

# File I

## Implementation

### 1 l3backend-basics Implementation

```
1 <*package>
```

Whilst there is a reasonable amount of code overlap between backends, it is much clearer to have the blocks more-or-less separated than run in together and DocStripped out in parts. As such, most of the following is set up on a per-backend basis, though there is some common code (again given in blocks not interspersed with other material).

All the file identifiers are up-front so that they come out in the right place in the files.

```
2 \ProvidesExplFile
3 <*dvipdfmx>
4   {l3backend-dvipdfmx.def}{2021-03-18}{}
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2021-03-18}{}
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2021-03-18}{}
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2021-03-18}{}
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2021-03-18}{}
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2021-03-18}{}
25   {L3 backend support: XeTeX}
26 </xetex>
```

Check if the loaded kernel is at least enough to load this file. The kernel date has to be at least equal to \ExplBackendFileDate or later. If \\_kernel\_dependency\\_version\\_check:Nn doesn't exist we're loading in an older kernel, so it's an error anyway. With time, this test should vanish and only the dependency check should remain.

```
27 \cs_if_exist:NTF \_kernel_dependency_version_check:nn
28 {
29   \_kernel_dependency_version_check:nn {2020-09-01}
30 <dvipdfmx>   {l3backend-dvipdfmx.def}
31 <dvips>      {l3backend-dvips.def}
32 <dvisvgm>    {l3backend-dvisvgm.def}
33 <luatex>     {l3backend-luatex.def}
34 <pdftex>     {l3backend-pdftex.def}
35 <xetex>      {l3backend-xetex.def}
```

```

36 }
37 {
38   \cs_if_exist_use:cF { @latex@error } { \errmessage }
39   {
40     Mismatched-LaTeX-support-files-detected. \MessageBreak
41     Loading~aborted!
42   }
43   { \use:c { @ehd } }
44   \tex_endinput:D
45 }

```

The order of the backend code here is such that we get somewhat logical outcomes in terms of code sharing whilst keeping things readable. (Trying to mix all of the code by concept is almost unmanageable.) The key parts which are shared are

- Color support is either dvips-like or LuaTeX/pdfTeX-like.
- LuaTeX/pdfTeX and dvipdfmx/X<sub>Y</sub>TeX share drawing routines.
- X<sub>Y</sub>TeX is the same as dvipdfmx other than image size extraction so takes most of the same code.

`\__kernel_backend_literal:e` The one shared function for all backends is access to the basic `\special` primitive: it has slightly odd expansion behaviour so a wrapper is provided.

```

\__kernel_backend_literal:n
\__kernel_backend_literal:x
46 \cs_new_eq:NN \__kernel_backend_literal:e \tex_special:D
47 \cs_new_protected:Npn \__kernel_backend_literal:n #1
48 { \__kernel_backend_literal:e { \exp_not:n {#1} } }
49 \cs_generate_variant:Nn \__kernel_backend_literal:n { x }

```

(End definition for `\__kernel_backend_literal:e`.)

## 1.1 dvips backend

```

50 <dvips>

```

`\__kernel_backend_literal_postscript:n` Literal PostScript can be included using a few low-level formats. Here, we use the form with no positioning; this is overall more convenient as a wrapper. Note that this does require that where position is important, an appropriate wrapper is included.

```

51 \cs_new_protected:Npn \__kernel_backend_literal_postscript:n #1
52 { \__kernel_backend_literal:n { ps:: #1 } }
53 \cs_generate_variant:Nn \__kernel_backend_literal_postscript:n { x }

```

(End definition for `\__kernel_backend_literal_postscript:n`.)

`\__kernel_backend_postscript:n` PostScript data that does have positioning, and also applying a shift to `SDict` (which is not done automatically by `ps:` or `ps::`, in contrast to `!` or `"`).

```

54 \cs_new_protected:Npn \__kernel_backend_postscript:n #1
55 { \__kernel_backend_literal:n { ps: SDict ~ begin ~ #1 ~ end } }
56 \cs_generate_variant:Nn \__kernel_backend_postscript:n { x }

```

(End definition for `\__kernel_backend_postscript:n`.)

PostScript for the header: a small saving but makes the code clearer. This is held until the start of shipout such that a document with no actual output does not write anything.

```

57 \bool_if:NT \g__kernel_backend_header_bool
58 {

```

```

59 \cs_if_exist:NTF \AtBeginDvi
60 { \AtBeginDvi }
61 { \use:n }
62 { \__kernel_backend_literal:n { header = l3backend-dvips.pro } }
63 }

```

`\__kernel_backend_align_begin:` In **dvips** there is no built-in saving of the current position, and so some additional PostScript is required to set up the transformation matrix and also to restore it afterwards. Notice the use of the stack to save the current position “up front” and to move back to it at the end of the process. Notice that the `[begin]/[end]` pair here mean that we can use a run of PostScript statements in separate lines: not *required* but does make the code and output more clear.

```

64 \cs_new_protected:Npn \__kernel_backend_align_begin:
65 {
66   \__kernel_backend_literal:n { ps::[begin] }
67   \__kernel_backend_literal_postscript:n { currentpoint }
68   \__kernel_backend_literal_postscript:n { currentpoint~translate }
69 }
70 \cs_new_protected:Npn \__kernel_backend_align_end:
71 {
72   \__kernel_backend_literal_postscript:n { neg-exch-neg-exch~translate }
73   \__kernel_backend_literal:n { ps::[end] }
74 }

```

(End definition for `\__kernel_backend_align_begin:` and `\__kernel_backend_align_end:.`)

`\__kernel_backend_scope_begin:` Saving/restoring scope for general operations needs to be done with **dvips** positioning (try without to see this!). Thus we need the `ps:` version of the special here. As only the graphics state is ever altered within this pairing, we use the lower-cost **g**-versions.

```

75 \cs_new_protected:Npn \__kernel_backend_scope_begin:
76 { \__kernel_backend_literal:n { ps:gsave } }
77 \cs_new_protected:Npn \__kernel_backend_scope_end:
78 { \__kernel_backend_literal:n { ps:grestore } }

```

(End definition for `\__kernel_backend_scope_begin:` and `\__kernel_backend_scope_end:.`)

```

79 </dvips>

```

## 1.2 LuaTeX and pdfTeX backends

```

80 <*luatex | pdftex>

```

Both LuaTeX and pdfTeX write PDFs directly rather than via an intermediate file. Although there are similarities, the move of LuaTeX to have more code in Lua means we create two independent files using shared DocStrip code.

`\__kernel_backend_literal_pdf:n` This is equivalent to `\special{pdf:}` but the engine can track it. Without the `direct` keyword everything is kept in sync: the transformation matrix is set to the current point automatically. Note that this is still inside the text (BT ...ET block).

`\__kernel_backend_literal_pdf:x`

```

81 \cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1
82 {
83   <*luatex>
84   \tex_pdfextension:D literal
85   </luatex>
86   <*pdftex>

```

```

87     \tex_pdfliteral:D
88   </pdftex>
89     { \exp_not:n {#1} }
90   }
91   \cs_generate_variant:Nn \__kernel_backend_literal_pdf:n { x }

```

(End definition for \\_\_kernel\_backend\_literal\_pdf:n.)

\\_\_kernel\_backend\_literal\_page:n Page literals are pretty simple. To avoid an expansion, we write out by hand.

```

92   \cs_new_protected:Npn \__kernel_backend_literal_page:n #1
93   {
94     <*luatex>
95       \tex_pdfextension:D literal ~
96     </luatex>
97     <*pdftex>
98       \tex_pdfliteral:D
99     </pdftex>
100       page { \exp_not:n {#1} }
101   }

```

(End definition for \\_\_kernel\_backend\_literal\_page:n.)

\\_\_kernel\_backend\_scope\_begin: Higher-level interfaces for saving and restoring the graphic state.

```

\__kernel_backend_scope_end:
102 \cs_new_protected:Npn \__kernel_backend_scope_begin:
103 {
104 <*luatex>
105   \tex_pdfextension:D save \scan_stop:
106 </luatex>
107 <*pdftex>
108   \tex_pdfsave:D
109 </pdftex>
110 }
111 \cs_new_protected:Npn \__kernel_backend_scope_end:
112 {
113 <*luatex>
114   \tex_pdfextension:D restore \scan_stop:
115 </luatex>
116 <*pdftex>
117   \tex_pdfrestore:D
118 </pdftex>
119 }

```

(End definition for \\_\_kernel\_backend\_scope\_begin: and \\_\_kernel\_backend\_scope\_end:.)

\\_\_kernel\_backend\_matrix:n Here the appropriate function is set up to insert an affine matrix into the PDF. With pdfTeX and LuaTeX in direct PDF output mode there is a primitive for this, which only needs the rotation/scaling/skew part.

```

120 \cs_new_protected:Npn \__kernel_backend_matrix:n #1
121 {
122 <*luatex>
123   \tex_pdfextension:D setmatrix
124 </luatex>
125 <*pdftex>
126   \tex_pdfsetmatrix:D
127 </pdftex>

```

```

128         { \exp_not:n {#1} }
129     }
130 \cs_generate_variant:Nn \__kernel_backend_matrix:n { x }

(End definition for \__kernel_backend_matrix:n.)

131 </luatex | pdftex>

```

### 1.3 dvipdfmx backend

```

132 <*dvipdfmx | xetex>

```

The `dvipdfmx` shares code with the PDF mode one (using the common section to this file) but also with `XYTeX`. The latter is close to identical to `dvipdfmx` and so all of the code here is extracted for both backends, with some `clean up` for `XYTeX` as required. Undocumented but equivalent to `pdfTeX`'s `literal` keyword. It's similar to be not the same as the documented `contents` keyword as that adds a `q/Q` pair.

```

\__kernel_backend_literal_pdf:n
\__kernel_backend_literal_pdf:x

```

```

133 \cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1
134 { \__kernel_backend_literal:n { pdf:literal~ #1 } }
135 \cs_generate_variant:Nn \__kernel_backend_literal_pdf:n { x }

