

# The pdfrender package

Heiko Oberdiek\*

2023-12-07 v1.8

## Abstract

The PDF format has some graphics parameter like line width or text rendering mode. This package provides an interface for setting these parameters.

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Usage	2
1.2	Macros	2
1.3	Parameters	2
1.3.1	Details	3
1.4	Color stack	4
<b>2</b>	<b>Implementation</b>	<b>4</b>
2.1	Look for pdfTEX, its mode and features	6
2.2	Enable color support of L <sup>A</sup> T <sub>E</sub> X	7
2.3	Hook into \normalcolor	8
2.4	Declare and setup parameters	12
2.5	Fill and stroke color support	13
<b>3</b>	<b>Installation</b>	<b>17</b>
3.1	Download	17
3.2	Package installation	17
3.3	Refresh file name databases	17
3.4	Some details for the interested	18
<b>4</b>	<b>Acknowledgement</b>	<b>18</b>
<b>5</b>	<b>References</b>	<b>18</b>
<b>6</b>	<b>History</b>	<b>18</b>
	[2010/01/26 v1.0]	18
	[2010/01/27 v1.1]	19
	[2010/01/28 v1.2]	19
	[2016/05/14 v1.3]	19
	[2016/05/17 v1.4]	19
	[2018/11/01 v1.5]	19
	[2019/12/29 v1.6]	19
	[2023-12-04 v1.7]	19
	[2023-12-07 v1.8]	19
<b>7</b>	<b>Index</b>	<b>19</b>

---

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

# 1 Documentation

This package `pdfrender` defines an interface for PDF specific parameters that affects the rendering of graphics or text. The interface and its implementation uses the same technique as package `color` for color settings. Therefore this package is loaded to enable L<sup>A</sup>T<sub>E</sub>X's color interface.

At different places L<sup>A</sup>T<sub>E</sub>X uses `\normalcolor` to avoid that header, footer or floats are print in the current color of the main text. `\setgroup@color` is used to start a save box with the color that is set at box saving time. Package `pdfrender` extends these macros to add its own hooks of its parameters. Therefore L<sup>A</sup>T<sub>E</sub>X<sub>3</sub> should generalize L<sup>A</sup>T<sub>E</sub>X<sub>2 $\epsilon$</sub> 's color interface.

## 1.1 Usage

In L<sup>A</sup>T<sub>E</sub>X the package is loaded as normal package. Options are not defined for this package.

```
\usepackage{pdfrender}
```

This package can also be used in plain T<sub>E</sub>X and even iniT<sub>E</sub>X:

```
input pdfrender.sty
```

## 1.2 Macros

```
\pdfrender {<key value list>}
```

The first parameter *<key value list>* contains a list of parameter settings. The key entry is the parameter name. The macro works like `\color` (without optional argument) for color setting.

```
\textpdfrender {<key value list>} {<text>}
```

In the same way as `\pdfrender` the first argument specifies the parameters that should be set. This parameter setting affects *<text>* only. Basically it works the same way as `\textcolor` (without optional argument).

## 1.3 Parameters

The following table shows an overview for the supported parameters and values:

Parameter	Value	Alias
TextRenderingMode	0	Fill
	1	Stroke
	2	FillStroke
	3	Invisible
	4	FillClip
	5	StrokeClip
	6	FillStrokeClip
LineWidth	7	Clip
	<i>positive number, unit is bp</i>	<i>T<sub>E</sub>X dimen</i>

Parameter	Value	Alias
LineCapStyle	0	Butt
	1	Round
	2	ProjectingSquare
LineJoinStyle	0	Miter
	1	Round
	2	Bevel
MiterLimit	<i>positive number</i>	
Flatness	<i>number between 0 and 100</i>	
LineDashPattern	<i>numbers in square brackets, followed by number, units are bp</i>	
RenderingIntent	AbsoluteColorimetric RelativeColorimetric Saturation Perceptual	
FillColor		<i>color specification</i>
StrokeColor		<i>color specification</i>

### 1.3.1 Details

The description and specification of these parameters are available in the PDF specification [1]. Therefore they are not repeated here.

**Value:** The values in the second column lists or describe the values that are specified by the PDF specification.

**Alias:** Instead of magic numbers the package also defines some aliases that can be given as value. Example: `LineCapStyle=Round` has the same effect as `LineCapStyle=1`.

**Number:** The term *number* means an integer or real number. The real number is given as plain decimal number without exponent. The decimal separator is a period. At least one digit must be present.

**LineWidth:** As alias a  $\TeX$  dimen specification can be given. This includes explicit specifications with number and unit, e.g. `LineWidth=0.5pt`. Also  $\LaTeX$  length registers may be used. If  $\varepsilon\text{-}\TeX$ 's `\dimexpr` is available, then it is automatically added. However package `calc` is not supported.

**FillColor, StrokeColor:** Package `color` or `xcolor` must be loaded before these options can be used (since version 1.2).  $\LaTeX$ 's color support sets both colors at the same time to the same value. However parameter `TextRenderingMode` offers the value `FillStroke` that makes only sense, if the two color types can be set separately. If one of the options `FillColor` or `StrokeColor` is specified, then also the color is set. For compatibility with the  $\LaTeX$  color packages (`color` or `xcolor`), always both colors must be set. Thus if one of them is not specified, it is taken from the current color.

Both options `FillColor` and `StrokeColor` expect a  $\LaTeX$  color specification as value. Also the optional color model argument is supported. Example:

```
FillColor=yellow,
StrokeColor=[cmyk]{1,.5,0,0}
```

## 1.4 Color stack

If the pdfTeX version provides color stacks, then each parameter is assigned a page based color stack. The assignment of a stack takes place, when its parameter is set the first time. This avoids the use of color stacks that are not needed.

