

The pdfescape package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/04/21 v1.4

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
2	Implementation	2
2.1	Catcodes	2
2.2	Reload check and package identification	3
2.3	Sanitizing	4
2.3.1	Space characters	4
2.3.2	Space normalization	5
2.4	<code>\EdefUnescapeName</code>	5
2.5	<code>\EdefUnescapeString</code>	7
2.6	User macros (pdfTeX analogues)	9
2.7	Help macros	10
2.7.1	Characters	10
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	12
2.8.4	Conversion to PDF name	13
2.8.5	Conversion to PDF string	14
3	Test	15
3.1	Test with <code>\pdfescape...</code> commands	15
3.2	Test without <code>\pdfescape...</code> , with ϵ -TeX	15
3.3	Test without <code>\pdfescape...</code> and ϵ -TeX	15
3.4	Check/ensure test preconditions	15
3.4.1	Check <code>\pdfescape...</code>	15
3.4.2	Check ϵ -TeX	16
3.5	Common part	16
4	Installation	22
4.1	Download	22
4.2	Bundle installation	22
4.3	Package installation	22
4.4	Refresh file name databases	23
4.5	Some details for the interested	23

5 History	23
[2007/02/21 v1.0]	23
[2007/02/25 v1.1]	23
[2007/03/20 v1.2]	24
[2007/04/11 v1.3]	24
[2007/04/21 v1.4]	24
6 Index	24

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.

2 Implementation

```
1 ⟨*package⟩
```

2.1 Catcodes

```
2 \expandafter\edef\curname PE@AtEnd\endcurname{%
3   \catcode64 \the\catcode64\relax
4 }
```

```

5 \catcode64 11 % @
6 \def\PE@EnsureCode#1#2#3{%
7   \edef\PE@AtEnd{%
8     \PE@AtEnd
9     #1#2 \the#1#2\relax
10  }%
11  #1#2 #3\relax
12 }
13 \PE@EnsureCode\catcode{0}{12}% ^^@
14 \PE@EnsureCode\catcode{34}{12}% "
15 \PE@EnsureCode\catcode{42}{12}% *
16 \PE@EnsureCode\catcode{45}{12}% -
17 \PE@EnsureCode\catcode{46}{12}% .
18 \PE@EnsureCode\catcode{60}{12}% <
19 \PE@EnsureCode\catcode{61}{12}% =
20 \PE@EnsureCode\catcode{62}{12}% >
21 \PE@EnsureCode\catcode{94}{7}% ^
22 \PE@EnsureCode\catcode{96}{12}% `
23 \PE@EnsureCode\uccode{34}{0}% "
24 \PE@EnsureCode\uccode{48}{0}% 0
25 \PE@EnsureCode\uccode{61}{0}% =

```

2.2 Reload check and package identification

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

```

26 \begingroup
27   \catcode44 12 % ,
28   \catcode45 12 % -
29   \catcode46 12 % .
30   \catcode58 12 % :
31   \catcode64 11 % @
32   \expandafter\let\expandafter\x\csname ver@pdfescape.sty\endcsname
33   \ifcase 0%
34     \ifx\x\relax % plain
35     \else
36       \ifx\x\empty % LaTeX
37       \else
38         1%
39       \fi
40     \fi
41   \else
42     \expandafter\ifx\csname PackageInfo\endcsname\relax
43     \def\x#1#2{%
44       \immediate\write-1{Package #1 Info: #2.}%
45     }%
46   \else
47     \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
48   \fi
49   \x{pdfescape}{The package is already loaded}%
50 \endgroup
51 \expandafter\endinput
52 \fi
53 \endgroup

```

Package identification:

```

54 \begingroup
55   \catcode44 12 % ,
56   \catcode45 12 % -
57   \catcode46 12 % .
58   \catcode58 12 % :
59   \catcode64 11 % @
60   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
61     \def\x#1#2#3[#4]{\endgroup

```

```

62     \immediate\write-1{Package: #3 #4}%
63     \xdef#1{#4}%
64 }%
65 \else
66     \def\x#1#2[#3]{\endgroup
67     #2[#{#3}]%
68     \ifx#1\relax
69     \xdef#1{#3}%
70     \fi
71 }%
72 \fi
73 \expandafter\x\csname ver@pdfescape.sty\endcsname
74 \ProvidesPackage{pdfescape}%
75 [2007/04/21 v1.4 Provides hex, PDF name and string conversions (H0)]

```

2.3 Sanitizing

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

```

76 \begingroup\expandafter\expandafter\expandafter\endgroup
77 \expandafter\ifx\csname detokenize\endcsname\relax
78 \long\def\PE@sanitize#1#2{%
79     \begingroup
80     \csname @safe@activetrue\endcsname
81     \edef#1{#2}%
82     \PE@onelevel@sanitize#1%
83     \expandafter\endgroup
84     \expandafter\def\expandafter#1\expandafter{#1}%
85 }%
86 \begingroup\expandafter\expandafter\expandafter\endgroup
87 \expandafter\ifx\csname @onelevel@sanitize\endcsname\relax
88     \def\PE@onelevel@sanitize#1{%
89         \edef#1{\expandafter\PE@strip@prefix\meaning#1}%
90     }%
91     \def\PE@strip@prefix#1>{%}%
92 \else
93     \let\PE@onelevel@sanitize\@onelevel@sanitize
94 \fi
95 \else
96 \long\def\PE@sanitize#1#2{%
97     \begingroup
98     \csname @safe@activetrue\endcsname
99     \edef#1{#2}%
100     \PE@onelevel@sanitize#1%
101     \expandafter\endgroup
102     \expandafter\def\expandafter#1\expandafter{#1}%
103 }%
104 \def\PE@onelevel@sanitize#1{%
105     \edef#1{\detokenize\expandafter{#1}}%
106 }%
107 \fi

```

2.3.1 Space characters

`\PE@space@other`

```

108 \begingroup
109 \catcode'\ =12\relax%
110 \def\x{\endgroup\def\PE@space@other{ }}\x\relax

```

`\PE@space@space`

```

111 \def\PE@space@space{ }

```

2.3.2 Space normalization

\PE@SanitizeSpaceOther

```
112 \def\PE@SanitizeSpaceOther#1{%  
113   \edef#1{\expandafter\PE@SpaceToOther#1 \relax}%  
114 }
```

\PE@SpaceToOther

```
115 \def\PE@SpaceToOther#1 #2\relax{%  
116   #1%  
117   \ifx\#2\%  
118     \else  
119       \PE@space@other  
120       \@ReturnAfterFi{%  
121         \PE@SpaceToOther#2\relax  
122       }%  
123   \fi  
124 }
```