```

(End definition for `\__kernel_backend_literal_pdf:n`.)

```

\__kernel_backend_literal_page:n

```

Whilst the manual says this is like `literal direct` in `pdfTeX`, it closes the BT block!

```

136 \cs_new_protected:Npn \__kernel_backend_literal_page:n #1
137 { \__kernel_backend_literal:n { pdf:literal~direct~ #1 } }

```

(End definition for `\__kernel_backend_literal_page:n`.)

```

\__kernel_backend_scope_begin:
\__kernel_backend_scope_end:

```

Scoping is done using the backend-specific specials. We use the versions originally from `xdvipfmx (x:)` as these are well-tested “in the wild”.

```

138 \cs_new_protected:Npn \__kernel_backend_scope_begin:
139 { \__kernel_backend_literal:n { x:gsave } }
140 \cs_new_protected:Npn \__kernel_backend_scope_end:
141 { \__kernel_backend_literal:n { x:grestore } }

```

(End definition for `\__kernel_backend_scope_begin:` and `\__kernel_backend_scope_end:.`)

```

142 <@@=sys>

```

```

\c__kernel_sys_dvipdfmx_version_int

```

A short excursion into the `sys` module to set up the backend version information.

```

143 \group_begin:
144 \cs_set:Npn \__sys_tmp:w #1 Version ~ #2 ~ #3 \q_stop {#2}
145 \sys_get_shell:nnTF { extractbb---version }
146 { \char_set_catcode_space:n { ‘\ } }
147 \l_sys_internal_tl
148 {
149     \int_const:Nn \c__kernel_sys_dvipdfmx_version_int
150     {
151         \exp_after:wN \__sys_tmp:w \l_sys_internal_tl
152         \q_stop
153     }
154 }
155 { \int_const:Nn \c__kernel_sys_dvipdfmx_version_int { 0 } }
156 \group_end:

```

(End definition for `\c__kernel_sys_dvipdfmx_version_int`.)

```
157 <@@=)
158 </dvipdfmx | xetex>
```

## 1.4 dvisvgm backend

```
159 <*dvisvgm>
```

```
\_kernel_backend_literal_svg:n
\_kernel_backend_literal_svg:x
```

Unlike the other backends, the requirements for making SVG files mean that we can’t conveniently transform all operations to the current point. That makes life a bit more tricky later as that needs to be accounted for. A new line is added after each call to help to keep the output readable for debugging.

```
160 \cs_new_protected:Npn \_kernel_backend_literal_svg:n #1
161 { \_kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }
162 \cs_generate_variant:Nn \_kernel_backend_literal_svg:n { x }
```

(End definition for `\_kernel_backend_literal_svg:n`.)

```
\g__kernel_backend_scope_int
\l__kernel_backend_scope_int
```

In SVG, we need to track scope nesting as properties attach to scopes; that requires a pair of `int` registers.

```
163 \int_new:N \g__kernel_backend_scope_int
164 \int_new:N \l__kernel_backend_scope_int
```

(End definition for `\g__kernel_backend_scope_int` and `\l__kernel_backend_scope_int`.)

```
\_kernel_backend_scope_begin:
\_kernel_backend_scope_end:
\_kernel_backend_scope_begin:n
\_kernel_backend_scope_begin:x
\_kernel_backend_scope:n
\_kernel_backend_scope:x
```

In SVG, the need to attach concepts to a scope means we need to be sure we will close all of the open scopes. That is easiest done if we only need an outer “wrapper” `begin/end` pair, and within that we apply operations as a simple scoped statements. To keep down the non-productive groups, we also have a `begin` version that does take an argument.

```
165 \cs_new_protected:Npn \_kernel_backend_scope_begin:
166 {
167   \_kernel_backend_literal_svg:n { <g> }
168   \int_set_eq:NN
169     \l__kernel_backend_scope_int
170     \g__kernel_backend_scope_int
171   \group_begin:
172     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
173 }
174 \cs_new_protected:Npn \_kernel_backend_scope_end:
175 {
176   \prg_replicate:nn
177     { \g__kernel_backend_scope_int }
178     { \_kernel_backend_literal_svg:n { </g> } }
179   \group_end:
180   \int_gset_eq:NN
181     \g__kernel_backend_scope_int
182     \l__kernel_backend_scope_int
183 }
184 \cs_new_protected:Npn \_kernel_backend_scope_begin:n #1
185 {
186   \_kernel_backend_literal_svg:n { <g ~ #1 > }
187   \int_set_eq:NN
188     \l__kernel_backend_scope_int
```

```

189     \g__kernel_backend_scope_int
190     \group_begin:
191     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
192   }
193   \cs_generate_variant:Nn \__kernel_backend_scope_begin:n { x }
194   \cs_new_protected:Npn \__kernel_backend_scope:n #1
195   {
196     \__kernel_backend_literal_svg:n { <g ~ #1 > }
197     \int_gincr:N \g__kernel_backend_scope_int
198   }
199   \cs_generate_variant:Nn \__kernel_backend_scope:n { x }

```

(End definition for \\_\_kernel\_backend\_scope\_begin: and others.)

```

200 </dvisvgm>
201 </package>

```

## 2 l3backend-box Implementation

```

202 <*package>
203 <@@=box>

```

### 2.1 dvips backend

```

204 <*dvips>

```

\\_\_box\_backend\_clip:N The **dvips** backend scales all absolute dimensions based on the output resolution selected and any T<sub>E</sub>X magnification. Thus for any operation involving absolute lengths there is a correction to make. See **normalscale** from **special.pro** for the variables, noting that here everything is saved on the stack rather than as a separate variable. Once all of that is done, the actual clipping is trivial.

```

205   \cs_new_protected:Npn \__box_backend_clip:N #1
206   {
207     \__kernel_backend_scope_begin:
208     \__kernel_backend_align_begin:
209     \__kernel_backend_literal_postscript:n { matrix~currentmatrix }
210     \__kernel_backend_literal_postscript:n
211     { Resolution~72~div~VResolution~72~div~scale }
212     \__kernel_backend_literal_postscript:n { DVImag~dup~scale }
213     \__kernel_backend_literal_postscript:x
214     {
215       0 ~
216       \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
217       \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
218       \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
219       rectclip
220     }
221     \__kernel_backend_literal_postscript:n { setmatrix }
222     \__kernel_backend_align_end:
223     \hbox_overlap_right:n { \box_use:N #1 }
224     \__kernel_backend_scope_end:
225     \skip_horizontal:n { \box_wd:N #1 }
226   }

```

(End definition for \\_\_box\_backend\_clip:N.)

`\__box_backend_rotate:Nn` Rotating using `dvips` does not require that the box dimensions are altered and has a very convenient built-in operation. Zero rotation must be written as 0 not -0 so there is a quick test.

```

227 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
228 { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
229 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
230 {
231   \__kernel_backend_scope_begin:
232   \__kernel_backend_align_begin:
233   \__kernel_backend_literal_postscript:x
234   {
235     \fp_compare:nNnTF {#2} = \c_zero_fp
236     { 0 }
237     { \fp_eval:n { round ( -(#2) , 5 ) } } ~
238     rotate
239   }
240   \__kernel_backend_align_end:
241   \box_use:N #1
242   \__kernel_backend_scope_end:
243 }

```

(End definition for `\__box_backend_rotate:Nn` and `\__box_backend_rotate_aux:Nn`.)

`\__box_backend_scale:Nnn` The `dvips` backend once again has a dedicated operation we can use here.

```

244 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
245 {
246   \__kernel_backend_scope_begin:
247   \__kernel_backend_align_begin:
248   \__kernel_backend_literal_postscript:x
249   {
250     \fp_eval:n { round ( #2 , 5 ) } ~
251     \fp_eval:n { round ( #3 , 5 ) } ~
252     scale
253   }
254   \__kernel_backend_align_end:
255   \hbox_overlap_right:n { \box_use:N #1 }
256   \__kernel_backend_scope_end:
257 }

```

(End definition for `\__box_backend_scale:Nnn`.)

258 `</dvips>`

## 2.2 LuaTeX and pdfTeX backends

259 `<*luatex | pdftex>`

`\__box_backend_clip:N` The general method is to save the current location, define a clipping path equivalent to the bounding box, then insert the content at the current position and in a zero width box. The “real” width is then made up using a horizontal skip before tidying up. There are other approaches that can be taken (for example using XForm objects), but the logic here shares as much code as possible and uses the same conversions (and so same rounding errors) in all cases.

```

260 \cs_new_protected:Npn \__box_backend_clip:N #1

```



```

261 {
262   \_kernel_backend_scope_begin:
263   \_kernel_backend_literal_pdf:x
264   {
265     0~
266     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
267     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
268     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
269     re~W~n
270   }
271   \hbox_overlap_right:n { \box_use:N #1 }
272   \_kernel_backend_scope_end:
273   \skip_horizontal:n { \box_wd:N #1 }
274 }

```

(End definition for \\_box\_backend\_clip:N.)

\\_box\_backend\_rotate:Nn    Rotations are set using an affine transformation matrix which therefore requires  
\\_box\_backend\_rotate\_aux:Nn    sine/cosine values not the angle itself. We store the rounded values to avoid round-  
  \\_l\\_box\_backend\_cos\_fp    ing twice. There are also a couple of comparisons to ensure that -0 is not written to the  
  \\_l\\_box\_backend\_sin\_fp    output, as this avoids any issues with problematic display programs. Note that numbers  
are compared to 0 after rounding.

```

275 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
276 { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
277 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
278 {
279   \_kernel_backend_scope_begin:
280   \box_set_wd:Nn #1 { Opt }
281   \fp_set:Nn \l\_box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
282   \fp_compare:nNnT \l\_box_backend_cos_fp = \c_zero_fp
283     { \fp_zero:N \l\_box_backend_cos_fp }
284   \fp_set:Nn \l\_box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
285   \_kernel_backend_matrix:x
286   {
287     \fp_use:N \l\_box_backend_cos_fp \c_space_tl
288     \fp_compare:nNnTF \l\_box_backend_sin_fp = \c_zero_fp
289       { 0~0 }
290       {
291         \fp_use:N \l\_box_backend_sin_fp
292         \c_space_tl
293         \fp_eval:n { -\l\_box_backend_sin_fp }
294       }
295     \c_space_tl
296     \fp_use:N \l\_box_backend_cos_fp
297   }
298   \box_use:N #1
299   \_kernel_backend_scope_end:
300 }
301 \fp_new:N \l\_box_backend_cos_fp
302 \fp_new:N \l\_box_backend_sin_fp

```

(End definition for \\_box\_backend\_rotate:Nn and others.)