## 2 Implementation

```
1 (*package)
```

Reload check, especially if the package is not used with L<sup>A</sup>T<sub>E</sub>X.

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

Package identification:

```
33 \begingroup\catcode61\catcode48\catcode32=10\relax%
34 \catcode13=5 % ^M
35 \endlinechar=13 %
36 \catcode35=6 % #
37 \catcode39=12 % '
38 \catcode40=12 % (
39 \catcode41=12 % )
40 \catcode44=12 % ,
41 \catcode45=12 % -
42 \catcode46=12 % .
43 \catcode47=12 % /
44 \catcode58=12 % :
45 \catcode64=11 % @
46 \catcode91=12 % [
47 \catcode93=12 % ]
48 \catcode123=1 % {
49 \catcode125=2 % }
50 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
51 \def\x#1#2#3[#4]{\endgroup
```

```

52     \immediate\write-1{Package: #3 #4}%
53     \xdef#1{#4}%
54 }%
55 \else
56   \def\x#1#2[#3]{\endgroup
57     #2[#3]}%
58   \ifx#1@\undefined
59     \xdef#1{#3}%
60   \fi
61   \ifx#1\relax
62     \xdef#1{#3}%
63   \fi
64 }%
65 \fi
66 \expandafter\x\csname ver@pdfrender.sty\endcsname
67 \ProvidesPackage{pdfrender}%
68 [2023-12-07 v1.8 Access to some PDF graphics parameters (HO)]%
69 \begingroup\catcode61\catcode48\catcode32=10\relax%
70 \catcode13=5 % ^~M
71 \endlinechar=13 %
72 \catcode123=1 % {
73 \catcode125=2 % }
74 \catcode64=11 % @
75 \def\x{\endgroup
76   \expandafter\edef\csname PdfRender@AtEnd\endcsname{%
77     \endlinechar=\the\endlinechar\relax
78     \catcode13=\the\catcode13\relax
79     \catcode32=\the\catcode32\relax
80     \catcode35=\the\catcode35\relax
81     \catcode61=\the\catcode61\relax
82     \catcode64=\the\catcode64\relax
83     \catcode123=\the\catcode123\relax
84     \catcode125=\the\catcode125\relax
85   }%
86 }%
87 \x\catcode61\catcode48\catcode32=10\relax%
88 \catcode13=5 % ^~M
89 \endlinechar=13 %
90 \catcode35=6 % #
91 \catcode64=11 % @
92 \catcode123=1 % {
93 \catcode125=2 % }
94 \def\TMP@EnsureCode#1#2{%
95   \edef\PdfRender@AtEnd{%
96     \PdfRender@AtEnd
97     \catcode#1=\the\catcode#1\relax
98   }%
99   \catcode#1=#2\relax
100 }
101 \TMP@EnsureCode{10}{12}% ^^J
102 \TMP@EnsureCode{36}{3}% $
103 \TMP@EnsureCode{39}{12}% '
104 \TMP@EnsureCode{40}{12}% (
105 \TMP@EnsureCode{41}{12}% )
106 \TMP@EnsureCode{42}{12}% *
107 \TMP@EnsureCode{43}{12}% +
108 \TMP@EnsureCode{44}{12}% ,
109 \TMP@EnsureCode{45}{12}% -
110 \TMP@EnsureCode{46}{12}% .
111 \TMP@EnsureCode{47}{12}% /
112 \TMP@EnsureCode{58}{12}% :
113 \TMP@EnsureCode{59}{12}% ;

```

```

114 \TMP@EnsureCode{60}{12}% <
115 \TMP@EnsureCode{62}{12}% >
116 \TMP@EnsureCode{63}{12}% ?
117 \TMP@EnsureCode{91}{12}% [
118 \TMP@EnsureCode{93}{12}% ]
119 \TMP@EnsureCode{94}{7}% ^ (superscript)
120 \TMP@EnsureCode{96}{12}% ‘
121 \TMP@EnsureCode{124}{12}% |

122 \def\PdfRender@AtEndHook{}
123 \expandafter\def\expandafter\PdfRender@AtEnd\expandafter{%
124   \expandafter\PdfRender@AtEndHook
125   \PdfRender@AtEnd
126   \endinput
127 }

```

## 2.1 Look for pdfTeX, its mode and features

\PdfRender@newif

```

128 \def\PdfRender@newif#1{%
129   \expandafter\edef\csname PdfRender@#1true\endcsname{%
130     \let
131     \expandafter\noexpand\csname ifPdfRender@#1\endcsname
132     \noexpand\iftrue
133   }%
134   \expandafter\edef\csname PdfRender@#1false\endcsname{%
135     \let
136     \expandafter\noexpand\csname ifPdfRender@#1\endcsname
137     \noexpand\iffalse
138   }%
139   \csname PdfRender@#1false\endcsname
140 }

```

\ifPdfRender@Stack

```
141 \PdfRender@newif{Stack}
```

\ifPdfRender@Match

```
142 \PdfRender@newif{Match}
```

\PdfRender@RequirePackage

```

143 \begingroup\expandafter\expandafter\expandafter\endgroup
144 \expandafter\ifx\csname RequirePackage\endcsname\relax
145   \def\PdfRender@RequirePackage#1[#2]{%
146     \expandafter\def\expandafter\PdfRender@AtEndHook\expandafter{%
147       \PdfRender@AtEndHook
148       \ltx@ifpackagelater{#1}{#2}{-}{%
149         \@PackageWarningNoLine{pdfrender}{%
150           You have requested version\MessageBreak
151           ‘#2’ of package ‘#1’,\MessageBreak
152           but only version\MessageBreak
153           ‘\csname ver@#1.\ltx@pkgextension\endcsname’\MessageBreak
154           is available%
155         }%
156       }%
157     }%
158     \input #1.sty\relax
159   }%
160 \else
161   \let\PdfRender@RequirePackage\RequirePackage
162 \fi