\@ReturnAfterFi

```
125 \long\def\@ReturnAfterFi#1\fi{\fi#1}
```

2.4 \EdefUnescapeName

\EdefUnescapeName

```
126 \def\EdefUnescapeName#1#2{%  
127   \PE@sanitize#1{#2}%  
128   \PE@SanitizeSpaceOther#1%  
129   \PE@UnescapeName#1%  
130   \PE@onelevel@sanitize#1%  
131 }
```

\PE@UnescapeName

```
132 \begingroup  
133   \catcode'\$=6 % hash  
134   \catcode'\#=12 % other  
135   \gdef\PE@UnescapeName$1{%  
136     \begingroup  
137       \PE@InitUccodeHexDigit  
138       \def\PE@result{}%  
139       \expandafter\PE@DeName$1#\relax\relax  
140       \expandafter\endgroup  
141       \expandafter\def\expandafter$1\expandafter{\PE@result}%  
142     }%  
143     \gdef\PE@DeName$1#$2$3{%  
144       \ifx\relax$2%  
145         \edef\PE@result{\PE@result$1}%  
146         \let\PE@next\relax  
147       \else  
148         \ifx\relax$3%  
149           % wrong escape sequence in input  
150           \edef\PE@result{\PE@result$1#}%  
151           \let\PE@next\relax  
152         \else  
153           \uppercase{%  
154             \def\PE@testA{$2}%  
155             \def\PE@testB{$3}%  
156           }%  
157           \ifcase\ifcase\expandafter\PE@TestUcHexDigit\PE@testA  
158             \ifcase\expandafter\PE@TestUcHexDigit\PE@testB  
159             \z@
```

```

160             \else
161             \@ne
162             \fi
163             \else
164             \@ne
165             \fi
166             \uccode\z@="\PE@testA\PE@testB\relax
167             \uppercase{%
168             \def\PE@temp{^^@}%
169             }%
170             \uccode\z@=\z@
171             \edef\PE@result{\PE@result$1\PE@temp}%
172             \let\PE@next\PE@DeName
173         \else
174             % wrong escape sequence in input
175             \edef\PE@result{\PE@result$1#}%
176             \def\PE@next{\PE@DeName$2$3}%
177         \fi
178     \fi
179 \fi
180 \PE@next
181 }%
182 \endgroup

```

\PE@InitUccodeHexDigit

```

183 \def\PE@InitUccodeHexDigit{%
184   \uccode'a='A\relax
185   \uccode'b='B\relax
186   \uccode'c='C\relax
187   \uccode'd='D\relax
188   \uccode'e='E\relax
189   \uccode'f='F\relax
190   \uccode'A=\z@
191   \uccode'B=\z@
192   \uccode'C=\z@
193   \uccode'D=\z@
194   \uccode'E=\z@
195   \uccode'F=\z@
196   \uccode'0=\z@
197   \uccode'1=\z@
198   \uccode'2=\z@
199   \uccode'3=\z@
200   \uccode'4=\z@
201   \uccode'5=\z@
202   \uccode'6=\z@
203   \uccode'7=\z@
204   \uccode'8=\z@
205   \uccode'9=\z@
206 }

```

\PE@TestUcHexDigit

```

207 \def\PE@TestUcHexDigit#1{%
208   \ifnum'#1<48 % 0
209     \@ne
210   \else
211     \ifnum'#1>70 % F
212       \@ne
213     \else
214       \ifnum'#1>57 % 9
215         \ifnum'#1<65 % A
216           \@ne
217         \else

```

```

218         \z@
219         \fi
220     \else
221         \z@
222         \fi
223     \fi
224 \fi
225 }

```

2.5 \EdefUnescapeString

\EdefUnescapeString

```

226 \def\EdefUnescapeString#1#2{%
227   \PE@sanitize#1{#2}%
228   \PE@SanitizeSpaceOther#1%
229   \PE@NormalizeLineEnd#1%
230   \PE@UnescapeString#1%
231   \PE@onelevel@sanitize#1%
232 }

233 \begingroup
234   \uccode'\8=10 % lf
235   \uccode'\9=13 % cr
236 \def\x#1#2{\endgroup

```

\PE@NormalizeLineEnd

```

237   \def\PE@NormalizeLineEnd##1{%
238     \def\PE@result{}%
239     \expandafter\PE@@NormalizeLineEnd##1#2\relax
240     \let##1\PE@result
241   }%

```

\PE@@NormalizeLineEnd

```

242   \def\PE@@NormalizeLineEnd##1#2##2{%
243     \ifx\relax##2%
244       \edef\PE@result{\PE@result##1}%
245       \let\PE@next\relax
246     \else
247       \edef\PE@result{\PE@result##1#1}%
248       \ifx#1##2% lf
249         \let\PE@next\PE@@NormalizeLineEnd
250       \else
251         \def\PE@next{\PE@@NormalizeLineEnd##2}%
252       \fi
253     \fi
254     \PE@next
255   }%
256 }%
257 \uppercase{%
258   \x 89%
259 }

260 \begingroup
261   \catcode'\|=0 %
262   \catcode'\|=12 %

```

\PE@UnescapeString

```

263 |gdef|PE@UnescapeString#1{%
264   |begingroup
265     |def|PE@result{}%
266     |expandafter|PE@DeString#1|\relax
267     |expandafter|endgroup

```

```

268     |expandafter|def|expandafter#1|expandafter{|PE@result}%
269 }%

\PE@DeString

270 |gdef|PE@DeString#1\#2{%
271     |ifx|relax#2%
272     |edef|PE@result{|PE@result#1}%
273     |let|PE@next|relax
274     |else
275     |if n#2%
276     |uccode|z@=10 %
277     |else|if r#2%
278     |uccode|z@=13 %
279     |else|if t#2%
280     |uccode|z@=9 %
281     |else|if b#2%
282     |uccode|z@=8 %
283     |else|if f#2%
284     |uccode|z@=12 %
285     |else
286     |uccode|z@=|z@
287     |fi|fi|fi|fi|fi
288     |ifnum|uccode|z@>|z@
289     |uppercase{%
290     |edef|PE@temp{^^@}%
291     }%
292     |edef|PE@result{|PE@result#1|PE@temp}%
293     |let|PE@next|PE@DeString
294     |else
295     |if\#2% backslash
296     |edef|PE@result{|PE@result#1}%
297     |let|PE@next|PE@CheckEndBackslash
298     |else
299     |ifnum'#2=10 % linefeed
300     |edef|PE@result{|PE@result#1}%
301     |let|PE@next|PE@DeString
302     |else
303     |ifcase|PE@TestOctDigit#2%
304     |edef|PE@result{|PE@result#1}%
305     |def|PE@next{|PE@OctI#2}%
306     |else
307     |edef|PE@result{|PE@result#1#2}%
308     |let|PE@next|PE@DeString
309     |fi
310     |fi
311     |fi
312     |fi
313     |fi
314     |PE@next
315 }%

\PE@CheckEndBackslash

316 |gdef|PE@CheckEndBackslash#1{%
317     |ifx|relax#1%
318     |else
319     |edef|PE@result{|PE@result\}%
320     |expandafter|PE@DeString|expandafter#1%
321     |fi
322 }%

323 |endgroup

\PE@TestOctDigit