\\_box\_backend\_scale:Nnn    The same idea as for rotation but without the complexity of signs and cosines.

```

303 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
304 {
305   \__kernel_backend_scope_begin:
306   \__kernel_backend_matrix:x
307   {
308     \fp_eval:n { round ( #2 , 5 ) } ~
309     0~0~
310     \fp_eval:n { round ( #3 , 5 ) }
311   }
312   \hbox_overlap_right:n { \box_use:N #1 }
313   \__kernel_backend_scope_end:
314 }

```

(End definition for \\_\_box\_backend\_scale:Nnn.)

```

315 </luatex | pdftex>

```

## 2.3 dvipdfmx/X<sub>Y</sub>TeX backend

```

316 < *dvipdfmx | xetex>

```

\\_\_box\_backend\_clip:N The code here is identical to that for LuaTeX/pdfTeX: unlike rotation and scaling, there is no higher-level support in the backend for clipping.

```

317 \cs_new_protected:Npn \__box_backend_clip:N #1
318 {
319   \__kernel_backend_scope_begin:
320   \__kernel_backend_literal_pdf:x
321   {
322     0~
323     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
324     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
325     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
326     re~W~n
327   }
328   \hbox_overlap_right:n { \box_use:N #1 }
329   \__kernel_backend_scope_end:
330   \skip_horizontal:n { \box_wd:N #1 }
331 }

```

(End definition for \\_\_box\_backend\_clip:N.)

\\_\_box\_backend\_rotate:Nn Rotating in dvipdfmx/X<sub>Y</sub>TeX can be implemented using either PDF or backend-specific code. The former approach however is not “aware” of the content of boxes: this means that any embedded links would not be adjusted by the rotation. As such, the backend-native approach is preferred: the code therefore is similar (though not identical) to the dvips version (notice the rotation angle here is positive). As for dvips, zero rotation is written as 0 not -0.

```

332 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
333 { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
334 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
335 {
336   \__kernel_backend_scope_begin:
337   \__kernel_backend_literal:x
338   {
339     x:rotate~

```

```

340     \fp_compare:nNnTF {#2} = \c_zero_fp
341     { 0 }
342     { \fp_eval:n { round ( #2 , 5 ) } } }
343   }
344   \box_use:N #1
345   \__kernel_backend_scope_end:
346 }

```

(End definition for \\_\_box\_backend\_rotate:Nn and \\_\_box\_backend\_rotate\_aux:Nn.)

\\_\_box\_backend\_scale:Nnn Much the same idea for scaling: use the higher-level backend operation to allow for box content.

```

347 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
348 {
349   \__kernel_backend_scope_begin:
350   \__kernel_backend_literal:x
351   {
352     x:scale~
353     \fp_eval:n { round ( #2 , 5 ) } ~
354     \fp_eval:n { round ( #3 , 5 ) }
355   }
356   \hbox_overlap_right:n { \box_use:N #1 }
357   \__kernel_backend_scope_end:
358 }

```

(End definition for \\_\_box\_backend\_scale:Nnn.)

359 </dvipdfmx | xetex>

## 2.4 dvisvgm backend

360 <\*dvisvgm>

\\_\_box\_backend\_clip:N  
\g\_\_box\_clip\_path\_int

Clipping in SVG is more involved than with other backends. The first issue is that the clipping path must be defined separately from where it is used, so we need to track how many paths have applied. The naming here uses l3cp as the namespace with a number following. Rather than use a rectangular operation, we define the path manually as this allows it to have a depth: easier than the alternative approach of shifting content up and down using scopes to allow for the depth of the TeX box and keep the reference point the same!

```

361 \cs_new_protected:Npn \__box_backend_clip:N #1
362 {
363   \int_gincr:N \g__box_clip_path_int
364   \__kernel_backend_literal_svg:x
365   { < clipPath-id = " l3cp \int_use:N \g__box_clip_path_int " > }
366   \__kernel_backend_literal_svg:x
367   {
368     <
369     path ~ d =
370     "
371     M ~ 0 ~
372     \dim_to_decimal:n { -\box_dp:N #1 } ~
373     L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
374     \dim_to_decimal:n { -\box_dp:N #1 } ~
375     L ~ \dim_to_decimal:n { \box_wd:N #1 } ~

```

```

376         \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
377         L ~ 0 ~
378         \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
379         Z
380         "
381     />
382 }
383 \__kernel_backend_literal_svg:n
384 { < /clipPath > }

```

In general the SVG set up does not try to transform coordinates to the current point. For clipping we need to do that, so have a transformation here to get us to the right place, and a matching one just before the  $\text{\TeX}$  box is inserted to get things back on track. The clip path needs to come between those two such that it lines up with the current point, as does the  $\text{\TeX}$  box.

```

385 \__kernel_backend_scope_begin:n
386 {
387     transform =
388     "
389         translate ( { ?x } , { ?y } ) ~
390         scale ( 1 , -1 )
391     "
392 }
393 \__kernel_backend_scope:x
394 {
395     clip-path =
396     "url ( \c_hash_str l3cp \int_use:N \g__box_clip_path_int ) "
397 }
398 \__kernel_backend_scope:n
399 {
400     transform =
401     "
402         scale ( -1 , 1 ) ~
403         translate ( { ?x } , { ?y } ) ~
404         scale ( -1 , -1 )
405     "
406 }
407 \box_use:N #1
408 \__kernel_backend_scope_end:
409 }
410 \int_new:N \g__box_clip_path_int

```

(End definition for  $\backslash\_box\_backend\_clip:N$  and  $\backslash g\_box\_clip\_path\_int$ .)

$\backslash\_box\_backend\_rotate:Nn$  Rotation has a dedicated operation which includes a centre-of-rotation optional pair. That can be picked up from the backend syntax, so there is no need to worry about the transformation matrix.

```

411 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
412 {
413     \__kernel_backend_scope_begin:x
414     {
415         transform =
416         "
417         rotate

```

```

418         ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
419     "
420 }
421 \box_use:N #1
422 \__kernel_backend_scope_end:
423 }

```

(End definition for `\__box_backend_rotate:Nn`.)

`\__box_backend_scale:Nnn` In contrast to rotation, we have to account for the current position in this case. That is done using a couple of translations in addition to the scaling (which is therefore done backward with a flip).

```

424 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
425 {
426     \__kernel_backend_scope_begin:x
427     {
428         transform =
429         "
430             translate ( { ?x } , { ?y } ) ~
431             scale
432             (
433                 \fp_eval:n { round ( -#2 , 5 ) } ,
434                 \fp_eval:n { round ( -#3 , 5 ) }
435             ) ~
436             translate ( { ?x } , { ?y } ) ~
437             scale ( -1 )
438         "
439     }
440     \hbox_overlap_right:n { \box_use:N #1 }
441     \__kernel_backend_scope_end:
442 }

```

(End definition for `\__box_backend_scale:Nnn`.)

```

443 </dvisvgm>
444 </package>

```

### 3 l3backend-color Implementation

```

445 <*package>
446 <@@=color>

```

Color support is split into parts: collecting data from L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, the color stack, general color, separations, and color for drawings. We have different approaches in each backend, and have some choices to make about `dvipdfmx`/X<sub>Y</sub>L<sub>A</sub>T<sub>E</sub>X in particular. Whilst it is in some ways convenient to use the same approach in multiple backends, the fact that `dvipdfmx`/X<sub>Y</sub>L<sub>A</sub>T<sub>E</sub>X is PDF-based means it (largely) sticks closer to direct PDF output.

#### 3.1 Collecting information from L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>

##### 3.1.1 dvips-style

```

447 <*dvisvgm | dvipdfmx | dvips | xetex>

```

`\_color_backend_pickup:N` Allow for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> color. Here, the possible input values are limited: dvips-style colors can mainly be taken as-is with the exception spot ones (here we need a model and a tint).  
`\_color_backend_pickup:w` The x-type expansion is there to cover the case where xcolor is in use.

```

448 \cs_new_protected:Npn \_color_backend_pickup:N #1 { }
449 \cs_if_exist:cT { ver@color.sty }
450 {
451   \cs_set_protected:Npn \_color_backend_pickup:N #1
452   {
453     \exp_args:NW \tl_if_head_is_space:nTF \current@color
454     {
455       \tl_set:Nx #1
456       {
457         { \exp_after:wN \use:n \current@color }
458         { 1 }
459       }
460     }
461     {
462       \exp_last_unbraced:Nx \_color_backend_pickup:w
463       { \current@color } \s__color_stop #1
464     }
465   }
466   \cs_new_protected:Npn \_color_backend_pickup:w #1 ~ #2 \s__color_stop #3
467   { \tl_set:Nn #3 { {#1} {#2} } }
468 }

```

(End definition for `\_color_backend_pickup:N` and `\_color_backend_pickup:w`.)

```

469 </divisvgn | dvipdfmx | dvips | xetex>