```

Luatex compatibility

```
163 \ifx\pdfextension\@undefined\else
```

```

164 \def\pdfcolorstackinit {\pdffeedback colorstackinit}
165 \protected\def\pdfcolorstack    {\pdfextension colorstack}
166 \protected\def\pdfliteral      {\pdfextension literal}
167 \fi

168 \PdfRender@RequirePackage{iftex}[2019/11/07]
169 \PdfRender@RequirePackage{infwarerr}[2007/09/09]
170 \PdfRender@RequirePackage{ltxcmds}[2010/01/28]

171 \ifpdf
172 \ltx@ifundefined{pdfcolorstackinit}{%
173   \@PackageWarning{pdfrender}{%
174     Missing \string\pdfcolorstackinit
175   }%
176 }{%
177   \PdfRender@Stacktrue
178 }%
179 \ltx@ifundefined{pdfmatch}{%
180   \@PackageInfoNoLine{pdfrender}{%
181     \string\pdfmatch\ltx@space not found. %
182     Therefore the values\MessageBreak
183     of some parameters are not validated%
184   }%
185 }{%
186   \PdfRender@Matchtrue
187 }%
188 \else
189   \@PackageWarning{pdfrender}{%
190     Missing pdfTeX in PDF mode%
191   }%
192 \ltx@ifundefined{newcommand}{%

\pdfrender
193   \def\pdfrender#1{%

\textpdfrender
194   \long\def\textpdfrender#1#2{#2}%

195 }{%

\pdfrender
196   \newcommand*\pdfrender}[1]{%

\textpdfrender
197   \newcommand{\textpdfrender}[2]{#2}%

198 }%
199 \expandafter\PdfRender@AtEnd
200 \fi%

```

## 2.2 Enable color support of L<sup>A</sup>T<sub>E</sub>X

```

201 \ltx@ifpackageloaded{color}{%
202   \def\color@setgroup{\begingroup\set@color}%
203   \let\color@begingroup\begingroup
204   \def\color@endgroup{\endgraf\endgroup}%
205   \def\color@hbox{\hbox\bgroup\color@begingroup}%
206   \def\color@vbox{\vbox\bgroup\color@begingroup}%
207   \def\color@endbox{\color@endgroup\egroup}%
208   \ltx@ifundefined{bgroup}{%
209     \let\bgroup=\let\egroup=%
210   }{%
211   \ltx@ifundefined{endgraf}{%

```

```

212   \let\endgraf=\par
213   }{}%
214 }

```

## 2.3 Hook into \normalcolor

With LaTeX we use cmd-hooks.

```
\PdfRender@NormalColorHook
```

```
215 \def\PdfRender@NormalColorHook{}
```

```
\PdfRender@ColorSetGroupHook
```

```
216 \def\PdfRender@ColorSetGroupHook{}
```

```
\PdfRender@TestBox
```

```
217 \def\PdfRender@TestBox#1{%
218   \setbox0=\color@hbox#1\color@endbox
219 }

```

```

220 \def\PdfRender@temp{LaTeX2e}
221 \ifx\PdfRender@temp\fmtname
222   \expandafter \ltx@firstoftwo
223 \else
224   \expandafter\ltx@secondoftwo
225 \fi
226 {
227   \AddToHook{cmd/normalcolor/after}{\PdfRender@NormalColorHook}
228   \AddToHook{cmd/color@setgroup/after}{\PdfRender@NormalColorHook}
229 }

```

with plain we simply append to \normalcolor and to \color@setgroup (but it seems not to work with miniltx anyway, so it is not quite clear if \normalcolor is ever used.).

```

230 {
231   \ltx@ifundefined{normalcolor}
232   {%
233     \gdef\normalcolor{\PdfRender@NormalColorHook}%
234   }
235   {%
236     \ltx@GlobalAppendToMacro\normalcolor{%
237       \PdfRender@NormalColorHook}%
238   }%
239   \ltx@GlobalAppendToMacro\color@setgroup{%
240     \PdfRender@ColorSetGroupHook
241   }%
242 }
243 \PdfRender@RequirePackage{kvsetkeys}[2010/01/28]

```

```
\PdfRender@texorpdfstring
```

```

244 \def\PdfRender@texorpdfstring{%
245   \ltx@ifundefined{texorpdfstring}\ltx@firstoftwo\texorpdfstring
246 }

```

```
\pdfrender
```

```

247 \ltx@ifundefined{DeclareRobustCommand}%
248 \ltx@firstoftwo\ltx@secondoftwo
249 {%
250   \def\pdfrender#1%
251   }{%
252   \newcommand{\pdfrender}{}%
253   \DeclareRobustCommand*\pdfrender[1]%
254   }%

```



```

255 {%
256   \PdfRender@texorpdfstring{%
257     \global\let\PdfRender@FillColor\ltx@empty
258     \global\let\PdfRender@StrokeColor\ltx@empty
259     \kvsetkeys{PDFRENDER}{#1}%
260     \PdfRender@SetColor
261   }{}%
262 }

```

\textpdfrender

```

263 \ltx@ifundefined{DeclareRobustCommand}%
264 \ltx@firstoftwo\ltx@secondoftwo
265 {%
266   \long\def\textpdfrender#1#2%
267 }{%
268   \newcommand{\textpdfrender}{}%
269   \DeclareRobustCommand{\textpdfrender}[2]%
270 }%
271 {%
272   \PdfRender@texorpdfstring{%
273     \begingroup
274       \pdfrender{#1}%
275       #2%
276     \endgroup
277   }{#2}%
278 }