```

```

324 \def\PE@TestOctDigit#1{%
325   \ifnum'#1<48 % 0
326     \@ne
327   \else
328     \ifnum'#1>55 % 7
329       \@ne
330     \else
331       \z@
332     \fi
333   \fi
334 }

```

\PE@OctI

```

335 \def\PE@OctI#1#2{%
336   \ifcase\PE@TestOctDigit#2%
337     \def\PE@next{\PE@OctII{#1#2}}%
338   \else
339     \def\PE@next{\PE@OctAll{#1#2}}%
340   \fi
341   \PE@next
342 }

```

\PE@OctII

```

343 \def\PE@OctII#1#2{%
344   \ifcase\PE@TestOctDigit#2%
345     \def\PE@next{\PE@OctAll{#1#2}}%
346   \else
347     \def\PE@next{\PE@OctAll{#1}#2}%
348   \fi
349   \PE@next
350 }

```

\PE@OctAll

```

351 \def\PE@OctAll#1{%
352   \uccode\z@='#1\relax
353   \uppercase{%
354     \edef\PE@result{\PE@result^^@}%
355   }%
356   \PE@DeString
357 }

```

2.6 User macros (pdfTeX analogues)

```

358 \begingroup\expandafter\expandafter\expandafter\endgroup
359 \expandafter\ifx\csname pdfescapehex\endcsname\relax

```

\EdefEscapeHex

```

360 \long\def\EdefEscapeHex#1#2{%
361   \PE@sanitize#1{#2}%
362   \PE@SanitizeSpaceOther#1%
363   \PE@EscapeHex#1%
364 }%

```

\EdefUnescapeHex

```

365 \def\EdefUnescapeHex#1#2{%
366   \PE@sanitize#1{#2}%
367   \PE@UnescapeHex#1%
368 }%

```

\EdefEscapeName

```

369 \long\def\EdefEscapeName#1#2{%

```

```

370     \PE@sanitize#1{#2}%
371     \PE@SanitizeSpaceOther#1%
372     \PE@EscapeName#1%
373 }%

\EdefEscapeString

374 \long\def\EdefEscapeString#1#2{%
375     \PE@sanitize#1{#2}%
376     \PE@SanitizeSpaceOther#1%
377     \PE@EscapeString#1%
378 }%

379 \else

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

380 \long\def\PE@edefbabel#1#2#3{%
381     \begingroup
382     \csname @save@activetrue\endcsname
383     \edef#1{#2{#3}}%
384     \expandafter\endgroup
385     \expandafter\def\expandafter#1\expandafter{#1}%
386 }%

\EdefEscapeHex

387 \long\def\EdefEscapeHex#1#2{%
388     \PE@edefbabel#1\pdfescapehex{#2}%
389 }%

\EdefUnescapeHex

390 \def\EdefUnescapeHex#1#2{%
391     \PE@edefbabel#1\pdfunescapehex{#2}%
392 }%

\EdefEscapeName

393 \long\def\EdefEscapeName#1#2{%
394     \PE@edefbabel#1\pdfescapename{#2}%
395 }%

\EdefEscapeString

396 \long\def\EdefEscapeString#1#2{%
397     \PE@edefbabel#1\pdfescapestring{#2}%
398 }%

399 \PE@AtEnd
400 \expandafter\endinput
401 \fi

```

2.7 Help macros

2.7.1 Characters

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

```

\PE@hash

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

\PE@backslash

403 \begingroup
404 \escapechar=-1 %
405 \edef\x{\endgroup
406     \def\noexpand\PE@backslash{\string\\}%
407 }
408 \x

```

2.7.2 Switch for ε -T_EX

```
409 \newif\ifPE@etex
410 \begingroup\expandafter\expandafter\expandafter\endgroup
411 \expandafter\ifx\csname numexpr\endcsname\relax
412 \else
413   \PE@etextrue
414 \fi
```

2.8 Conversions

2.8.1 Conversion to hex string

\PE@EscapeHex

```
415 \ifPE@etex
416   \def\PE@EscapeHex#1{%
417     \edef#1{\expandafter\PE@ToHex#1\relax}%
418   }%
419 \else
420   \def\PE@EscapeHex#1{%
421     \def\PE@result{%
422       \expandafter\PE@ToHex#1\relax
423       \let#1\PE@result
424     }%
425 \fi
```

\PE@ToHex

```
426 \def\PE@ToHex#1{%
427   \ifx\relax#1%
428   \else
429     \PE@HexChar{#1}%
430     \expandafter\PE@ToHex
431   \fi
432 }%
```

\PE@HexChar

```
433 \ifPE@etex
434   \def\PE@HexChar#1{%
435     \PE@HexDigit{\numexpr\dimexpr.0625\dimexpr'#1sp\relax\relax\relax}%
436     \PE@HexDigit{%
437       \numexpr'#1-16*\dimexpr.0625\dimexpr'#1sp\relax\relax\relax
438     }%
439   }%
440 \else
441   \def\PE@HexChar#1{%
442     \dimen0='#1sp%
443     \dimen2=.0625\dimen0 %
444     \advance\dimen0-16\dimen2 %
445     \edef\PE@result{%
446       \PE@result
447       \PE@HexDigit{\dimen2 }%
448       \PE@HexDigit{\dimen0 }%
449     }%
450   }%
451 \fi
```

\PE@HexDigit

```
452 \def\PE@HexDigit#1{%
453   \expandafter\string
454   \ifcase#1%
455     0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or 9\or
456     A\or B\or C\or D\or E\or F%
457   \fi
458 }
```

2.8.2 Character code to octal number

\PE@OctChar

```

459 \ifPE@etex
460 \def\PE@OctChar#1{%
461   \expandafter\PE@@OctChar
462     \the\numexpr\dimexpr.015625\dimexpr'#1sp\relax\relax
463     \expandafter\relax
464     \expandafter\relax
465     \the\numexpr\dimexpr.125\dimexpr'#1sp\relax\relax\relax
466     \relax
467     #1%
468 }%
469 \def\PE@@OctChar#1\relax#2\relax#3{%
470   \PE@backslash
471   #1%
472   \the\numexpr#2-8*#1\relax
473   \the\numexpr\dimexpr'#3sp\relax-8*#2\relax
474 }%
475 \else
476 \def\PE@OctChar#1{%
477   \dimen0='#1sp%
478   \dimen2=.125\dimen0 %
479   \dimen4=.125\dimen2 %
480   \advance\dimen0-8\dimen2 %
481   \advance\dimen2-8\dimen4 %
482   \edef\PE@result{%
483     \PE@result
484     \PE@backslash
485     \number\dimen4 %
486     \number\dimen2 %
487     \number\dimen0 %
488   }%
489 }%
490 \fi