```

### 3.1.2 Lua<sub>T</sub><sub>E</sub>X and pdf<sub>T</sub><sub>E</sub>X

```

470 < *luatex | pdftex>

```

`\_color_backend_pickup:N` The current color in driver-dependent format: pick up the package-mode data if available. We end up converting back and forward in this route as we store our color data in dvips format. The `\current@color` needs to be x-expanded before `\_color_backend_pickup:w` breaks it apart, because for instance xcolor sets it to be instructions to generate a color  
`\_color_backend_pickup:w`

```

471 \cs_new_protected:Npn \_color_backend_pickup:N #1 { }
472 \cs_if_exist:cT { ver@color.sty }
473 {
474   \cs_set_protected:Npn \_color_backend_pickup:N #1
475   {
476     \exp_last_unbraced:Nx \_color_backend_pickup:w
477     { \current@color } ~ 0 ~ 0 ~ 0 \s__color_stop #1
478   }
479   \cs_new_protected:Npn \_color_backend_pickup:w
480   #1 ~ #2 ~ #3 ~ #4 ~ #5 ~ #6 \s__color_stop #7
481   {
482     \str_if_eq:nnTF {#2} { g }
483     { \tl_set:Nn #7 { { gray } {#1} } }
484     {
485       \str_if_eq:nnTF {#4} { rg }
486       { \tl_set:Nn #7 { { rgb } { #1 ~ #2 ~ #3 } } }

```

```

487         {
488             \str_if_eq:nnTF {#5} { k }
489             { \tl_set:Nn #7 { { cmyk } { #1 ~ #2 ~ #3 ~ #4 } } }
490             {
491                 \str_if_eq:nnTF {#2} { cs }
492                 {
493                     \tl_set:Nx #7 { { \use:n #1 } { #5 } }
494                 }
495                 {
496                     \tl_set:Nn #7 { { gray } { 0 } }
497                 }
498             }
499         }
500     }
501 }
502 }

```

(End definition for `\__color_backend_pickup:N` and `\__color_backend_pickup:w`)

503 `</luatex | pdftex>`

## 3.2 The color stack

For PDF-based engines, we have a color stack available inside the specials. This is used for concepts beyond color itself: it is needed to manage the graphics state generally. The exact form depends on the engine, and for `dvipdfmx/XYTeX` the backend version.

### 3.2.1 Common code

504 `<*dvipdfmx | luatex | pdftex | xetex>`

`\l__color_backend_stack_int` pdf<sub>Y</sub>TeX, Lua<sub>Y</sub>TeX and recent (x)dvipdfmx have multiple stacks available, and to track which one is in use a variable is required.

505 `\int_new:N \l__color_backend_stack_int`

(End definition for `\l__color_backend_stack_int`.)

506 `</dvipdfmx | luatex | pdftex | xetex>`

### 3.2.2 dvipdfmx/X<sub>Y</sub>TeX

507 `<*dvipdfmx | xetex>`

`\__kernel_color_backend_stack_init:Nnn` In (x)dvipdfmx, the base color stack is not set up, so we have to force that, as well as providing a mechanism more generally.

```

\g__color_backend_stack_int
\c__color_backend_main_stack_int
508 \int_compare:nNnTF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
509 { \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3 { } }
510 {
511     \int_new:N \g__color_backend_stack_int
512     \cs_new_protected:Npx \__kernel_color_backend_stack_init:Nnn #1#2#3
513     {
514         \int_gincr:N \exp_not:N \g__color_backend_stack_int
515         \int_const:Nn #1 { \exp_not:N \g__color_backend_stack_int }
516         \use:x
517         {
518             \cs_if_exist:NTF \AtBeginDvi

```

```

519         { \exp_not:N \AtBeginDvi }
520         { \exp_not:N \use:n }
521         {
522             \__kernel_backend_literal:n
523             {
524                 pdfcolorstackinit ~
525                 \exp_not:N \int_use:N \exp_not:N \g__color_backend_stack_int
526                 \c_space_tl
527                 \exp_not:N \tl_if_blank:nF {#2} { #2 ~ }
528                 (#3)
529             }
530         }
531     }
532 }
533 \cs_if_exist:cTF { main@pdfcolorstack }
534 {
535     \int_set:Nn \l__color_backend_stack_int
536     { \int_use:c { main@pdfcolorstack } }
537 }
538 {
539     \__kernel_color_backend_stack_init:Nnn \c__color_backend_main_stack_int
540     { page ~ direct } { 0 ~ g ~ 0 ~ G }
541     \int_set_eq:NN \l__color_backend_stack_int
542     \c__color_backend_main_stack_int
543     \int_const:cn { main@pdfcolorstack } { \c__color_backend_main_stack_int }
544 }

```

The backend automatically restores the stack color from the “classical” approach (`pdf:bcolor`) after a scope. That will be an issue for us, so we manually ensure that the one we are using is inserted.

```

545     \cs_gset_protected:Npn \__kernel_backend_scope_end:
546     {
547         \__kernel_backend_literal:n { x:grestore }
548         \__kernel_backend_literal:n
549         { pdfcolorstack ~ \g__color_backend_stack_int current }
550     }
551 }

```

(End definition for `\__kernel_color_backend_stack_init:Nnn`, `\g__color_backend_stack_int`, and `\c__color_backend_main_stack_int`.)

```

\__kernel_color_backend_stack_push:nn
\__kernel_color_backend_stack_push:nx
\__kernel_color_backend_stack_pop:n

```

Simple enough but needs a version check.

```

552 \int_compare:nNnF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
553 {
554     \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
555     {
556         \__kernel_backend_literal:x
557         {
558             pdfcolorstack ~
559             \int_eval:n {#1} ~
560             push ~ (#2)
561         }
562     }
563     \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
564     \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1

```



```

565     {
566       \__kernel_backend_literal:x
567       {
568         pdfcolorstack ~
569         \int_eval:n {#1} ~
570         pop
571       }
572     }
573   }

```

(End definition for \\_\_kernel\_color\_backend\_stack\_push:nn and \\_\_kernel\_color\_backend\_stack\_pop:n.)

```

574 </dvipdfmx | xetex>

```

### 3.2.3 LuaTeX and pdfTeX

```

575 <*luatex | pdftex>

```

\\_\_kernel\_color\_backend\_stack\_init:Nnn

```

576 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
577 {
578   \int_const:Nn #1
579   {
580     <*luatex>
581     \tex_pdffeedback:D colorstackinit ~
582     </luatex>
583     <*pdftex>
584     \tex_pdfcolorstackinit:D
585     </pdftex>
586     \tl_if_blank:nF {#2} { #2 ~ }
587     {#3}
588   }
589 }

```

(End definition for \\_\_kernel\_color\_backend\_stack\_init:Nnn.)

\\_\_kernel\_color\_backend\_stack\_push:nn

\\_\_kernel\_color\_backend\_stack\_push:nx

\\_\_kernel\_color\_backend\_stack\_pop:n

```

590 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
591 {
592   <*luatex>
593   \tex_pdfextension:D colorstack ~
594   </luatex>
595   <*pdftex>
596   \tex_pdfcolorstack:D
597   </pdftex>
598   \int_eval:n {#1} ~ push ~ {#2}
599 }
600 \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
601 \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
602 {
603   <*luatex>
604   \tex_pdfextension:D colorstack ~
605   </luatex>
606   <*pdftex>
607   \tex_pdfcolorstack:D

```

```

608 </pdfTeX>
609     \int_eval:n {#1} ~ pop \scan_stop:
610 }

(End definition for \__kernel_color_backend_stack_push:nn and \__kernel_color_backend_stack_pop:n.)

611 </luatex | pdfTeX>

```

### 3.3 General color

#### 3.3.1 dvips-style

```

612 <*dvips | dvisvgm>

```

Push the data to the stack. In the case of dvips also saves the drawing color in raw PostScript.

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc
613 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
614 { \__color_backend_select:n { cmyk ~ #1 } }
615 \cs_new_protected:Npn \__color_backend_select_gray:n #1
616 { \__color_backend_select:n { gray ~ #1 } }
617 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
618 { \__color_backend_select:n { rgb ~ #1 } }
619 \cs_new_protected:Npn \__color_backend_select:n #1
620 {
621   \__kernel_backend_literal:n { color~push~ #1 }
622 <*dvips>
623   \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
624 </dvips>
625   \group_insert_after:N \__color_backend_reset:
626 }
627 \cs_new_protected:Npn \__color_backend_reset:
628 { \__kernel_backend_literal:n { color~pop } }

```

(End definition for \\_\_color\_backend\_select\_cmyk:n and others. This function is documented on page ??.)

```

629 </dvips | dvisvgm>

```

#### 3.3.2 LuaTeX and pdfTeX

```

630 <*dvipdfmx | luatex | pdfTeX | xetex>

```

```

\l__color_backend_fill_tl
\l__color_backend_stroke_tl
631 \tl_new:N \l__color_backend_fill_tl
632 \tl_new:N \l__color_backend_stroke_tl

```

(End definition for \l\_\_color\_backend\_fill\_tl and \l\_\_color\_backend\_stroke\_tl.)

Store the values then pass to the stack.

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_select:nn
\__color_backend_reset:
633 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
634 { \__color_backend_select:nn { #1 ~ k } { #1 ~ K } }
635 \cs_new_protected:Npn \__color_backend_select_gray:n #1
636 { \__color_backend_select:nn { #1 ~ g } { #1 ~ G } }
637 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
638 { \__color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
639 \cs_new_protected:Npn \__color_backend_select:nn #1#2

```

```

640 {
641   \tl_set:Nn \l__color_backend_fill_tl {#1}
642   \tl_set:Nn \l__color_backend_stroke_tl {#2}
643   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int { #1 ~ #2 }
644   \group_insert_after:N \__color_backend_reset:
645 }
646 \cs_new_protected:Npn \__color_backend_reset:
647 { \__kernel_color_backend_stack_pop:n \l__color_backend_stack_int }

```

(End definition for \\_\_color\_backend\_select\_cmyk:n and others.)

```

648 </dvipdfmx | luatex | pdftex | xetex>

```

### 3.3.3 dvipdfmx/X<sub>q</sub>TeX

```

649 <*dvipdfmx | xetex>

```

These backends have the most possible approaches: it recognises both dvips-based color specials and it's own format, plus one can include PDF statements directly. Recent releases also have a color stack approach similar to pdfTeX. Of the stack methods, the dedicated the most versatile is the latter as it can cover all of the use cases we have. Thus it is used in preference to the dvips-style interface or the “native” color specials (which have only one stack).

Push the data to the stack.