```

\ifPdfRender@Values

```
279 \PdfRender@newif{Values}
```

\PdfRender@NewClassValues

```

280 \def\PdfRender@NewClassValues#1#2#3#4{%
281   \PdfRender@Valuestrue
282   \PdfRender@NewClass{#1}{#2}{#3}{#4}{}%
283 }

```

\PdfRender@NewClass

```

284 \def\PdfRender@NewClass#1#2#3#4#5{%
285   \PdfRender@newif{Active#1}%
286   \expandafter\def\csname PdfRender@Default#1\endcsname{#2}%
287   \expandafter\let\csname PdfRender@Current#1\expandafter\endcsname
288     \csname PdfRender@Default#1\endcsname
289   \ifPdfRender@Stack
290     \expandafter\edef\csname PdfRender@Init#1\endcsname{%
291       \global\chardef
292       \expandafter\noexpand\csname PdfRender@Stack#1\endcsname=%
293       \noexpand\pdfcolorstackinit page direct{%
294         \noexpand#3%
295       \expandafter\noexpand\csname PdfRender@Default#1\endcsname
296     }\relax
297     \noexpand\@PackageInfo{pdfrender}{}%
298     New color stack '#1' = \noexpand\number
299     \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
300   }%
301   \gdef\expandafter\noexpand\csname PdfRender@Init#1\endcsname{}%
302 }%
303 \expandafter\edef\csname PdfRender@Set#1\endcsname{%
304   \expandafter\noexpand\csname PdfRender@Init#1\endcsname
305   \noexpand\pdfcolorstack
306   \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
307   push{%
308     #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%

```

```

309     }%
310     \noexpand\aftergroup
311     \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
312 }%
313 \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
314     \expandafter\noexpand\csname PdfRender@Init#1\endcsname
315     \noexpand\pdfcolorstack
316     \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
317     pop\relax
318 }%
319 \else
320     \expandafter\edef\csname PdfRender@Set#1\endcsname{%
321         \noexpand\pdfliteral direct{%
322             #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
323         }%
324         \noexpand\aftergroup
325         \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
326     }%
327     \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
328         \noexpand\pdfliteral direct{%
329             #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
330         }%
331     }%
332 \fi
333 \expandafter\edef\csname PdfRender@Normal#1\endcsname{%
334     \let
335     \expandafter\noexpand\csname PdfRender@Current#1\endcsname
336     \expandafter\noexpand\csname PdfRender@Default#1\endcsname
337     \noexpand\PdfRender@Set{#1}%
338 }%
339 \expandafter\ltx@GlobalAppendToMacro\expandafter\PdfRender@NormalColorHook
340 \expandafter{%
341     \csname PdfRender@Normal#1\endcsname
342 }%
343 \ltx@GlobalAppendToMacro\PdfRender@ColorSetGroupHook{%
344     \PdfRender@Set{#1}%
345 }%
346 \ifPdfRender@Values
347     \kv@parse@normalized{#4}{%
348         \expandafter\let\csname PdfRender@#1@\kv@key\endcsname\kv@key
349         \ifx\kv@value\relax
350         \else
351             \expandafter\let\csname PdfRender@#1@\kv@value\endcsname\kv@key
352         \fi
353     \ltx@gobbletwo
354 }%
355 \PdfRender@define@key{PDFRENDER}{#1}{%
356     \global\csname PdfRender@Active#1true\endcsname
357     \def\PdfRender@Current{##1}%
358     \PdfRender@SetValidateValues{#1}%
359 }%
360 \PdfRender@Valuesfalse
361 \else
362     \PdfRender@define@key{PDFRENDER}{#1}{%
363         \global\csname PdfRender@Active#1true\endcsname
364         \expandafter\def\csname PdfRender@Current#1\endcsname{##1}%
365         \ltx@ifundefined{PdfRender@PostProcess#1}{%
366             }{%
367                 \csname PdfRender@PostProcess#1\endcsname
368             }%
369         \PdfRender@SetValidate{#1}{#4}{#5}%
370     }%

```

```
371 \fi
372 }%
```

```
\PdfRender@define@key
```

```
373 \ltx@ifundefined{define@key}{%
374 \def\PdfRender@define@key#1#2{%
375 \expandafter\def\csname KV@#1@#2\endcsname##1%
376 }%
377 }{%
378 \let\PdfRender@define@key\define@key
379 }
```

```
\PdfRender@Set
```

```
380 \def\PdfRender@Set#1{%
381 \csname ifPdfRender@Active#1\endcsname
382 \csname PdfRender@Set#1\expandafter\endcsname
383 \fi
384 }
```

```
\PdfRender@Reset
```

```
385 \def\PdfRender@Reset#1{%
386 \csname ifPdfRender@Active#1\endcsname
387 \csname PdfRender@Reset#1\expandafter\endcsname
388 \fi
389 }
```

```
\PdfRender@ErrorInvalidValue
```

```
390 \def\PdfRender@ErrorInvalidValue#1{%
391 \PackageError{pdfrender}{%
392 Ignoring parameter setting for ‘#1’\MessageBreak
393 because of invalid value %
394 ‘\csname PdfRender@Current#1\endcsname’%
395 }@ehc
396 \expandafter\let\csname PdfRender@Current#1\endcsname\ltx@empty
397 }%
```

```
\PdfRender@SetValidate
```

```
398 \ifPdfRender@Match
399 \def\PdfRender@SetValidate#1#2#3{%
400 \ifnum\pdfmatch{^(#2)$}{\csname PdfRender@Current#1\endcsname}=1 %
401 \csname PdfRender@Set#1\expandafter\endcsname
402 \else
403 \PdfRender@ErrorInvalidValue{#1}%
404 \fi
405 }%
406 \else
407 \def\PdfRender@SetValidate#1#2#3{%
408 \expandafter\let\expandafter\PdfRender@Current
409 \csname PdfRender@Current#1\endcsname
410 #3%
411 \ifx\PdfRender@Current\@empty
412 \PdfRender@ErrorInvalidValue{#1}%
413 \else
414 \csname PdfRender@Set#1\expandafter\endcsname
415 \fi
416 }%
417 \fi
```

```
\PdfRender@SetValidateValues
```

```
418 \def\PdfRender@SetValidateValues#1{%
419 \ltx@ifundefined{PdfRender@#1@\PdfRender@Current}{%
420 \expandafter\let\csname PdfRender@Current#1\endcsname
```

```

421             \PdfRender@Current
422     \PdfRender@ErrorInvalidValue{#1}%
423   }{%
424     \expandafter\edef\csname PdfRender@Current#1\endcsname{%
425       \csname PdfRender@#1@\PdfRender@Current\endcsname
426     }%
427     \csname PdfRender@Set#1\endcsname
428   }%
429 }