```

2.8.3 Unpack hex string

\PE@UnescapeHex

```

491 \def\PE@UnescapeHex#1{%
492   \begingroup
493     \PE@InitUccodeHexDigit
494     \def\PE@result{}%
495     \expandafter\PE@DeHex#1\relax\relax
496   \expandafter\endgroup
497   \expandafter\def\expandafter#1\expandafter{\PE@result}%
498 }

```

\PE@DeHex

```

499 \def\PE@DeHex#1#2{%
500   \ifx#2\relax
501     \ifx#1\relax
502       \let\PE@next\relax
503     \else
504       \uppercase{%
505         \def\PE@testA{#1}%
506       }%
507       \ifcase\expandafter\PE@TestUcHexDigit\PE@testA
508         \def\PE@next{%
509           \PE@DeHex#10\relax\relax
510         }%
511       \else

```

```

512         \let\PE@next\relax
513     \fi
514 \fi
515 \else
516     \uppercase{%
517         \def\PE@testA{#1}%
518         \def\PE@testB{#2}%
519     }%
520     \ifcase\expandafter\PE@TestUcHexDigit\PE@testA
521     \ifcase\expandafter\PE@TestUcHexDigit\PE@testB
522         \uccode\z@="\PE@testA\PE@testB\relax
523         \ifnum\uccode\z@=32 %
524             \let\PE@temp\PE@space@space
525         \else
526             \uppercase{%
527                 \def\PE@temp{^^@}%
528             }%
529         \fi
530         \edef\PE@result{\PE@result\PE@temp}%
531         \let\PE@next\PE@DeHex
532     \else
533         % invalid input sequence
534         \def\PE@next{%
535             \PE@DeHex#1%
536         }%
537     \fi
538 \else
539     % invalid input sequence
540     \def\PE@next{\PE@DeHex#2}%
541 \fi
542 \fi
543 \PE@next
544 }

```

2.8.4 Conversion to PDF name

\PE@EscapeName

```

545 \ifPE@etex
546     \def\PE@EscapeName#1{%
547         \edef#1{\expandafter\PE@EscapeNameTokens#1\relax}%
548     }%
549 \else
550     \def\PE@EscapeName#1{%
551         \def\PE@result{%
552             \expandafter\PE@EscapeNameTokens#1\relax
553             \let#1\PE@result
554         }%
555     \fi

```

\PE@EscapeNameTokens

```

556 \def\PE@EscapeNameTokens#1{%
557     \ifx\relax#1%
558     \else
559         \ifnum'#1<33 %
560             \ifcase'#1 %
561                 % drop illegal zero
562             \else
563                 \PE@EscapeNameAdd\PE@hash
564                 \PE@HexChar#1%
565             \fi
566         \else
567             \ifnum'#1>126 %

```

```

568     \PE@EscapeNameAdd\PE@hash
569     \PE@HexChar#1%
570     \else \ifnum'#1=35 \PE@EscapeNameHashChar 23% #
571     \else\ifnum'#1=37 \PE@EscapeNameHashChar 25% %
572     \else\ifnum'#1=40 \PE@EscapeNameHashChar 28% (
573     \else\ifnum'#1=41 \PE@EscapeNameHashChar 29% )
574     \else\ifnum'#1=47 \PE@EscapeNameHashChar 2F% /
575     \else\ifnum'#1=60 \PE@EscapeNameHashChar 3C% <
576     \else\ifnum'#1=62 \PE@EscapeNameHashChar 3E% >
577     \else\ifnum'#1=91 \PE@EscapeNameHashChar 5B% [
578     \else\ifnum'#1=93 \PE@EscapeNameHashChar 5D% ]
579     \else\ifnum'#1=123 \PE@EscapeNameHashChar 7B% {
580     \else\ifnum'#1=125 \PE@EscapeNameHashChar 7D% }
581     \else
582     \PE@EscapeNameAdd{#1}%
583     \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
584     \fi
585     \fi
586     \expandafter\PE@EscapeNameTokens
587     \fi
588 }%
589 \def\PE@EscapeNameHashChar#1#2{%
590 \PE@EscapeNameAdd{\PE@hash\string#1\string#2}%
591 }%

```

\PE@EscapeNameAdd

```

592 \ifPE@etex
593 \def\PE@EscapeNameAdd#1{#1}%
594 \else
595 \def\PE@EscapeNameAdd#1{%
596 \edef\PE@result{%
597 \PE@result
598 #1%
599 }%
600 }%
601 \fi

```

2.8.5 Conversion to PDF string

\PE@EscapeString

```

602 \ifPE@etex
603 \def\PE@EscapeString#1{%
604 \edef#1{\expandafter\PE@EscapeStringTokens#1\relax}%
605 }%
606 \else
607 \def\PE@EscapeString#1{%
608 \begingroup
609 \def\PE@result{}%
610 \expandafter\PE@EscapeStringTokens#1\relax
611 \expandafter\endgroup
612 \expandafter\def\expandafter#1\expandafter{\PE@result}%
613 }%
614 \fi

```

\PE@EscapeStringTokens

```

615 \def\PE@EscapeStringTokens#1{%
616 \ifx\relax#1%
617 \else
618 \ifnum'#1<33 %
619 \PE@OctChar#1%
620 \else
621 \ifnum'#1>126 %

```

```

622      \PE@OctChar#1%
623      \else \ifnum'#1=40 \PE@EscapeStringAdd{\string\}% (
624      \else\ifnum'#1=41 \PE@EscapeStringAdd{\string\)}% )
625      \else\ifnum'#1=92 \PE@EscapeStringAdd{\string\\}% \
626      \else
627      \PE@EscapeStringAdd{#1}%
628      \fi\fi\fi
629      \fi
630      \fi
631      \expandafter\PE@EscapeStringTokens
632      \fi
633 }%

```

`\PE@EscapeStringAdd`

```

634 \ifPE@etex
635   \def\PE@EscapeStringAdd#1{#1}%
636 \else
637   \def\PE@EscapeStringAdd#1{%
638     \edef\PE@result{%
639       \PE@result
640       #1%
641     }%
642   }%
643 \fi

644 \PE@AtEnd
645 </package>

```

3 Test

```

646 <*test1 | test2 | test3>
647 \NeedsTeXFormat{LaTeX2e}
648 \makeatletter

```

3.1 Test with `\pdfescape...` commands

```

649 <*test1>
650 \ProvidesFile{pdfescape-test1.tex}%
651   [2007/04/21 v1.4 Test with \string\pdfescape... commands]%
652 </test1>