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_reset:
650 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
651 {
652   \cs_gset_protected:Npn \__color_backend_select_cmyk:n #1
653   {
654     \__kernel_backend_literal:n { pdf: bc ~ [#1] }
655     \group_insert_after:N \__color_backend_reset:
656   }
657   \cs_gset_eq:NN \__color_backend_select_gray:n \__color_backend_select_cmyk:n
658   \cs_gset_eq:NN \__color_backend_select_rgb:n \__color_backend_select_cmyk:n
659   \cs_gset_protected:Npn \__color_backend_reset:
660   { \__kernel_backend_literal:n { pdf: ec } }
661 }

```

(End definition for \\_\_color\_backend\_select\_cmyk:n and others.)

```

662 </dvipdfmx | xetex>

```

## 3.4 Separations

Here, life gets interesting and we need essentially one approach per backend.

```

663 <*dvips>

```

```

\__color_backend_select_separation:nn
\__color_backend_select_devicen:nn
664 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
665 { \__color_backend_select:n { separation ~ #1 ~ #2 } }
666 \cs_new_eq:NN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn

```

(End definition for \\_\_color\_backend\_select\_separation:nn and \\_\_color\_backend\_select\_devicen:nn.)

```

    \_color_backend_separation_init:nnnnn
    \_color_backend_separation_init:nxxnn
    \_color_backend_separation_init_aux:nnnnn
    \_color_backend_separation_init_DeviceCMYK:nnn
    \_color_backend_separation_init_DeviceGray:nnn
    \_color_backend_separation_init_DeviceRGB:nnn
    \_color_backend_separation_init_Device:Nn
    \_color_backend_separation_init:nnn
    \_color_backend_separation_init_count:n
    \_color_backend_separation_init_count:w
    \_color_backend_separation_init:nnnn
    \_color_backend_separation_init:w
    \_color_backend_separation_init:n
    \_color_backend_separation_init:nw
    \_color_backend_separation_init_CIELAB:nnn

```

Initialising here means creating a small header set up plus massaging some data. This comes about as we have to deal with PDF-focussed data, which makes most sense “higher-up”. The approach is based on ideas from <https://tex.stackexchange.com/q/560093> plus using the PostScript manual for other aspects.

```

667 \cs_new_protected:Npx \_color_backend_separation_init:nnnnn #1#2#3#4#5
668 {
669     \bool_if:NT \g__kernel_backend_header_bool
670     {
671         \cs_if_exist:NTF \AtBeginDvi
672         { \exp_not:N \AtBeginDvi }
673         { \use:n }
674         {
675             \exp_not:N \_color_backend_separation_init_aux:nnnnn
676             {#1} {#2} {#3} {#4} {#5}
677         }
678     }
679 }
680 \cs_generate_variant:Nn \_color_backend_separation_init:nnnnn { nxx }
681 \cs_new_protected:Npn \_color_backend_separation_init_aux:nnnnn #1#2#3#4#5
682 {
683     \_kernel_backend_literal:e
684     {
685         !
686         TeXDict ~ begin ~
687         /color \int_use:N \g__color_model_int
688         {
689             [ ~
690             /Separation ~ ( \str_convert_pdfname:n {#1} ) ~
691             [ ~ #2 ~ ] ~
692             {
693                 \cs_if_exist_use:cF { \_color_backend_separation_init_ #2 :nnn }
694                 { \_color_backend_separation_init:nnn }
695                 {#3} {#4} {#5}
696             }
697             ] ~ setcolorspace
698             } ~ def ~
699         end
700     }
701 }
702 \cs_new:cpn { \_color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
703 { \_color_backend_separation_init_Device:Nn 4 {#3} }
704 \cs_new:cpn { \_color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
705 { \_color_backend_separation_init_Device:Nn 1 {#3} }
706 \cs_new:cpn { \_color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
707 { \_color_backend_separation_init_Device:Nn 2 {#3} }
708 \cs_new:Npn \_color_backend_separation_init_Device:Nn #1#2
709 {
710     #2 ~
711     \prg_replicate:nn {#1}
712     { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
713     \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
714 }

```

For the generic case, we cannot use /FunctionType 2 unfortunately, so we have to code

that idea up in PostScript. Here, we will therefore assume that a range is *always* given. First, we count values in each argument: at the backend level, we can assume there are always well-behaved with spaces present.

```

715 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3
716 {
717   \exp_args:Ne \__color_backend_separation_init:nnnn
718   { \__color_backend_separation_init_count:n {#2} }
719   {#1} {#2} {#3}
720 }
721 \cs_new:Npn \__color_backend_separation_init_count:n #1
722 { \int_eval:n { 0 \__color_backend_separation_init_count:w #1 ~ \s__color_stop } }
723 \cs_new:Npn \__color_backend_separation_init_count:w #1 ~ #2 \s__color_stop
724 {
725   +1
726   \tl_if_blank:nF {#2}
727   { \__color_backend_separation_init_count:w #2 \s__color_stop }
728 }

```

Now we implement the algorithm. In the terms in the PostScript manual, we have **N** = 1 and **Domain** = [0 1], with **Range** as #2, **C0** as #3 and **C1** as #4, with the number of output components in #1. So all we have to do is implement  $y_i = \mathbf{C0}_i + x(\mathbf{C1}_i - \mathbf{C0}_i)$  with lots of stack manipulation, then check the ranges. That's done by adding everything to the stack first, then using the fact we know all of the offsets. As manipulating the stack is tricky, we start by re-formatting the **C0** and **C1** arrays to be interleaved, and add a 0 to each pair: this is used to keep the stack of constant length while we are doing the first pass of mathematics. We then working through that list, calculating from the last to the first value before tidying up by removing all of the input values. We do that by first copying all of the final  $y$  values to the end of the stack, then rolling everything so we can pop the now-unneeded material.

```

729 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
730 {
731   \__color_backend_separation_init:w #3 ~ \s__color_stop #4 ~ \s__color_stop
732   \prg_replicate:nn {#1}
733   {
734     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
735     \int_eval:n { 3 * #1 } ~ index ~ mul ~
736     2 ~ index ~ add ~
737     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
738   }
739   \int_step_function:nnnN {#1} { -1 } { 1 }
740   \__color_backend_separation_init:n
741   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
742   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
743   \tl_if_blank:nF {#2}
744   { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
745 }
746 \cs_new:Npn \__color_backend_separation_init:w
747 #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
748 {
749   #1 ~ #3 ~ 0 ~
750   \tl_if_blank:nF {#2}
751   { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }
752 }

```

```

753 \cs_new:Npn \__color_backend_separation_init:n #1
754 { \int_eval:n { #1 * 2 } ~ index ~ }

```

Finally, we deal with the range limit if required. This is handled by splitting the range into pairs. It's then just a question of doing the comparisons, this time dropping everything except the desired result.

```

755 \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s_color_stop
756 {
757     #2 ~ #3 ~
758     2 ~ index ~ 2 ~ index ~ lt ~
759     { ~ pop ~ excl ~ pop ~ } ~
760     { ~
761         2 ~ index ~ 1 ~ index ~ gt ~
762         { ~ excl ~ pop ~ excl ~ pop ~ } ~
763         { ~ pop ~ pop ~ } ~
764         ifelse ~
765     }
766     ifelse ~
767     #1 ~ 1 ~ roll ~
768     \tl_if_blank:nF {#4}
769     { \__color_backend_separation_init:nw {#1} #4 \s_color_stop }
770 }

```

CIELAB support uses the detail from the PostScript reference, page 227; other than that block of PostScript, this is the same as for PDF-based routes.

```

771 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
772 {
773     \__color_backend_separation_init:nxxxnn
774     {#2}
775     {
776         /CIEBasedABC ~
777         << ~
778         /RangeABC ~ [ ~ \c__color_model_range_CIELAB_tl \c_space_tl ] ~
779         /DecodeABC ~
780         [ ~
781             { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
782             { ~ 500 ~ div ~ } ~ bind ~
783             { ~ 200 ~ div ~ } ~ bind ~
784         ] ~
785         /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
786         /DecodeLMN ~
787         [ ~
788             { ~
789                 dup ~ 6 ~ 29 ~ div ~ ge ~
790                 { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
791                 { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
792                 ifelse ~
793                 0.9505 ~ mul ~
794             } ~ bind ~
795             { ~
796                 dup ~ 6 ~ 29 ~ div ~ ge ~
797                 { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
798                 { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
799                 ifelse ~
800             } ~ bind ~

```

```

801         { ~
802         dup ~ 6 ~ 29 ~ div ~ ge ~
803         { ~ dup ~ dup ~ mul ~ mul ~ } ~
804         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
805         ifelse ~
806         1.0890 ~ mul ~
807         } ~ bind
808       ] ~
809       /WhitePoint ~
810       [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
811       >>
812     }
813     { \c__color_model_range_CIELAB_tl }
814     { 100 ~ 0 ~ 0 }
815     {#3}
816   }

```

(End definition for \\_color\_backend\_separation\_init:nnnnn and others.)

\\_color\_backend\_devicen\_init:nnn

Trivial as almost all of the work occurs in the shared code.

```

817 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
818 {
819   \_kernel_backend_literal:e
820   {
821     !
822     TeXDict ~ begin ~
823     /color \int_use:N \g__color_model_int
824     {
825       [ ~
826       /DeviceN ~
827       [ ~ #1 ~ ] ~
828       #2 ~
829       { ~ #3 ~ } ~
830       ] ~ setcolorspace
831     } ~ def ~
832   } end
833 }
834 }

```

(End definition for \\_color\_backend\_devicen\_init:nnn.)

