate the documentation.

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

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

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

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

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

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

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

5 History

[2007/02/21 v1.0]

- First version.

[2007/02/25 v1.1]

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

[2007/03/20 v1.2]

- Fix: Wrong year in \ProvidesPackage.

[2007/04/11 v1.3]

- Line ends sanitized.

[2007/04/21 v1.4]

- \EdefUnescapeName and \EdefUnescapeString added.

[2007/08/27 v1.5]

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

[2007/09/09 v1.6]

- Fix in catcode setup.

[2007/10/27 v1.7]

- More efficient `\EdefSanitize`.

[2007/11/11 v1.8]

- Use of package `pdftexcmds` for LuaTeX support.

[2010/03/01 v1.9]

- Compatibility with `ini-TeX`.

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; plain numbers refer to the code lines where the entry is used.

Symbols			
<code>\#</code>	161, 297, 322, 682, 744, 864, 891, 966, 1051	<code>\~</code>	892
<code>\\$</code>	160	Numbers	
<code>\%</code>	747, 889	<code>\0</code>	895, 896, 897
<code>\(</code>	656, 897	<code>\1</code>	902
<code>\)</code>	657, 897	<code>\2</code>	903, 904, 905, 906
<code>\@</code>	683, 740, 1014	<code>\3</code>	907, 908, 909, 910
<code>\@ReturnAfterFi</code>	147, <u>152</u>	<code>\8</code>	261
<code>\@backslashchar</code>	871	<code>\9</code>	262
<code>\@firstofone</code>	691, 694	<code>_</code>	136, 658, 748
<code>\@gobble</code>	688, 696	A	
<code>\@ifundefined</code>	777, 791, 799	<code>\advance</code>	477, 513, 514, 721, 729, 836, 1034
<code>\@ne</code>	188, 191, 236, 239, 243, 353, 356	<code>\aftergroup</code>	26
<code>\@onelevel@sanitize</code>	118, 814, 815, 857, 884, 913, 975, 1048, 1085	<code>\AllBytes</code>	824, 830, 833, 927, 928, 935, 936, 941, 942, 1001, 1002, 1007, 1008
<code>\@percentchar</code>	889	<code>\AllBytesHex</code>	839, 857, 860, 929, 930, 933, 934
<code>\@undefined</code>	52, 785, 786, 787, 788, 917, 919	<code>\AllBytesHexLC</code>	859, 939, 940
<code>\@whilenum</code>	827	<code>\AllBytesName</code>	862, 865, 884, 1003, 1004
<code>\[</code>	745	<code>\AllBytesString</code>	886, 913, 1009, 1010
<code>\]</code>	144, 289, 439, 658, 741, 893, 900, 1103, 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, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1149, 1150, 1159	B	
<code>\{</code>	680, 742	<code>\begin</code>	927, 933, 939, 945, 950, 956, 965, 971, 1001, 1007, 1013, 1044, 1081
<code>\}</code>	346, 681, 743	<code>\bel</code>	1098, 1110, 1130, 1147
<code>\]</code>	746	<code>\body</code>	700, 704
<code>\ </code>	288, 293, 888, 890	C	
		<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, 65, 68, 69, 70, 71, 75, 76, 77, 78, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 136, 160, 161, 288, 289, 680, 681, 682, 683, 718, 727, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 826, 864, 890, 891, 892, 893, 966, 1014, 1015, 1051, 1052, 1088	\ifnum 235, 238, 241, 242, 352, 355, 556, 592, 600, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 651, 654, 656, 657, 658, 720, 728, 829, 1032
\count 1030, 1032, 1033, 1034	\ifPE@etex 442, 448, 466, 492, 578, 625, 635, 667
\count@ 685, 714, 718, 720, 721, 725, 727, 728, 729, 825, 827, 828, 829, 836	\ifx 11, 14, 18, 44, 52, 55, 102, 112, 144, 171, 175, 270, 275, 386, 392, 444, 460, 533, 534, 590, 649, 684, 687, 690, 693, 732, 811, 819, 981
\countdef 685	\immediate 20, 46
\cr .. 1102, 1114, 1134, 1147, 1151, 1152, 1154, 1155, 1156, 1158, 1159	\IncludeTests 807
\csname 10, 18, 44, 60, 67, 102, 105, 112, 123, 386, 392, 415, 444, 684, 687, 690, 693, 732, 754	\input 387, 733
	\iterate 701, 703, 705
D	L
\DefChar 1089, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103	\lccode 828
\detokenize ... 127, 131, 916, 917, 923	\lf 1100, 1112, 1132, 1147, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159
\dimen . 475, 476, 477, 480, 481, 510, 511, 512, 513, 514, 518, 519, 520	\LoadCommand 733, 750
\dimexpr 468, 470, 495, 498, 506	\LogTests 808
\do 827	\loop 699, 715, 726, 1031
	\lowercase 832, 858
E	M
\EdefEscapeHex 2, 393, 420, 928	\makeatletter 759, 822
\EdefEscapeName 402, 426, 1002	\meaning 114
\EdefEscapeString 407, 429, 1008	
\EdefSanitize 3, 101, 134, 154, 254, 394, 399, 403, 408	N
\EdefUnescapeHex 398, 423, 934, 940, 946, 951, 957, 967, 973	\NeedsTeXFormat 758
\EdefUnescapeName 2, 153, 1046	\newcommand ... 810, 839, 859, 862, 886
\EdefUnescapeString 2, 253, 1083	\newif 442
\empty 13, 14	\next 705, 707, 709
\end 755, 931, 937, 943, 948, 954, 963, 969, 999, 1005, 1011, 1042, 1079, 1160	\nul 1096, 1109, 1129, 1147
\endcsname . 10, 18, 44, 60, 67, 102, 105, 112, 123, 386, 392, 415, 444, 684, 687, 690, 693, 732, 754	\number 518, 519, 520
\endinput 26, 433	\numexpr 468, 470, 495, 498, 505, 506, 918, 919, 924
\escapechar 437	
\eTeX 793	O
\Expect 819, 929, 935, 941, 947, 952, 961, 1003, 1009, 1021	\one 1097, 1120, 1121, 1122, 1123, 1124, 1125, 1140, 1141, 1142, 1143, 1144, 1145
\ExpectVar 810, 930, 936, 942, 953, 962, 968, 976, 1004, 1010, 1049, 1086	\org@detokenize 916, 923
	\org@numexpr 918, 924
F	P
\ff 1101, 1113, 1133, 1147	\PackageError 778, 792, 800
	\PackageInfo 23
G	\pdf@escapehex 421
\gdef 162, 170, 824	\pdf@escapename 427
	\pdf@escapestring 430
	\pdf@unescapehex 424
	\pdfescape 762, 766, 770, 779
	\pdfescapehex 785
	\pdfescapename 787
	\pdfescapestring 788
	\pdfunescapehex 786, 979
I	\PE@@NormalizeLineEnd 266, 269
\ifcase 184, 185, 363, 371, 487, 540, 553, 554, 593, 1022	\PE@@OctChar 494, 502
	\PE@AtEnd 80, 81, 432, 677