```

3.2 Test without `\pdfescape...`, with ε -TeX

```

653 <*test2>
654 \ProvidesFile{pdfescape-test2.tex}%
655   [2007/04/21 v1.4 Test without \string\pdfescape..., with e-TeX]%
656 </test2>

```

3.3 Test without `\pdfescape...` and ε -TeX

```

657 <*test3>
658 \ProvidesFile{pdfescape-test3.tex}%
659   [2007/04/21 v1.4 Test without \string\pdfescape... and e-TeX]%
660 </test3>

```

3.4 Check/ensure test preconditions

3.4.1 Check `\pdfescape...`

```

661 <*test1>
662 \@ifundefined{pdfescapehex}{%
663   \PackageError{pdfescape-test1}{%
664     Missing \string\pdfescape... commands%
665   }{Test aborted.}%

```

```

666 \stop
667 }{}
668 </test1>

669 <*test2 | test3>
670 \let\pdfescapehex\@undefined
671 \let\pdfunescapehex\@undefined
672 \let\pdfescapename\@undefined
673 \let\pdfescapestring\@undefined
674 </test2 | test3>

```

3.4.2 Check ε -TeX

```

675 <*test2>
676 \ifundefined{numexpr}{%
677 \PackageError{pdfescape-test2}{%
678 Missing \eTeX
679 }{Test aborted.}%
680 \stop
681 }{}
682 </test2>

```

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

3.5 Common part

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

```

683 \RequirePackage{qstest}
684 \IncludeTests{*}
685 \LogTests{log}{*}{*}
686
687 \newcommand*{\ExpectVar}[2]{%
688 \ifx#1#2%
689 \else
690 \begingroup
691 \@onelevel@sanitize#1%
692 \@onelevel@sanitize#2%
693 \typeout{[#1] <> [#2]}%
694 \endgroup
695 \fi
696 \Expect*{\ifx#1#2true\else false\fi}{true}%
697 }
698
699 \makeatletter
700 \begingroup
701 \gdef\AllBytes{}%
702 \count@=0 %
703 \catcode@=12 %
704 \@whilenum\count@<256 \do{%
705 \lccode@=\count@
706 \ifnum\count@=32 %
707 \xdef\AllBytes{\AllBytes\space}%
708 \else
709 \lowercase{%
710 \xdef\AllBytes{\AllBytes^^@}%
711 }%
712 \fi
713 \advance\count@ by 1 %
714 }%
715 \endgroup
716 \newcommand*{\AllBytesHex}{%
717 000102030405060708090A0B0C0D0E0F%
718 101112131415161718191A1B1C1D1E1F%

```

```

719 202122232425262728292A2B2C2D2E2F%
720 303132333435363738393A3B3C3D3E3F%
721 404142434445464748494A4B4C4D4E4F%
722 505152535455565758595A5B5C5D5E5F%
723 606162636465666768696A6B6C6D6E6F%
724 707172737475767778797A7B7C7D7E7F%
725 808182838485868788898A8B8C8D8E8F%
726 909192939495969798999A9B9C9D9E9F%
727 A0A1A2A3A4A5A6A7A8A9AABACADAFAF%
728 B0B1B2B3B4B5B6B7B8B9BABBBBCDBEBF%
729 C0C1C2C3C4C5C6C7C8C9CACBCCDCECF%
730 D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
731 E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
732 F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF%
733 }
734 \@onelevel@sanitize\AllBytesHex
735 \expandafter\lowercase\expandafter{%
736 \expandafter\newcommand\expandafter*\expandafter\AllBytesHexLC
737 \expandafter{\AllBytesHex}%
738 }
739 \newcommand*{\AllBytesName}{%
740 \begingroup
741 \catcode'\# =12 %
742 \xdef\AllBytesName{%
743 #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
744 #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
745 #20!"#23$#25&'#28#29*+,-.#2F%
746 0123456789:;#3C=#3E?%
747 @ABCDEFGHIJKLMNO%
748 PQRSTUVWXYZ#5B\@backslashchar#5D^_%
749 'abcdefghijklmnopqrstuvwxyz%
750 pqrstuvwxyz7B|#7D\string~#7F%
751 #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
752 #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
753 #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
754 #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
755 #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
756 #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
757 #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
758 #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
759 }%
760 \endgroup
761 \@onelevel@sanitize\AllBytesName
762
763 \newcommand*{\AllBytesString}{%
764 \begingroup
765 \def\|{|}%
766 \edef\%{\@percentchar}%
767 \catcode'\| =0 %
768 \catcode'\# =12 %
769 \catcode'\~ =12 %
770 \catcode'\ =12 %
771 |xdef|AllBytesString{%
772 \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
773 \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
774 \040!"#$%&'(\ )*+,-./%
775 0123456789:;<=>?%
776 @ABCDEFGHIJKLMNO%
777 PQRSTUVWXYZ[\ ]^_%
778 'abcdefghijklmnopqrstuvwxyz%
779 pqrstuvwxyz{|}~^177%
780 \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%

```

```

781 \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
782 \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
783 \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
784 \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
785 \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
786 \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
787 \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
788 }%
789 |endgroup
790 \@onelevel@sanitize\AllBytesString
791
792 <*test3>
793 \let\org@detokenize\detokenize
794 \let\detokenize\@undefined
795 \let\org@numexpr\numexpr
796 \let\numexpr\@undefined
797 </test3>
798 \RequirePackage{pdfescape}
799 <*test3>
800 \let\detokenize\org@detokenize
801 \let\numexpr\org@numexpr
802 </test3>
803
804 \begin{qstest}{all-hex}{\AllBytes, escapehex}
805 \EdefEscapeHex\x{\AllBytes}%
806 \Expect*{\x}*{\AllBytesHex}%
807 \ExpectVar\x\AllBytesHex
808 \end{qstest}
809
810 \begin{qstest}{all-unhex}{\AllBytesHex, unescapehex}
811 \EdefUnescapeHex\x{\AllBytesHex}%
812 \Expect*{\x}*{\AllBytes}%
813 \ExpectVar\x\AllBytes
814 \end{qstest}
815
816 \begin{qstest}{all-unhex-lc}{\AllBytesHexLC, unescapehex, lowercase}
817 \EdefUnescapeHex\x{\AllBytesHexLC}%
818 \Expect*{\x}*{\AllBytes}%
819 \ExpectVar\x\AllBytes
820 \end{qstest}
821
822 \begin{qstest}{unhex-incomplete}{unescapehex, incomplete}
823 \EdefUnescapeHex\x{4}%
824 \Expect*{\x}{@}%
825 \end{qstest}
826
827 \begin{qstest}{unhex-space}{unescapehex, space}
828 \EdefUnescapeHex\x{20}%
829 \Expect*{\x}{ }%
830 \ExpectVar\x\space
831 \end{qstest}
832
833 \begin{qstest}{unhex-spaces}{unescapehex, spaces}
834 \EdefUnescapeHex\x{204020204120}%
835 \def\y#1{%
836 \edef\z{#1\string @#1\string A#1}%
837 }\y{ }%
838 \Expect*{\x}*{\z}%
839 \ExpectVar\x\z
840 \end{qstest}
841
842 \begin{qstest}{unhex-hash}{unescapehex, hash}