```

835 </dvips>
836 <*dvisvgm>

```

\\_color\_backend\_select\_separation:nn

No support at present.

\\_color\_backend\_select\_devicen:nn

```

837 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2 { }
838 \cs_new_protected:Npn \_color_backend_select_devicen:nn #1#2 { }

```

(End definition for \\_color\_backend\_select\_separation:nn and \\_color\_backend\_select\_devicen:nn.)

\\_color\_backend\_separation\_init:nnnnn

No support at present.

\\_color\_backend\_separation\_init\_CIELAB:nnn

```

839 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }
840 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnnnnn #1#2#3 { }

```

(End definition for `\_color_backend_separation_init:nnnnn` and `\_color_backend_separation_init_CIELAB:nnn`.)

841 `</dvisvgm>`

842 `<*dvipdfmx | luatex | pdftex | xetex>`

`\_color_backend_select_separation:nn`  
`\_color_backend_select_devicen:nn`

Although (x)dvipdfmx has a built-in approach to color spaces, that can't be used with the generic color stacks. So we take an approach in which we share the same code as for pdfTeX.

843 `\cs_new_protected:Npn \_color_backend_select_separation:nn #1#2`  
844 `{ \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }`  
845 `\cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn`

(End definition for `\_color_backend_select_separation:nn` and `\_color_backend_select_devicen:nn`.)

`\_color_backend_separation_init:nnnnn`  
`\_color_backend_separation_init:n`  
`\_color_backend_separation_init_CIELAB:nnn`

Initialising the PDF structures needs two parts: creating an object containing the “real” name of the Separation, then adding a reference to that to each page. We use a separate object for the tint transformation following the model in the PDF reference.

846 `\cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5`  
847 `{`  
848 `\pdf_object_unnamed_write:nx { dict }`  
849 `{`  
850 `/FunctionType ~ 2`  
851 `/Domain ~ [0 ~ 1]`  
852 `\tl_if_blank:nF {#3} { /Range ~ [#3] }`  
853 `/C0 ~ [#4] ~`  
854 `/C1 ~ [#5] /N ~ 1`  
855 `}`  
856 `\_color_backend_separation_init:n`  
857 `{`  
858 `/Separation ~`  
859 `/ \str_convert_pdfname:n {#1} ~ #2 ~`  
860 `\pdf_object_ref_last:`  
861 `}`  
862 `\cs_if_exist:NT \pdfmanagement_add:nnn`  
863 `{`  
864 `\use:x`  
865 `{`  
866 `\pdfmanagement_add:nnn`  
867 `{ Page / Resources / ColorSpace }`  
868 `{ color \int_use:N \g_color_model_int }`  
869 `{ \pdf_object_ref_last: }`  
870 `}`  
871 `}`  
872 `}`  
873 `\cs_new_protected:Npn \_color_backend_separation_init:n #1`  
874 `{`  
875 `\pdf_object_unnamed_write:nx { array } {#1}`  
876 `}`

For CIELAB colors, we need one object per document for the illuminant, plus initialisation of the color space referencing that object.

877 `\cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3`  
878 `{`



```

879 \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
880 {
881   \pdf_object_new:nn { __color_illuminant_CIELAB_ #1 } { array }
882   \pdf_object_write:nx { __color_illuminant_CIELAB_ #1 }
883   {
884     /Lab ~
885     <<
886     /WhitePoint ~
887     [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
888     /Range ~ [ \c__color_model_range_CIELAB_tl ]
889     >>
890   }
891 }
892 \__color_backend_separation_init:nnnnn
893 {#2}
894 { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
895 { \c__color_model_range_CIELAB_tl }
896 { 100 ~ 0 ~ 0 }
897 {#3}
898 }
899 \cs_if_exist:NF \pdf_object_unnamed_write:nn
900 {
901   \cs_gset_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
902   { }
903 }

```

(End definition for \\_\_color\_backend\_separation\_init:nnnnn, \\_\_color\_backend\_separation\_init:n, and \\_\_color\_backend\_separation\_init\_CIELAB:nnn.)

\\_\_color\_backend\_devicen\_init:nnn  
 \\_\_color\_backend\_devicen\_init:w  
 \\_\_color\_backend\_devicen\_init:n

Similar to the Separations case, but with an arbitrary function for the alternative space work.

```

904 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
905 {
906   \pdf_object_unnamed_write:nx { stream }
907   {
908     {
909       /FunctionType ~ 4 ~
910       /Domain ~
911       [ ~
912         \prg_replicate:nn
913         { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
914         { 0 ~ 1 ~ } ~
915       ] ~
916       /Range ~
917       [ ~
918         \str_case:nn {#2}
919         {
920           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
921           { /DeviceGray } { 0 ~ 1 }
922           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
923         } ~
924       ]
925     }
926     {#3}

```

```

927     }
928     \_color_backend_separation_init:n
929     {
930         /DeviceN ~
931         [ ~ #1 ~ ] ~
932         #2 ~
933         \pdf_object_ref_last:
934     }
935     \cs_if_exist:NT \pdfmanagement_add:nnn
936     {
937         \use:x
938         {
939             \pdfmanagement_add:nnn
940             { Page / Resources / ColorSpace }
941             { color \int_use:N \g_color_model_int }
942             { \pdf_object_ref_last: }
943         }
944     }
945 }
946 \cs_new:Npn \_color_backend_devicen_init:w #1 ~ #2 \s_color_stop
947 {
948     + 1
949     \tl_if_blank:nF {#2}
950     { \_color_backend_devicen_init:w #2 \s_color_stop }
951 }
952 \cs_new_eq:NN \_color_backend_devicen_init:n \_color_backend_separation_init:n
(End definition for \_color_backend_devicen_init:nnn, \_color_backend_devicen_init:w, and \_color_backend_devicen_init:n.)
953 </dvipdfmx | luatex | pdftex | xetex>
954 <*dvipdfmx | xetex>

```

For older (x)dvipdfmx, we *could* support separations using a dedicated mechanism, but it was not added that long before the color stacks. So instead of having two complex paths, just disable here.

```

955 \int_compare:nNnT \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
956 {
957     \cs_gset_protected:Npn \_color_backend_select_separation:nn #1#2 { }
958     \cs_gset_eq:NN \_color_backend_select_devicen:nn
959     \_color_backend_select_separation:nn
960 }
(End definition for \_color_backend_select_separation:nn and \_color_backend_select_devicen:nn.)
961 </dvipdfmx | xetex>

```

### 3.5 Fill and stroke color

Here, dvipdfmx/X<sub>q</sub>TeX follows LuaTeX and pdfTeX, while for dvips we have to manage fill and stroke color ourselves. We also handle dvisvgm independently, as there we can create SVG directly.

```

962 <*dvipdfmx | luatex | pdftex | xetex>

```

Drawing (fill/stroke) color is handled in dvipdfmx/X<sub>Y</sub>TeX in the same way as LuaTeX/pdfTeX. We use the same approach as earlier, except the color stack is not involved so the generic direct PDF operation is used. There is no worry about the nature of strokes: everything is handled automatically.

```

\__color_backend_fill_cmyk:n
\__color_backend_fill_gray:n
\__color_backend_fill_rgb:n
  \__color_backend_fill:n
    \__color_backend_stroke_cmyk:n
    \__color_backend_stroke_gray:n
    \__color_backend_stroke_rgb:n
  \__color_backend_stroke:n
963 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
964   { \__color_backend_fill:n { #1 ~ k } }
965 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
966   { \__color_backend_fill:n { #1 ~ g } }
967 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
968   { \__color_backend_fill:n { #1 ~ rg } }
969 \cs_new_protected:Npn \__color_backend_fill:n #1
970   {
971     \tl_set:Nn \l__color_backend_fill_tl {#1}
972     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
973     { #1 ~ \l__color_backend_stroke_tl }
974     \group_insert_after:N \__color_backend_reset:
975   }
976 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
977   { \__color_backend_stroke:n { #1 ~ K } }
978 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
979   { \__color_backend_stroke:n { #1 ~ G } }
980 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
981   { \__color_backend_stroke:n { #1 ~ RG } }
982 \cs_new_protected:Npn \__color_backend_stroke:n #1
983   {
984     \tl_set:Nn \l__color_backend_stroke_tl {#1}
985     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
986     { \l__color_backend_fill_tl \c_space_tl #1 }
987     \group_insert_after:N \__color_backend_reset:
988   }

(End definition for \__color_backend_fill_cmyk:n and others.)

\__color_backend_fill_separation:nn
\__color_backend_stroke_separation:nn
  \__color_backend_fill_devicen:nn
  \__color_backend_stroke_devicen:nn
989 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
990   { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
991 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
992   { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
993 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
994 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

(End definition for \__color_backend_fill_separation:nn and others.)

995 </dvipdfmx | luatex | pdftex | xetex>
996 <*dvipdfmx | xetex>

\__color_backend_fill_cmyk:n Deal with older (x)dvipdfmx.
\__color_backend_fill_gray:n
\__color_backend_fill_rgb:n
  \__color_backend_reset:
  \__color_backend_stroke:n
    \__color_backend_fill_separation:nn
    \__color_backend_stroke_separation:nn
1000   {
1001     \__kernel_backend_literal:n { pdf: bc ~ [#1] }
1002     \group_insert_after:N \__color_backend_reset:
1003   }

```

```

1004 \cs_gset_eq:NN \_color_backend_fill_gray:n \_color_backend_fill_cmyk:n
1005 \cs_gset_eq:NN \_color_backend_fill_rgb:n \_color_backend_fill_cmyk:n
1006 \cs_gset_protected:Npn \_color_backend_reset:
1007 { \_kernel_backend_literal:n { pdf: ec } }
1008 \cs_gset_protected:Npn \_color_backend_stroke:n #1
1009 { \_kernel_backend_literal:n {#1} }
1010 \cs_gset_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1011 \cs_gset_eq:NN \_color_backend_fill_devicen:nn
1012 \_color_backend_fill_separation:nn
1013 \cs_gset_eq:NN \_color_backend_stroke_separation:nn
1014 \_color_backend_fill_separation:nn
1015 \cs_gset_eq:NN \_color_backend_stroke_devicen:nn
1016 \_color_backend_stroke_separation:nn
1017 }

```

(End definition for \\_color\_backend\_fill\_cmyk:n and others.)

```

1018 </dvipdfmx | xetex>

```

```

1019 <*dvips>

```

\\_color\_backend\_fill\_cmyk:n Fill color here is the same as general color *except* we skip the stroke part.