```

\PdfRender@OpValue

```
430 \def\PdfRender@OpValue#1#2{#2\ltx@space#1}%
```

\PdfRender@OpName

```
431 \def\PdfRender@OpName#1#2{/#2\ltx@space#1}%
```

## 2.4 Declare and setup parameters

```

432 \PdfRender@NewClassValues{TextRenderingMode}%
433     {0}%
434     {\PdfRender@OpValue{Tr}}{%
435   0=Fill,%
436   1=Stroke,%
437   2=FillStroke,%
438   3=Invisible,%
439   4=FillClip,%
440   5=StrokeClip,%
441   6=FillStrokeClip,%
442   7=Clip,%
443 }%
444 \PdfRender@NewClass{LineWidth}{1}{\PdfRender@OpValue{w}}{%
445   [0-9]+\string\.[0-9]*|\string\.[0-9]+%
446 }{%
447   \ltx@ifundefined{dimexpr}{%
448     \def\PdfRender@dimexpr{%
449   }{%
450     \let\PdfRender@dimexpr\dimexpr
451   }
452 \def\PdfRender@PostProcessLineWidth{%
453   \begingroup
454   \afterassignment\PdfRender@@PostProcessLineWidth
455   \dimen0=\PdfRender@dimexpr\PdfRender@CurrentLineWidth bp %
456   \PdfRender@let\PdfRender@relax\PdfRender@relax
457 }
458 \let\PdfRender@let\let
459 \let\PdfRender@relax\relax
460 \def\PdfRender@@PostProcessLineWidth#1\PdfRender@let{%
461   \ifx\#1\%
462     \endgroup
463   \else
464     \dimen0=.996264\dimen0 % 72/72.27
465     \edef\x{\endgroup
466       \def\noexpand\PdfRender@CurrentLineWidth{%
467         \strip@pt\dimen0%
468       }%
469     }%
470     \expandafter\x
471     \fi
472 }
473 \PdfRender@NewClassValues{LineCapStyle}{0}{\PdfRender@OpValue{J}}{%
474   0=Butt,%
475   1=Round,%

```

```

476 2=ProjectingSquare,%
477 }%
478 \PdfRender@NewClassValues{LineJoinStyle}{0}{\PdfRender@OpValue{j}}{%
479 0=Miter,%
480 1=Round,%
481 2=Bevel,%
482 }%
483 \PdfRender@NewClass{MiterLimit}{10}{\PdfRender@OpValue{M}}{%
484 [0-9]*[1-9][0-9]*\string\.[0-9]*|
485 [0-9]*\string\.[0-9]*[1-9][0-9]*%
486 }{%
487 \PdfRender@NewClass{Flatness}{0}{\PdfRender@OpValue{i}}{%
488 100(\string\.[0-9]*| [0-9][0-9](\string\.[0-9]*)|\string\.[0-9]+%
489 }{%
490 \PdfRender@NewClass{LineDashPattern}{[]}{\PdfRender@OpValue{d}}{%
491 \string\%
492 ( ?([0-9]+\string\.[0-9]*|\string\.[0-9]+) ?)*%
493 \string\ ?%
494 ([0-9]+\string\.[0-9]*|\string\.[0-9]+)%
495 }{%
496 \PdfRender@NewClassValues{RenderingIntent}%
497 {RelativeColorimetric}%
498 {\PdfRender@OpName{ri}}{%
499 AbsoluteColorimetric,%
500 RelativeColorimetric,%
501 Saturation,%
502 Perceptual,%
503 }%

```

## 2.5 Fill and stroke color support

```

504 \PdfRender@define@key{PDFRENDER}{FillColor}{%
505 \begingroup
506 \def\PdfRender@Color{#1}%
507 \ifx\PdfRender@Color\ltx@empty
508 \global\let\PdfRender@FillColor\ltx@empty
509 \else
510 \PdfRender@ColorAvailable{%
511 \PdfRender@TestBox{%
512 \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
513 \PdfRender@GetFillColor
514 \ifx\PdfRender@FillColor\ltx@empty
515 \PackageWarning{pdfrender}{%
516 Cannot extract fill color\MessageBreak
517 from value '#1'%
518 }%
519 \fi
520 }%
521 }%
522 \fi
523 \endgroup
524 }
525 \PdfRender@define@key{PDFRENDER}{StrokeColor}{%
526 \begingroup
527 \def\PdfRender@Color{#1}%
528 \ifx\PdfRender@Color\ltx@empty
529 \global\let\PdfRender@StrokeColor\ltx@empty
530 \else
531 \PdfRender@ColorAvailable{%
532 \PdfRender@TestBox{%
533 \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
534 \PdfRender@GetStrokeColor
535 \ifx\PdfRender@StrokeColor\ltx@empty