```

```

843 \catcode'\#=12 %
844 \EdefUnescapeHex\x{#20}%
845 \ExpectVar\x\space
846 \end{qstest}
847
848 \begin{qstest}{unhex-invalid}{unescapehex, invalid}
849 \def\test#1#2{%
850   \EdefUnescapeHex\x{#1}%
851   \edef\y{#2}%
852   \@onelevel@sanitize\y
853   \ExpectVar\x\y
854 }%
855 <*test1>
856 \edef\x{\pdfunescapehex{4X}}%
857 \edef\y{\string @}%
858 \ifx\x\y
859 \else
860   \def~{\space}%
861   \typeout{*****}%
862   \typeout{* Your pdfTeX contains bug 777.~~~~*}%
863   \typeout{* This test is redefined as dummy, *}%
864   \typeout{* because it triggers the bug.~~~~*}%
865   \typeout{*****}%
866   \def\test#1#2{%
867     \fi
868 </test1>
869 \test{X}{}%
870 \test{XY}{}%
871 \test{XYZ}{}%
872 \test{A}{^~a0}%
873 \test{AX}{^^a0}%
874 \test{XA}{^^a0}%
875 \test{XXAXX}{^^a0}%
876 \end{qstest}
877
878 \begin{qstest}{all-name}{\AllBytes, escapename}
879 \EdefEscapeName\x{\AllBytes}%
880 \Expect*{\x}*{\AllBytesName}%
881 \ExpectVar\x\AllBytesName
882 \end{qstest}
883
884 \begin{qstest}{all-string}{\AllBytes, escapestring}
885 \EdefEscapeString\x{\AllBytes}%
886 \Expect*{\x}*{\AllBytesString}%
887 \ExpectVar\x\AllBytesString
888 \end{qstest}
889
890 \begin{qstest}{uchexdigit}{unescape, uppercase hex digit}
891 \catcode'\@=11 %
892 \catcode0=12 %
893 \def\test#1#2{%
894   \uccode0=#1\relax
895   \uppercase{%
896     \def\x{^^@}%
897   }%
898   \Expect*{%
899     \ifcase\expandafter\PE@TestUcHexDigit\x
900       true%
901     \else
902       false%
903     \fi
904   }{#2}%

```

```

905 }%
906 \def\range#1#2#3{%
907   \count0=#1\relax
908   \loop
909     \ifnum\count0<#2\relax
910       \test{\count0}{#3}%
911       \advance\count0 by 1 %
912     \repeat
913 }%
914 \range{0}{47}{false}%
915 \range{48}{57}{true}%
916 \range{58}{64}{false}%
917 \range{65}{70}{true}%
918 \range{71}{255}{false}%
919 \end{qstest}
920
921 \begin{qstest}{unescapename}{unescapename}
922   \def\test#1#2{%
923     \EdefUnescapeName\x{#1}%
924     \edef\y{#2}%
925     \@onelevel@sanitize\y
926     \ExpectVar\x\y
927   }%
928   \catcode'\#=12 %
929   \catcode0=12 %
930   \test{}{}%
931   \test{x}{x}%
932   \test{xy}{xy}%
933   \test{#}{#}%
934   \test{##}{##}%
935   \test{###}{###}%
936   \test{####}{####}%
937   \test{#x}{#x}%
938   \test{#xy}{#xy}%
939   \test{#1}{#1}%
940   \test{#40}{@}%
941   \test{#400}{@0}%
942   \test{#4x0}{#4x0}%
943   \test{#ab}{^~ab}%
944   \test{#00}{^~@}%
945   \test{x#40y#40z}{x@y@z}%
946   \test{#40#40#40#40}{@@@}%
947   \test{a#x}{a#x}%
948   \test{a#xy}{a#xy}%
949   \test{a#1}{a#1}%
950   \test{a#40}{a@}%
951   \test{a#400}{a@0}%
952   \test{#20}{ }%
953   \test{a#20}{a }%
954   \test{a#20b}{a b}%
955   \test{a#20#20#20b}{a \space\space b}%
956 \end{qstest}
957
958 \begin{qstest}{unescapestring}{unescapestring}
959   \def\test#1#2{%
960     \EdefUnescapeString\x{#1}%
961     \edef\y{#2}%
962     \@onelevel@sanitize\y
963     \ExpectVar\x\y
964   }%
965   \catcode0=12 %
966   \def\DefChar#1#2{%

```

```

967     \begingroup
968     \uccode0=#2\relax
969     \uppercase{\endgroup
970     \def#1{^^@}%
971     }%
972 }%
973 \DefChar\nul{0}%
974 \DefChar\one{1}%
975 \DefChar\bel{8}%
976 \DefChar\tab{9}%
977 \DefChar\lf{10}%
978 \DefChar\ff{12}%
979 \DefChar\cr{13}%
980 \DefChar\{\{92}%
981 \test{}{}%
982 \test{a}{a}%
983 \test{\}{}%
984 \test{\\}{}%
985 \test{\\y}{\y}%
986 \test{\\000}{\nul}%
987 \test{\\b}{\bel}%
988 \test{\\t}{\tab}%
989 \test{\\n}{\lf}%
990 \test{\\f}{\ff}%
991 \test{\\r}{\cr}%
992 \test{\\(){}%
993 \test{\\)}{)}%
994 \test{\\040}{ }%
995 \test{\\100}{@}%
996 \test{\\40}{ }%
997 \test{\\1}{\one}%
998 \test{\\01}{\one}%
999 \test{\\001}{\one}%
1000 \test{\\18}{\one8}%
1001 \test{\\018}{\one8}%
1002 \test{\\0018}{\one8}%
1003 \test{x\\}{x}%
1004 \test{x\\\\}{x\\}%
1005 \test{x\\\\y}{x\y}%
1006 \test{x\\000}{x\nul}%
1007 \test{x\\b}{x\bel}%
1008 \test{x\\t}{x\tab}%
1009 \test{x\\n}{x\lf}%
1010 \test{x\\f}{x\ff}%
1011 \test{x\\r}{x\cr}%
1012 \test{x\\(){x}%
1013 \test{x\\)}{x}%
1014 \test{x\\040}{x }%
1015 \test{x\\100}{x@}%
1016 \test{x\\40}{x }%
1017 \test{x\\1}{x\one}%
1018 \test{x\\01}{x\one}%
1019 \test{x\\001}{x\one}%
1020 \test{x\\18}{x\one8}%
1021 \test{x\\018}{x\one8}%
1022 \test{x\\0018}{x\one8}%
1023 \test{\\b\\t\\n\\f\\r\\(\\)\\\\\\000\\040}{%
1024     \bel\tab\lf\ff\cr()\\nul\space
1025 }%
1026 \test{\\lf}{}%
1027 \test{x\\lf}{x}%
1028 \test{\cr}{\lf}%