```

\_color_backend_fill_gray:n 1020 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
\_color_backend_fill_rgb:n 1021 { \_color_backend_fill:n { cmyk ~ #1 } }
\_color_backend_fill:n 1022 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
\_color_backend_stroke_cmyk:n 1023 { \_color_backend_fill:n { gray ~ #1 } }
\_color_backend_stroke_gray:n 1024 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
\_color_backend_stroke_rgb:n 1025 { \_color_backend_fill:n { rgb ~ #1 } }
1026 \cs_new_protected:Npn \_color_backend_fill:n #1
1027 {
1028 \_kernel_backend_literal:n { color~push~ #1 }
1029 \group_insert_after:N \_color_backend_reset:
1030 }
1031 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1032 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1033 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1034 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1035 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1036 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }

```

(End definition for \\_color\_backend\_fill\_cmyk:n and others.)

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn 1037 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
\_color_backend_fill_devicen:nn 1038 { \_color_backend_fill:n { separation ~ #1 ~ #2 } }
\_color_backend_stroke_devicen:nn 1039 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1040 { \_kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1041 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1042 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

(End definition for \\_color\_backend\_fill\_separation:nn and others.)

```

1043 </dvips>

```

```

1044 <*dvisvgm>

```

`\_color_backend_fill_cmyk:n` Fill color here is the same as general color *except* we skip the stroke part.

```

\_color_backend_fill_gray:n 1045 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
\_color_backend_fill_rgb:n 1046 { \_color_backend_fill:n { cmyk ~ #1 } }
\_color_backend_fill:n 1047 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1048 { \_color_backend_fill:n { gray ~ #1 } }
1049 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1050 { \_color_backend_fill:n { rgb ~ #1 } }
1051 \cs_new_protected:Npn \_color_backend_fill:n #1
1052 {
1053 \_kernel_backend_literal:n { color~push~ #1 }
1054 \group_insert_after:N \_color_backend_reset:
1055 }

```

(End definition for `\_color_backend_fill_cmyk:n` and others.)

`\_color_backend_stroke_cmyk:n` For drawings in SVG, we use scopes for all stroke colors. That requires using RGB values, which luckily are easy to convert here (cmyk to RGB is a fixed function).

```

\_color_backend_stroke_cmyk:w 1056 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
\_color_backend_stroke_gray:n 1057 { \_color_backend_cmyk:w #1 \s__color_stop }
\_color_backend_stroke_gray_aux:n 1058 \cs_new_protected:Npn \_color_backend_stroke_cmyk:w
\_color_backend_stroke_rgb:n 1059 #1 ~ #2 ~ #3 ~ #4 \s__color_stop
\_color_backend_stroke_rgb:w 1060 {
\_color_backend:nnn 1061 \use:x
1062 {
1063 \_color_backend:nnn
1064 { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1065 { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1066 { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }
1067 }
1068 }
1069 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1070 {
1071 \use:x
1072 {
1073 \_color_backend_stroke_gray_aux:n
1074 { \fp_eval:n { 100 * (#1) } }
1075 }
1076 }
1077 \cs_new_protected:Npn \_color_backend_stroke_gray_aux:n #1
1078 { \_color_backend:nnn {#1} {#1} {#1} }
1079 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1080 { \_color_backend_rgb:w #1 \s__color_stop }
1081 \cs_new_protected:Npn \_color_backend_stroke_rgb:w
1082 #1 ~ #2 ~ #3 \s__color_stop
1083 {
1084 \use:x
1085 {
1086 \_color_backend:nnn
1087 { \fp_eval:n { 100 * (#1) } }
1088 { \fp_eval:n { 100 * (#2) } }
1089 { \fp_eval:n { 100 * (#3) } }
1090 }
1091 }
1092 \cs_new_protected:Npx \_color_backend:nnn #1#2#3

```

```

1093 {
1094   \_kernel_backend_scope:n
1095   {
1096     stroke =
1097     "
1098     rgb
1099     (
1100       #1 \c_percent_str ,
1101       #2 \c_percent_str ,
1102       #3 \c_percent_str
1103     )
1104     "
1105   }
1106 }

```

(End definition for \\_color\_backend\_stroke\_cmyk:n and others.)

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
1107 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1108 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2 { }
1109 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1110 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

(End definition for \\_color\_backend\_fill\_separation:nn and others.)

```

1111 </dvisvgm>
1112 </package>

```

## 4 l3backend-draw Implementation

```

1113 <*package>
1114 <@@=draw>

```

### 4.1 dvips backend

```

1115 <*dvips>

```

```

\_draw_backend_literal:n
\_draw_backend_literal:x
1116 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
1117 \cs_generate_variant:Nn \_draw_backend_literal:n { x }

```

(End definition for \\_draw\_backend\_literal:n.)

```

\_draw_backend_begin:
  \_draw_backend_end:

```

The **ps::[begin]** special here deals with positioning but allows us to continue on to a matching **ps::[end]**: contrast with **ps:**, which positions but where we can't split material between separate calls. The **@beginspecial/@endspecial** pair are from **special.pro** and correct the scale and *y*-axis direction. In contrast to **pgf**, we don't save the current point: discussion with Tom Rokici suggested a better way to handle the necessary translations (see **\\_draw\_backend\_box\_use:Nnnnn**). (Note that **@beginspecial/@endspecial** forms a backend scope.) The **[begin]/[end]** lines are handled differently from the rest as they are conceptually different: not really drawing literals but instructions to **dvips** itself.

```

1118 \cs_new_protected:Npn \_draw_backend_begin:
1119 {

```

```

1120     \__kernel_backend_literal:n { ps::[begin] }
1121     \__draw_backend_literal:n { @beginspecial }
1122   }
1123   \cs_new_protected:Npn \__draw_backend_end:
1124   {
1125     \__draw_backend_literal:n { @endspecial }
1126     \__kernel_backend_literal:n { ps::[end] }
1127   }

```

(End definition for \\_\_draw\_backend\_begin: and \\_\_draw\_backend\_end:.)

\\_\_draw\_backend\_scope\_begin: Scope here may need to contain saved definitions, so the entire memory rather than just  
 \\_\_draw\_backend\_scope\_end: the graphic state has to be sent to the stack.

```

1128   \cs_new_protected:Npn \__draw_backend_scope_begin:
1129   { \__draw_backend_literal:n { save } }
1130   \cs_new_protected:Npn \__draw_backend_scope_end:
1131   { \__draw_backend_literal:n { restore } }

```

(End definition for \\_\_draw\_backend\_scope\_begin: and \\_\_draw\_backend\_scope\_end:.)

\\_\_draw\_backend\_moveto:nn Path creation operations mainly resolve directly to PostScript primitive steps, with only  
 \\_\_draw\_backend\_lineto:nn the need to convert to bp. Notice that x-type expansion is included here to ensure that  
 \\_\_draw\_backend\_rectangle:nnnn any variable values are forced to literals before any possible caching. There is no native  
 \\_\_draw\_backend\_curveto:nnnnnn rectangular path command (without also clipping, filling or stroking), so that task is  
 done using a small amount of PostScript.

```

1132   \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1133   {
1134     \__draw_backend_literal:x
1135     {
1136       \dim_to_decimal_in_bp:n {#1} ~
1137       \dim_to_decimal_in_bp:n {#2} ~ moveto
1138     }
1139   }
1140   \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1141   {
1142     \__draw_backend_literal:x
1143     {
1144       \dim_to_decimal_in_bp:n {#1} ~
1145       \dim_to_decimal_in_bp:n {#2} ~ lineto
1146     }
1147   }
1148   \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1149   {
1150     \__draw_backend_literal:x
1151     {
1152       \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1153       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1154       moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1155     }
1156   }
1157   \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1158   {
1159     \__draw_backend_literal:x
1160     {

```

```

1161         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1162         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1163         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1164         curveto
1165     }
1166 }

```

(End definition for `\__draw_backend_moveto:nn` and others.)

```

\__draw_backend_evenodd_rule: The even-odd rule here can be implemented as a simply switch.
\__draw_backend_nonzero_rule:
\g__draw_draw_eor_bool
1167 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1168 { \bool_gset_true:N \g__draw_draw_eor_bool }
1169 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1170 { \bool_gset_false:N \g__draw_draw_eor_bool }
1171 \bool_new:N \g__draw_draw_eor_bool

```

(End definition for `\__draw_backend_evenodd_rule:`, `\__draw_backend_nonzero_rule:`, and `\g__draw_draw_eor_bool`.)

```

\__draw_backend_closepath: Unlike PDF, PostScript doesn't track separate colors for strokes and other elements. It is
\__draw_backend_stroke: also desirable to have the clip keyword after a stroke or fill. To achieve those outcomes,
\__draw_backend_closestroke: there is some work to do. For color, the stroke color is simple but the fill one has to be
\__draw_backend_fill: inserted by hand. For clipping, the required ordering is achieved using a TEX switch.
\__draw_backend_fillstroke: All of the operations end with a new path instruction as they do not terminate (again in
\__draw_backend_clip: contrast to PDF).
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
1172 \cs_new_protected:Npn \__draw_backend_closepath:
1173 { \__draw_backend_literal:n { closepath } }
1174 \cs_new_protected:Npn \__draw_backend_stroke:
1175 {
1176     \__draw_backend_literal:n { gsave }
1177     \__draw_backend_literal:n { color.sc }
1178     \__draw_backend_literal:n { stroke }
1179     \__draw_backend_literal:n { grestore }
1180     \bool_if:NT \g__draw_draw_clip_bool
1181     {
1182         \__draw_backend_literal:x
1183         {
1184             \bool_if:NT \g__draw_draw_eor_bool { eo }
1185             clip
1186         }
1187     }
1188     \__draw_backend_literal:n { newpath }
1189     \bool_gset_false:N \g__draw_draw_clip_bool
1190 }
1191 \cs_new_protected:Npn \__draw_backend_closestroke:
1192 {
1193     \__draw_backend_closepath:
1194     \__draw_backend_stroke:
1195 }
1196 \cs_new_protected:Npn \__draw_backend_fill:
1197 {
1198     \__draw_backend_literal:x
1199     {
1200         \bool_if:NT \g__draw_draw_eor_bool { eo }

```