\PE@backslash [436](#), [503](#), [517](#)
 \PE@CheckEndBackslash [343](#)
 \PE@DeHex [528](#), [532](#)
 \PE@DeName [166](#), [170](#), [199](#), [203](#)
 \PE@DeString [297](#), [383](#)
 \PE@edefbabel . [413](#), [421](#), [424](#), [427](#), [430](#)
 \PE@EscapeHex [396](#), [448](#)
 \PE@EscapeName [405](#), [578](#)
 \PE@EscapeNameAdd
 [596](#), [601](#), [615](#), [623](#), [625](#)
 \PE@EscapeNameHashChar
 [603](#), [604](#), [605](#), [606](#), [607](#),
 [608](#), [609](#), [610](#), [611](#), [612](#), [613](#), [622](#)
 \PE@EscapeNameTokens .. [580](#), [585](#), [589](#)
 \PE@EscapeString [410](#), [635](#)
 \PE@EscapeStringAdd
 [656](#), [657](#), [658](#), [660](#), [667](#)
 \PE@EscapeStringTokens [637](#), [643](#), [648](#)
 \PE@etexttrue [446](#)
 \PE@hash [435](#), [596](#), [601](#), [623](#)
 \PE@HexChar [462](#), [466](#), [597](#), [602](#)
 \PE@HexDigit .. [468](#), [469](#), [480](#), [481](#), [485](#)
 \PE@InitUccodeHexDigit [164](#), [210](#), [526](#)
 \PE@next [173](#), [178](#), [199](#),
 [203](#), [207](#), [272](#), [276](#), [278](#), [281](#),
 [364](#), [366](#), [368](#), [372](#), [374](#), [376](#),
 [535](#), [541](#), [545](#), [564](#), [567](#), [573](#), [576](#)
 \PE@NormalizeLineEnd [256](#), [264](#)
 \PE@OctAll [366](#), [372](#), [374](#), [378](#)
 \PE@OctChar [492](#), [652](#), [655](#)
 \PE@OctI [362](#)
 \PE@OctII [364](#), [370](#)
 \PE@onelevel@sanitize
 [107](#), [113](#), [118](#), [130](#), [157](#), [258](#)
 \PE@result [165](#), [168](#), [172](#),
 [177](#), [198](#), [202](#), [265](#), [267](#), [271](#),
 [274](#), [381](#), [454](#), [456](#), [478](#), [479](#),
 [515](#), [516](#), [527](#), [530](#), [563](#), [584](#),
 [586](#), [629](#), [630](#), [642](#), [645](#), [671](#), [672](#)
 \PE@sanitize [134](#)
 \PE@SanitizeSpaceOther
 [139](#), [155](#), [255](#), [395](#), [404](#), [409](#)
 \PE@space@other [135](#), [146](#)
 \PE@space@space [138](#), [557](#)
 \PE@SpaceToOther [140](#), [142](#)
 \PE@strip@prefix [114](#), [116](#)
 \PE@temp [195](#), [198](#), [557](#), [560](#), [563](#)
 \PE@testA [181](#),
 [184](#), [193](#), [538](#), [540](#), [550](#), [553](#), [555](#)
 \PE@testB . [182](#), [185](#), [193](#), [551](#), [554](#), [555](#)
 \PE@TestOctDigit [351](#), [363](#), [371](#)
 \PE@TestUcHexDigit
 [184](#), [185](#), [234](#), [540](#), [553](#), [554](#), [1022](#)
 \PE@ToHex [450](#), [455](#), [459](#)
 \PE@UnescapeHex [400](#), [524](#)
 \PE@UnescapeName [156](#), [159](#)
 \PE@UnescapeString [257](#), [290](#)
 \ProvidesFile [761](#), [765](#), [769](#), [773](#)
 \ProvidesPackage [15](#), [61](#)