```

```

1029 \test{\cr\lf}{\lf}%
1030 \test{\lf}{\lf}%
1031 \test{\lf\cr}{\lf\lf}%
1032 \test{x\cr}{x\lf}%
1033 \test{x\cr\lf}{x\lf}%
1034 \test{x\lf}{x\lf}%
1035 \test{x\lf\cr}{x\lf\lf}%
1036 \test{x\\cr\lf y\cr}{xy\lf}%
1037 \end{qstest}
1038 \stop
1039 </test1 | test2 | test3>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

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

Bundle. All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

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

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-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
pdfescape-test1.tex → doc/latex/oberdiek/pdfescape-test1.tex
pdfescape-test2.tex → doc/latex/oberdiek/pdfescape-test2.tex
pdfescape-test3.tex → doc/latex/oberdiek/pdfescape-test3.tex
pdfescape.dtx      → source/latex/oberdiek/pdfescape.dtx

```

¹<http://ftp.ctan.org/tex-archive/>

If you have a `docstrip.cfg` that configures and enables `docstrip`'s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

4.4 Refresh file name databases

If your \TeX distribution (`teTeX`, `mikTeX`, ...) relies on file name databases, you must refresh these. For example, `teTeX` users run `texhash` or `mktextlsr`.

4.5 Some details for the interested

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

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

Unpacking with \LaTeX . The `.dtx` chooses its action depending on the format:

plain- \TeX : Run `docstrip` and extract the files.

\LaTeX : Generate the documentation.

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

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

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

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

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

An example follows how to generate the documentation with `pdf \LaTeX` :

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

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> . . .	134, 270, 295, 741, 768, 843, 928	<code>_</code>	109, 625
<code>\\$</code>	133		
<code>\%</code>	766		
<code>\(</code>	623, 774		
<code>\)</code>	624, 774		
<code>\@</code>	891		
<code>\@ReturnAfterFi</code>	120, <u>125</u>		
<code>\@backslashchar</code>	748		
<code>\@ifundefined</code>	662, 676		
<code>\@ne</code> . .	161, 164, 209, 212, 216, 326, 329		
<code>\@onelevel@sanitize</code>	93, 691, 692, 734, 761, 790, 852, 925, 962		
<code>\@percentchar</code>	766		
<code>\@undefined</code>	670, 671, 672, 673, 794, 796		
<code>\@whilenum</code>	704		
<code>\</code>	117, 262, 406, 625, 770, 777, 980, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1026, 1027, 1036		
<code>\}</code>	319		
<code>\ </code>	261, 266, 765, 767		
<code>\~</code>	769		

Numbers

<code>\0</code>	772, 773, 774
<code>\1</code>	779
<code>\2</code>	780, 781, 782, 783
<code>\3</code>	784, 785, 786, 787
<code>\8</code>	234
<code>\9</code>	235

	A
<code>\advance</code>	444, 480, 481, 713, 911
<code>\AllBytes</code>	701, 707, 710, 804, 805, 812, 813, 818, 819, 878, 879, 884, 885
<code>\AllBytesHex</code>	716, 734, 737, 806, 807, 810, 811
<code>\AllBytesHexLC</code>	736, 816, 817
<code>\AllBytesName</code>	739, 742, 761, 880, 881
<code>\AllBytesString</code>	763, 790, 886, 887

	B
<code>\begin</code>	804, 810, 816, 822, 827, 833, 842, 848, 878, 884, 890, 921, 958
<code>\bel</code>	975, 987, 1007, 1024

	C
<code>\catcode</code>	3, 5, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 27, 28, 29, 30, 31, 55, 56, 57, 58, 59, 109, 133, 134, 261, 262, 703, 741, 767, 768, 769, 770, 843, 891, 892, 928, 929, 965
<code>\count</code>	907, 909, 910, 911
<code>\count@</code>	702, 704, 705, 706, 713
<code>\cr</code>	979, 991, 1011, 1024, 1028, 1029, 1031, 1032, 1033, 1035, 1036
<code>\csname</code>	2, 32, 42, 60, 73, 77, 80, 87, 98, 359, 382, 411

	D
<code>\DefChar</code>	966, 973, 974, 975, 976, 977, 978, 979, 980
<code>\detokenize</code>	105, 793, 794, 800