```

```

536         \@PackageWarning{pdfrender}{%
537             Cannot extract stroke color\MessageBreak
538             from value '#1'%
539         }%
540     \fi
541 }%
542 }%
543 \fi
544 \endgroup
545 }

\PdfRender@ColorAvailable

546 \def\PdfRender@ColorAvailable{%
547     \@ifundefined{set@color}{%
548         \@PackageError{pdfrender}{%
549             Ignoring color options, because neither\MessageBreak
550             package 'color' nor package 'xcolor' is loaded%
551         }\@ehc
552         \global\let\PdfRender@ColorAvailable\ltx@gobble
553     }{%
554         \global\let\PdfRender@ColorAvailable\ltx@firstofone
555     }%
556     \PdfRender@ColorAvailable
557 }

\PdfRender@TryColor

558 \def\PdfRender@TryColor{%
559     \@ifnextchar[\color\PdfRender@@TryColor
560 }

\PdfRender@@TryColor

561 \def\PdfRender@@TryColor#1\ltx@empty{%
562     \expandafter\color\expandafter{\PdfRender@Color}%
563 }

\PdfRender@SetColor

564 \def\PdfRender@SetColor{%
565     \chardef\PdfRender@NeedsCurrentColor=0 %
566     \ifx\PdfRender@FillColor\ltx@empty
567         \ifx\PdfRender@StrokeColor\ltx@empty
568             \else
569                 \edef\PdfRender@CurrentColor{%
570                     \noexpand\PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
571                 }%
572                 \chardef\PdfRender@NeedsCurrentColor=1 %
573             \fi
574         \else
575             \ifx\PdfRender@StrokeColor\ltx@empty
576                 \edef\PdfRender@CurrentColor{%
577                     \PdfRender@FillColor\ltx@space\noexpand\PdfRender@StrokeColor
578                 }%
579                 \chardef\PdfRender@NeedsCurrentColor=2 %
580             \else
581                 \edef\current@color{%
582                     \PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
583                 }%
584                 \set@color
585             \fi
586         \fi
587     \ifnum\PdfRender@NeedsCurrentColor=1 %
588         \PdfRender@GetFillColor
589         \ifx\PdfRender@FillColor\ltx@empty
590             \@PackageWarning{pdfrender}{%

```

```

591     Cannot extract current fill color%
592   }%
593   \else
594     \edef\current@color{\PdfRender@CurrentColor}%
595     \set@color
596   \fi
597 \else
598   \ifnum\PdfRender@NeedsCurrentColor=2 %
599     \PdfRender@GetStrokeColor
600     \ifx\PdfRender@StrokeColor\ltx@empty
601       \@PackageWarning{pdfrender}{%
602         Cannot extract current stroke color%
603       }%
604     \else
605       \edef\current@color{\PdfRender@CurrentColor}%
606       \set@color
607     \fi
608   \fi
609 \fi
610 }

```

\PdfRender@PatternFillColor

```

611 \edef\PdfRender@PatternFillColor{ % space
612   (%
613     [0-9\string\.] + g|%
614     [0-9\string\.] + [0-9\string\.] + [0-9\string\.] + rg|%
615     [0-9\string\.] + [0-9\string\.] + %
616     [0-9\string\.] + [0-9\string\.] + k%
617   ) % space
618   (.*)$%
619 }

```

\PdfRender@PatternStrokeColor

```

620 \edef\PdfRender@PatternStrokeColor{ % space
621   (%
622     [0-9\string\.] + G|%
623     [0-9\string\.] + [0-9\string\.] + [0-9\string\.] + RG|%
624     [0-9\string\.] + [0-9\string\.] + %
625     [0-9\string\.] + [0-9\string\.] + K%
626   ) % space
627   (.*)$%
628 }

```

\PdfRender@MatchPattern

```

629 \def\PdfRender@MatchPattern#1{%
630   \ifnum\pdfmatch{\PdfRender@Pattern}{\PdfRender@String}=1 %
631     \xdef#1{%
632       \expandafter\strip@prefix\pdflastmatch 1%
633     }%
634   \edef\PdfRender@String{%
635     \expandafter\strip@prefix\pdflastmatch 2%
636   }%
637   \ifx\PdfRender@String\ltx@empty
638     \else
639     \expandafter\expandafter\expandafter\PdfRender@MatchPattern
640     \expandafter\expandafter\expandafter#1%
641   \fi
642 \fi
643 }

```

\PdfRender@GetFillColor

```

644 \def\PdfRender@GetFillColor{%

```

```

645 \global\let\PdfRender@FillColor\ltx@empty
646 \begingroup
647   \ifPdfRender@Match
648     \let\PdfRender@Pattern\PdfRender@PatternFillColor
649     \edef\PdfRender@String{\ltx@space\current@color\ltx@space}%
650     \PdfRender@MatchPattern\PdfRender@FillColor
651   \else
652     \edef\current@color{\current@color\ltx@space}%
653     \let\PdfRender@OP\relax
654     \PdfRender@FindOp{g}0%
655     \PdfRender@FindOp{G}1%
656     \PdfRender@FindOp{rg}0%
657     \PdfRender@FindOp{RG}1%
658     \PdfRender@FindOp{k}0%
659     \PdfRender@FindOp{K}1%
660     \PdfRender@FilterOp 0\PdfRender@FillColor
661   \fi
662 \endgroup
663 }