R

\range [1029](#), [1037](#), [1038](#), [1039](#), [1040](#), [1041](#)

\RangeCatcodeInvalid
 [724](#), [736](#), [737](#), [738](#), [739](#)
 \repeat [699](#), [711](#), [722](#), [730](#), [1035](#)
 \RequirePackage [389](#), [806](#), [921](#)
 \RestoreCatcodes .. [713](#), [716](#), [717](#), [751](#)

S

\space .. [830](#), [953](#), [968](#), [983](#), [1078](#), [1147](#)
 \stop [781](#), [795](#), [803](#), [1161](#)

T

\tab [1099](#), [1111](#), [1131](#), [1147](#)
 \Test [735](#), [753](#)
 \test .. [972](#), [989](#), [992](#), [993](#), [994](#), [995](#),
 [996](#), [997](#), [998](#), [1016](#), [1033](#), [1045](#),
 [1053](#), [1054](#), [1055](#), [1056](#), [1057](#),
 [1058](#), [1059](#), [1060](#), [1061](#), [1062](#),
 [1063](#), [1064](#), [1065](#), [1066](#), [1067](#),
 [1068](#), [1069](#), [1070](#), [1071](#), [1072](#),
 [1073](#), [1074](#), [1075](#), [1076](#), [1077](#),
 [1078](#), [1082](#), [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](#),
 [1137](#), [1138](#), [1139](#), [1140](#), [1141](#),
 [1142](#), [1143](#), [1144](#), [1145](#), [1146](#),
 [1149](#), [1150](#), [1151](#), [1152](#), [1153](#),
 [1154](#), [1155](#), [1156](#), [1157](#), [1158](#), [1159](#)
 \the [68](#), [69](#),
 [70](#), [71](#), [82](#), [495](#), [498](#), [505](#), [506](#), [718](#)
 \TMP@EnsureCode [79](#), [86](#), [87](#), [88](#), [89](#), [90](#),
 [91](#), [92](#), [93](#), [94](#), [95](#), [96](#), [97](#), [98](#), [99](#), [100](#)
 \typeout .. [816](#), [984](#), [985](#), [986](#), [987](#), [988](#)

U

\uccode [98](#), [99](#), [100](#),
 [193](#), [197](#), [211](#), [212](#), [213](#), [214](#),
 [215](#), [216](#), [217](#), [218](#), [219](#), [220](#),
 [221](#), [222](#), [223](#), [224](#), [225](#), [226](#),
 [227](#), [228](#), [229](#), [230](#), [231](#), [232](#),
 [261](#), [262](#), [379](#), [555](#), [556](#), [1017](#), [1091](#)
 \uppercase [180](#), [194](#),
 [284](#), [380](#), [537](#), [549](#), [559](#), [1018](#), [1092](#)

W

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

X

\x . [10](#), [11](#), [14](#), [19](#), [23](#), [25](#), [45](#), [50](#), [60](#),
 [66](#), [74](#), [137](#), [263](#), [285](#), [438](#), [441](#),
 [928](#), [929](#), [930](#), [934](#), [935](#), [936](#),
 [940](#), [941](#), [942](#), [946](#), [947](#), [951](#),
 [952](#), [953](#), [957](#), [961](#), [962](#), [967](#),
 [968](#), [973](#), [976](#), [979](#), [981](#), [1002](#),
 [1003](#), [1004](#), [1008](#), [1009](#), [1010](#),
 [1019](#), [1022](#), [1046](#), [1049](#), [1083](#), [1086](#)

Y

\y [958](#), [960](#), [974](#), [975](#), [976](#), [980](#), [981](#),
 [1047](#), [1048](#), [1049](#), [1084](#), [1085](#), [1086](#)

	Z	
\z 959, 961, 962	220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231,
\z@	... 186, 193, 197, 217, 218, 219,	232, 245, 248, 358, 379, 555, 556