<code>\dimen</code>	442, 443, 444, 447, 448, 477, 478, 479, 480, 481, 485, 486, 487
<code>\dimexpr</code>	435, 437, 462, 465, 473
<code>\do</code>	704
E	
<code>\EdefEscapeHex</code>	2, 360, 387, 805
<code>\EdefEscapeName</code>	369, 393, 879
<code>\EdefEscapeString</code>	374, 396, 885
<code>\EdefUnescapeHex</code>	365, 390, 811, 817, 823, 828, 834, 844, 850
<code>\EdefUnescapeName</code>	2, 126, 923
<code>\EdefUnescapeString</code>	2, 226, 960
<code>\empty</code>	36
<code>\end</code>	808, 814, 820, 825, 831, 840, 846, 876, 882, 888, 919, 956, 1037
<code>\endcsname</code>	2, 32, 42, 60, 73, 77, 80, 87, 98, 359, 382, 411
<code>\endinput</code>	51, 400
<code>\escapechar</code>	404
<code>\eTeX</code>	678
<code>\Expect</code>	696, 806, 812, 818, 824, 829, 838, 880, 886, 898
<code>\ExpectVar</code>	687, 807, 813, 819, 830, 839, 845, 853, 881, 887, 926, 963
F	
<code>\ff</code>	978, 990, 1010, 1024
G	
<code>\gdef</code>	135, 143, 701
I	
<code>\ifcase</code>	33, 157, 158, 336, 344, 454, 507, 520, 521, 560, 899
<code>\ifnum</code>	208, 211, 214, 215, 325, 328, 523, 559, 567, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 618, 621, 623, 624, 625, 706, 909
<code>\ifPE@etex</code>	409, 415, 433, 459, 545, 592, 602, 634
<code>\ifx</code>	34, 36, 42, 60, 68, 77, 87, 117, 144, 148, 243, 248, 359, 411, 427, 500, 501, 557, 616, 688, 696, 858
<code>\immediate</code>	44, 62
<code>\IncludeTests</code>	684
L	
<code>\lccode</code>	705
<code>\lf</code>	977, 989, 1009, 1024, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036
<code>\LogTests</code>	685
<code>\loop</code>	908
<code>\lowercase</code>	709, 735
M	
<code>\makeatletter</code>	648, 699
<code>\meaning</code>	89
N	
<code>\NeedsTeXFormat</code>	647
<code>\newcommand</code>	687, 716, 736, 739, 763
<code>\newif</code>	409
<code>\nul</code>	973, 986, 1006, 1024
<code>\number</code>	485, 486, 487
<code>\numexpr</code>	435, 437, 462, 465, 472, 473, 795, 796, 801
O	
<code>\one</code>	974, 997, 998, 999, 1000, 1001, 1002, 1017, 1018, 1019, 1020, 1021, 1022
<code>\org@detokenize</code>	793, 800
<code>\org@numexpr</code>	795, 801
P	
<code>\PackageError</code>	663, 677
<code>\PackageInfo</code>	47
<code>\pdfescape</code>	651, 655, 659, 664
<code>\pdfescapehex</code>	388, 670
<code>\pdfescapename</code>	394, 672
<code>\pdfescapestring</code>	397, 673
<code>\pdfunescapehex</code>	391, 671, 856
<code>\PE@@NormalizeLineEnd</code>	239, 242
<code>\PE@@OctChar</code>	461, 469
<code>\PE@AtEnd</code>	7, 8, 399, 644
<code>\PE@backslash</code>	403, 470, 484
<code>\PE@CheckEndBackslash</code>	316
<code>\PE@DeHex</code>	495, 499
<code>\PE@DeName</code>	139, 143, 172, 176
<code>\PE@DeString</code>	270, 356
<code>\PE@edefbabel</code>	380, 388, 391, 394, 397
<code>\PE@EnsureCode</code>	6, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
<code>\PE@EscapeHex</code>	363, 415
<code>\PE@EscapeName</code>	372, 545
<code>\PE@EscapeNameAdd</code>	563, 568, 582, 590, 592
<code>\PE@EscapeNameHashChar</code>	570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 589
<code>\PE@EscapeNameTokens</code>	547, 552, 556
<code>\PE@EscapeString</code>	377, 602
<code>\PE@EscapeStringAdd</code>	623, 624, 625, 627, 634
<code>\PE@EscapeStringTokens</code>	604, 610, 615
<code>\PE@etexttrue</code>	413
<code>\PE@hash</code>	402, 563, 568, 590
<code>\PE@HexChar</code>	429, 433, 564, 569
<code>\PE@HexDigit</code>	435, 436, 447, 448, 452
<code>\PE@InitUccodeHexDigit</code>	137, 183, 493
<code>\PE@next</code>	146, 151, 172, 176, 180, 245, 249, 251, 254, 337, 339, 341, 345, 347, 349, 502, 508, 512, 531, 534, 540, 543
<code>\PE@NormalizeLineEnd</code>	229, 237
<code>\PE@OctAll</code>	339, 345, 347, 351
<code>\PE@OctChar</code>	459, 619, 622
<code>\PE@OctI</code>	335
<code>\PE@OctII</code>	337, 343
<code>\PE@onelevel@sanitize</code>	82, 88, 93, 100, 104, 130, 231
<code>\PE@result</code>	138, 141, 145, 150, 171, 175, 238, 240, 244,

247, 354, 421, 423, 445, 446,
 482, 483, 494, 497, 530, 551,
 553, 596, 597, 609, 612, 638, 639
 \PE@sanitize
 .. 76, 127, 227, 361, 366, 370, 375
 \PE@SanitizeSpaceOther
 112, 128, 228, 362, 371, 376
 \PE@space@other 108, 119
 \PE@space@space 111, 524
 \PE@SpaceToOther 113, 115
 \PE@strip@prefix 89, 91
 \PE@temp 168, 171, 524, 527, 530
 \PE@testA 154,
 157, 166, 505, 507, 517, 520, 522
 \PE@testB . 155, 158, 166, 518, 521, 522
 \PE@TestOctDigit 324, 336, 344
 \PE@TestUcHexDigit
 . 157, 158, 207, 507, 520, 521, 899
 \PE@ToHex 417, 422, 426
 \PE@UnescapeHex 367, 491
 \PE@UnescapeName 129, 132
 \PE@UnescapeString 230, 263
 \ProvidesFile 650, 654, 658
 \ProvidesPackage 74

R

\range 906, 914, 915, 916, 917, 918
 \repeat 912
 \RequirePackage 683, 798

S

\space ... 707, 830, 845, 860, 955, 1024
 \stop 666, 680, 1038

T

\tab 976, 988, 1008, 1024
 \test 849, 866, 869,
 870, 871, 872, 873, 874, 875,
 893, 910, 922, 930, 931, 932,
 933, 934, 935, 936, 937, 938,
 939, 940, 941, 942, 943, 944,
 945, 946, 947, 948, 949, 950,
 951, 952, 953, 954, 955, 959,
 981, 982, 983, 984, 985, 986,
 987, 988, 989, 990, 991, 992,

993, 994, 995, 996, 997, 998,
 999, 1000, 1001, 1002, 1003,
 1004, 1005, 1006, 1007, 1008,
 1009, 1010, 1011, 1012, 1013,
 1014, 1015, 1016, 1017, 1018,
 1019, 1020, 1021, 1022, 1023,
 1026, 1027, 1028, 1029, 1030,
 1031, 1032, 1033, 1034, 1035, 1036
 \the 3, 9, 462, 465, 472, 473
 \typeout .. 693, 861, 862, 863, 864, 865

U

\uccode 23, 24,
 25, 166, 170, 184, 185, 186, 187,
 188, 189, 190, 191, 192, 193,
 194, 195, 196, 197, 198, 199,
 200, 201, 202, 203, 204, 205,
 234, 235, 352, 522, 523, 894, 968
 \uppercase 153, 167,
 257, 353, 504, 516, 526, 895, 969

W

\write 44, 62

X

\x 32, 34, 36, 43,
 47, 49, 61, 66, 73, 110, 236, 258,
 405, 408, 805, 806, 807, 811,
 812, 813, 817, 818, 819, 823,
 824, 828, 829, 830, 834, 838,
 839, 844, 845, 850, 853, 856,
 858, 879, 880, 881, 885, 886,
 887, 896, 899, 923, 926, 960, 963

Y

\y 835, 837, 851, 852, 853, 857,
 858, 924, 925, 926, 961, 962, 963

Z

\z 836, 838, 839
 \z@ ... 159, 166, 170, 190, 191, 192,
 193, 194, 195, 196, 197, 198,
 199, 200, 201, 202, 203, 204,
 205, 218, 221, 331, 352, 522, 523