```

\PdfRender@GetStrokeColor

```

664 \def\PdfRender@GetStrokeColor{%
665   \global\let\PdfRender@StrokeColor\ltx@empty
666   \begingroup
667     \ifPdfRender@Match
668       \let\PdfRender@Pattern\PdfRender@PatternStrokeColor
669       \edef\PdfRender@String{\ltx@space\current@color\ltx@space}%
670       \PdfRender@MatchPattern\PdfRender@StrokeColor
671     \else
672       \edef\current@color{\current@color\ltx@space}%
673       \let\PdfRender@OP\relax
674       \PdfRender@FindOp{g}0%
675       \PdfRender@FindOp{G}1%
676       \PdfRender@FindOp{rg}0%
677       \PdfRender@FindOp{RG}1%
678       \PdfRender@FindOp{k}0%
679       \PdfRender@FindOp{K}1%
680       \PdfRender@FilterOp 1\PdfRender@StrokeColor
681     \fi
682   \endgroup
683 }

684 \ifPdfRender@Match
685   \expandafter\PdfRender@AtEnd
686 \fi%

```

\PdfRender@FindOp

```

687 \def\PdfRender@FindOp#1#2{%
688   \def\PdfRender@temp##1 #1 ##2\@nil{%
689     ##1%
690     \ifx\##2\%
691       \expandafter\@gobble
692     \else
693       \PdfRender@OP{#1}#2%
694       \expandafter\@firstofone
695     \fi
696     {%
697       \PdfRender@temp##2\@nil
698     }%
699   }%
700   \edef\current@color{%
701     \@firstofone{\expandafter\PdfRender@temp\current@color} #1 \@nil
702   }%

```



```

703 }

\PdfRender@FilterOp
704 \def\PdfRender@FilterOp#1#2{%
705   \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
706     \current@color\PdfRender@OP{}{}%
707 }

\PdfRender@@FilterOp
708 \def\PdfRender@@FilterOp#1#2#3\PdfRender@OP#4#5{%
709   \ifx\\#4#5\\%
710     \else
711       \ifnum#1=#5 %
712         \xdef#2{#3 #4}%
713       \fi
714       \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
715     \fi
716 }

717 \PdfRender@AtEnd%
718 </package>

```

## 3 Installation

### 3.1 Download

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

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

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

### 3.2 Package installation

**Unpacking.** The `.dtx` file is a self-extracting docstrip archive. The files are extracted by running the `.dtx` through plain  $\TeX$ :

```
tex pdfrender.dtx
```

**TDS.** Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

```

pdfrender.sty → tex/generic/pdfrender/pdfrender.sty
pdfrender.pdf → doc/latex/pdfrender/pdfrender.pdf
pdfrender.dtx → source/latex/pdfrender/pdfrender.dtx

```

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

### 3.3 Refresh file name databases

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

---

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

### 3.4 Some details for the interested

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

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

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

If you insist on using L<sup>A</sup>T<sub>E</sub>X for `docstrip` (really, `docstrip` does not need L<sup>A</sup>T<sub>E</sub>X), then inform the `autodetect` routine about your intention:

```
latex \let\install=y\input{pdfrender.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<sup>A</sup>T<sub>E</sub>X:

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

## 4 Acknowledgement

**Friedrich Vosberg** asked in the newsgroup `de.comp.text.tex` for the font outline feature [2].

**Gaius Pupus** proposed the basic method using `\pdfliteral` in this thread [3].

**Rolf Niepraschk** added color support [4].

## 5 References

- [1] Adobe Systems Incorporated. *PDF Reference – Adobe Portable Document format – Version 1.7*. 6th ed. 2006. URL: [http://www.adobe.com/devnet/acrobat/pdfs/pdf\\_reference\\_1-7.pdf](http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf).
- [2] Friedrich Vosberg, *Text in Buchstabenumrissen*, `de.comp.text.tex`, 2010-01-22. URL: <https://groups.google.com/group/de.comp.text.tex/msg/f442310ac8b2d506>.
- [3] Gaius Pupus, *Re: Text in Buchstabenumrissen*, `de.comp.text.tex`, 2010-01-23. URL: <https://groups.google.com/group/de.comp.text.tex/msg/95d890d77ac47eb1>.
- [4] Rolf Niepraschk, *Re: Text in Buchstabenumrissen*, `de.comp.text.tex`, 2010-01-24. URL: <https://groups.google.com/group/de.comp.text.tex/msg/4eb61a5879db54db>.

## 6 History

[2010/01/26 v1.0]

- The first version.

### [2010/01/27 v1.1]

- Macros `\pdfrender` and `\textpdfrender` are made robust.
- Color extraction rewritten for the case that `\pdfmatch` is not available. This fixes wrong color assignments in case of nesting.
- Color extraction of case `\pdfmatch` is fixed for the case that the color string contains several fill or several stroke operations.

### [2010/01/28 v1.2]

- Dependency from package `color` is removed.
- Compatibility for plain `TEX` and even `iniTEX` added.

### [2016/05/14 v1.3]

- Use package `luatex85` for compatibility with new `LuaTEX`.

### [2016/05/17 v1.4]

- Documentation updates.
- adjust `luatex85` reference so that it works in plain `TeX`.

### [2018/11/01 v1.5]

- Remove `luatex85` dependency

### [2019/12/29 v1.6]

- `iftex` package.

### [2023-12-04 v1.7]

- Use (with `LaTEX`) `cmd-hooks` to patch `\normalcolor` and `\color@setgroup`. This avoids a clash with `xcolor`.

### [2023-12-07 v1.8]

- Fix wrong command name, issue #3.

## 7 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>	<i>445, 484, 485, 488, 492, 494, 613, 614, 615, 616, 622, 623, 624, 625</i>
<code>\@PackageError</code>	<i>548</i>
<code>\@PackageInfo</code>	<i>297</i>
<code>\@PackageInfoNoLine</code>	<i>180</i>
<code>\@PackageWarning</code>	<i>173, 189, 515, 536, 590, 601</i>
<code>\@PackageWarningNoLine</code>	<i>149</i>
<code>\@ehc</code>	<i>395, 551</i>
<code>\@empty</code>	<i>411</i>
<code>\@firstofone</code>	<i>694, 701</i>
<code>\@gobble</code>	<i>691</i>
<code>\@ifnextchar</code>	<i>559</i>
<code>\@ifundefined</code>	<i>547</i>
<code>\@nil</code>	<i>688, 697, 701</i>
<code>\@undefined</code>	<i>58, 163</i>
<code>\[</code>	<i>491</i>
<code>\]</code>	<i>461, 690, 709</i>
<code>\]</code>	<i>493</i>
<b>A</b>	
<code>\AddToHook</code>	<i>227, 228</i>

<code>\afterassignment</code>	454	<code>\ifnum</code>	400, 587, 598, 630, 711
<code>\aftergroup</code>	29, 310, 324	<code>\ifpdf</code>	171
<b>C</b>			
<code>\catcode</code>	2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 69, 70, 72, 73, 74, 78, 79, 80, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 97, 99	<code>\ifPdfRender@Match</code>	142, 398, 647, 667, 684
<code>\chardef</code>	291, 565, 572, 579	<code>\ifPdfRender@Stack</code>	141, 289
<code>\color</code>	559, 562	<code>\ifPdfRender@Values</code>	279, 346
<code>\color@begingroup</code>	203, 205, 206	<code>\iftrue</code>	132
<code>\color@endbox</code>	207, 218	<code>\ifx</code>	15, 18, 21, 50, 58, 61, 144, 163, 221, 349, 411, 461, 507, 514, 528, 535, 566, 567, 575, 589, 600, 637, 690, 709
<code>\color@endgroup</code>	204, 207	<code>\immediate</code>	23, 52
<code>\color@hbox</code>	205, 218	<code>\input</code>	158
<code>\color@setgroup</code>	202, 239	<b>K</b>	
<code>\color@vbox</code>	206	<code>\kv@key</code>	348, 351
<code>\csname</code>	14, 21, 50, 66, 76, 129, 131, 134, 136, 139, 144, 153, 286, 287, 288, 290, 292, 295, 299, 301, 303, 304, 306, 308, 311, 313, 314, 316, 320, 322, 325, 327, 329, 333, 335, 336, 341, 348, 351, 356, 363, 364, 367, 375, 381, 382, 386, 387, 394, 396, 400, 401, 409, 414, 420, 424, 425, 427	<code>\kv@parse@normalized</code>	347
<code>\current@color</code>	581, 594, 605, 649, 652, 669, 672, 700, 701, 706	<code>\kv@value</code>	349, 351
<b>D</b>			
<code>\DeclareRobustCommand</code>	253, 269	<code>\kvsetkeys</code>	259
<code>\define@key</code>	378	<b>L</b>	
<code>\dimen</code>	455, 464, 467	<code>\ltx@empty</code>	257, 258, 396, 507, 508, 512, 514, 528, 529, 533, 535, 561, 566, 567, 575, 589, 600, 637, 645, 665
<code>\dimexpr</code>	450	<code>\ltx@firstofone</code>	554
<b>E</b>			
<code>\empty</code>	17, 18	<code>\ltx@firstoftwo</code>	222, 245, 248, 264
<code>\endcsname</code>	14, 21, 50, 66, 76, 129, 131, 134, 136, 139, 144, 153, 286, 287, 288, 290, 292, 295, 299, 301, 303, 304, 306, 308, 311, 313, 314, 316, 320, 322, 325, 327, 329, 333, 335, 336, 341, 348, 351, 356, 363, 364, 367, 375, 381, 382, 386, 387, 394, 396, 400, 401, 409, 414, 420, 424, 425, 427	<code>\ltx@GlobalAppendToMacro</code>	236, 239, 339, 343
<code>\endgraf</code>	204, 212	<code>\ltx@gobble</code>	552
<code>\endinput</code>	29, 126	<code>\ltx@gobbletwo</code>	353
<code>\endlinechar</code>	4, 35, 71, 77, 89	<code>\ltx@ifpackagelater</code>	148
<b>F</b>			
<code>\fmtname</code>	221	<code>\ltx@ifpackageloaded</code>	201
<b>G</b>			
<code>\gdef</code>	233, 301	<code>\ltx@ifundefined</code>	172, 179, 192, 245, 247, 263, 365, 373, 419, 447
<b>H</b>			
<code>\hbox</code>	205	<code>\ltx@ifundefined</code>	208, 211, 231
<b>I</b>			
<code>\iffalse</code>	137	<code>\ltx@pkgextension</code>	153
		<code>\ltx@secondoftwo</code>	224, 248, 264
		<code>\ltx@space</code>	181, 430, 431, 570, 577, 582, 649, 652, 669, 672
		<b>M</b>	
		<code>\MessageBreak</code>	150, 151, 152, 153, 182, 392, 516, 537, 549
		<b>N</b>	
		<code>\newcommand</code>	196, 197, 252, 268
		<code>\normalcolor</code>	233, 236
		<code>\number</code>	298
		<b>P</b>	
		<code>\PackageError</code>	391
		<code>\PackageInfo</code>	26
		<code>\par</code>	212
		<code>\pdfcolorstack</code>	165, 305, 315
		<code>\pdfcolorstackinit</code>	164, 174, 293
		<code>\pdfextension</code>	163, 165, 166
		<code>\pdffeedback</code>	164
		<code>\pdflastmatch</code>	632, 635
		<code>\pdfliteral</code>	166, 321, 328
		<code>\pdfmatch</code>	181, 400, 630
		<code>\pdfrender</code>	2, 193, 196, 247, 274
		<code>\PdfRender@@FilterOp</code>	705, 708