```

1201         fill
1202     }
1203     \bool_if:NT \g__draw_draw_clip_bool
1204     {
1205         \__draw_backend_literal:x
1206         {
1207             \bool_if:NT \g__draw_draw_eor_bool { eo }
1208             clip
1209         }
1210     }
1211     \__draw_backend_literal:n { newpath }
1212     \bool_gset_false:N \g__draw_draw_clip_bool
1213 }
1214 \cs_new_protected:Npn \__draw_backend_fillstroke:
1215 {
1216     \__draw_backend_literal:x
1217     {
1218         \bool_if:NT \g__draw_draw_eor_bool { eo }
1219         fill
1220     }
1221     \__draw_backend_literal:n { gsave }
1222     \__draw_backend_literal:n { color.sc }
1223     \__draw_backend_literal:n { stroke }
1224     \__draw_backend_literal:n { grestore }
1225     \bool_if:NT \g__draw_draw_clip_bool
1226     {
1227         \__draw_backend_literal:x
1228         {
1229             \bool_if:NT \g__draw_draw_eor_bool { eo }
1230             clip
1231         }
1232     }
1233     \__draw_backend_literal:n { newpath }
1234     \bool_gset_false:N \g__draw_draw_clip_bool
1235 }
1236 \cs_new_protected:Npn \__draw_backend_clip:
1237 { \bool_gset_true:N \g__draw_draw_clip_bool }
1238 \bool_new:N \g__draw_draw_clip_bool
1239 \cs_new_protected:Npn \__draw_backend_discardpath:
1240 {
1241     \bool_if:NT \g__draw_draw_clip_bool
1242     {
1243         \__draw_backend_literal:x
1244         {
1245             \bool_if:NT \g__draw_draw_eor_bool { eo }
1246             clip
1247         }
1248     }
1249     \__draw_backend_literal:n { newpath }
1250     \bool_gset_false:N \g__draw_draw_clip_bool
1251 }

```

(End definition for \\_\_draw\_backend\_closepath: and others.)

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_but:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:

```

Converting paths to output is again a case of mapping directly to PostScript operations.

```

1252 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1253 {
1254   \__draw_backend_literal:x
1255   {
1256     [
1257       \exp_args:Nf \use:n
1258       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1259     ] ~
1260     \dim_to_decimal_in_bp:n {#2} ~ setdash
1261   }
1262 }
1263 \cs_new:Npn \__draw_backend_dash:n #1
1264 { ~ \dim_to_decimal_in_bp:n {#1} }
1265 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1266 {
1267   \__draw_backend_literal:x
1268   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1269 }
1270 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1271 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1272 \cs_new_protected:Npn \__draw_backend_cap_but:
1273 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1274 \cs_new_protected:Npn \__draw_backend_cap_round:
1275 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1276 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1277 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1278 \cs_new_protected:Npn \__draw_backend_join_miter:
1279 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1280 \cs_new_protected:Npn \__draw_backend_join_round:
1281 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1282 \cs_new_protected:Npn \__draw_backend_join_bevel:
1283 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

(End definition for \\_\_draw\_backend\_dash\_pattern:nn and others.)

\\_\_draw\_backend\_cm:nnnn In dvips, keeping the transformations in line with the engine is unfortunately not possible for scaling and rotations: even if we decompose the matrix into those operations, there is still no backend tracking (cf. dvipdfmx/X<sub>Y</sub>TEX). Thus we take the shortest path available and simply dump the matrix as given.

```

1284 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1285 {
1286   \__draw_backend_literal:n
1287   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1288 }

```

(End definition for \\_\_draw\_backend\_cm:nnnn.)

\\_\_draw\_backend\_box\_use:Nnnnn Inside a picture @beginspecial/@endspecial are active, which is normally a good thing but means that the position and scaling would be off if the box was inserted directly. To deal with that, there are a number of possible approaches. The implementation here was suggested by Tom Rokici (author of dvips). We end the current special placement, then set the current point with a literal [begin]. As for general literals, we then use the stack to store the current point and move to it. To insert the required transformation, we have

to flip the  $y$ -axis, once before and once after it. Then we get back to the  $\text{\TeX}$  reference point to insert our content. The clean up has to happen in the right places, hence the `[begin]/[end]` pair around `restore`. Finally, we can return to “normal” drawing mode. Notice that the set up here is very similar to that in `\__draw_align_currentpoint_...`, but the ordering of saving and restoring is different (intermixed).

```

1289 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1290 {
1291   \__draw_backend_literal:n { @endspecial }
1292   \__draw_backend_literal:n { [end] }
1293   \__draw_backend_literal:n { [begin] }
1294   \__draw_backend_literal:n { save }
1295   \__draw_backend_literal:n { currentpoint }
1296   \__draw_backend_literal:n { currentpoint~translate }
1297   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1298   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1299   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1300   \__draw_backend_literal:n { neg~exch~neg~exch~translate }
1301   \__draw_backend_literal:n { [end] }
1302   \hbox_overlap_right:n { \box_use:N #1 }
1303   \__draw_backend_literal:n { [begin] }
1304   \__draw_backend_literal:n { restore }
1305   \__draw_backend_literal:n { [end] }
1306   \__draw_backend_literal:n { [begin] }
1307   \__draw_backend_literal:n { @beginspecial }
1308 }

```

(End definition for `\__draw_backend_box_use:Nnnnn`.)

```

1309 </dvips>

```

## 4.2 Lua $\text{\TeX}$ , pdf $\text{\TeX}$ , dvipdfmx and X $\text{\TeX}$

Lua $\text{\TeX}$ , pdf $\text{\TeX}$ , dvipdfmx and X $\text{\TeX}$  directly produce PDF output and understand a shared set of specials for drawing commands.

```

1310 <*dvipdfmx | luatex | pdftex | xetex>

```

### 4.2.1 Drawing

```

\__draw_backend_literal:n Pass data through using a dedicated interface.
\__draw_backend_literal:x
1311 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1312 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for `\__draw_backend_literal:n`.)

```

\__draw_backend_begin: No special requirements here, so simply set up a drawing scope.
\__draw_backend_end:
1313 \cs_new_protected:Npn \__draw_backend_begin:
1314 { \__draw_backend_scope_begin: }
1315 \cs_new_protected:Npn \__draw_backend_end:
1316 { \__draw_backend_scope_end: }

```

(End definition for `\__draw_backend_begin:` and `\__draw_backend_end:.`)

`\_draw_backend_scope_begin:` Use the backend-level scope mechanisms.

`\_draw_backend_scope_end:`

```

1317 \cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:
1318 \cs_new_eq:NN \_draw_backend_scope_end: \_kernel_backend_scope_end:

```

(End definition for `\_draw_backend_scope_begin:` and `\_draw_backend_scope_end:.`)

`\_draw_backend_moveto:nn` Path creation operations all resolve directly to PDF primitive steps, with only the need

`\_draw_backend_lineto:nn` to convert to bp.

`\_draw_backend_curveto:nnnnnn`

`\_draw_backend_rectangle:nnnn`

```

1319 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1320 {
1321   \_draw_backend_literal:x
1322   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1323 }
1324 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1325 {
1326   \_draw_backend_literal:x
1327   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1328 }
1329 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1330 {
1331   \_draw_backend_literal:x
1332   {
1333     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1334     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1335     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1336     c
1337   }
1338 }
1339 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1340 {
1341   \_draw_backend_literal:x
1342   {
1343     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1344     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1345     re
1346   }
1347 }

```

(End definition for `\_draw_backend_moveto:nn` and others.)

`\_draw_backend_evenodd_rule:` The even-odd rule here can be implemented as a simply switch.

`\_draw_backend_nonzero_rule:`

`\g__draw_draw_eor_bool`

```

1348 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1349 { \bool_gset_true:N \g__draw_draw_eor_bool }
1350 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1351 { \bool_gset_false:N \g__draw_draw_eor_bool }
1352 \bool_new:N \g__draw_draw_eor_bool

```

(End definition for `\_draw_backend_evenodd_rule:`, `\_draw_backend_nonzero_rule:`, and `\g__draw_draw_eor_bool.`)

`\_draw_backend_closepath:` Converting paths to output is again a case of mapping directly to PDF operations.

`\_draw_backend_stroke:`

`\_draw_backend_closestroke:`

`\_draw_backend_fill:`

`\_draw_backend_fillstroke:`

`\_draw_backend_clip:`

`\_draw_backend_discardpath:`

```

1356 { \_draw_backend_literal:n { S } }
1357 \cs_new_protected:Npn \_draw_backend_closestroke:
1358 { \_draw_backend_literal:n { s } }
1359 \cs_new_protected:Npn \_draw_backend_fill:
1360 {
1361   \_draw_backend_literal:x
1362   { f \bool_if:NT \g__draw_draw_eor_bool * }
1363 }
1364 \cs_new_protected:Npn \_draw_backend_fillstroke:
1365 {
1366   \_draw_backend_literal:x
1367   { B \bool_if:NT \g__draw_draw_eor_bool * }
1368 }
1369 \cs_new_protected:Npn \_draw_backend_clip:
1370 {
1371   \_draw_backend_literal:x
1372   { W \bool_if:NT \g__draw_draw_eor_bool * }
1373 }
1374 \cs_new_protected:Npn \_draw_backend_discardpath:
1375 { \_draw_backend_literal:n { n } }

```

(End definition for \\_draw\_backend\_closepath: and others.)

Converting paths to output is again a case of mapping directly to PDF operations.

```

\_draw_backend_dash_pattern:nn
\_draw_backend_dash:n
\_draw_backend_linewidth:n
\_draw_backend_miterlimit:n
\_draw_backend_cap_butt:
\_draw_backend_cap_round:
\_draw_backend_cap_rectangle:
\_draw_backend_join_miter:
\_draw_backend_join_round:
\_draw_backend_join_bevel:
1376 \cs_new_protected:Npn \_draw_backend_dash_pattern:nn #1#2
1377 {
1378   \_draw_backend_literal:x
1379   {
1380     [
1381       \exp_args:Nf \use:n
1382       { \clist_map_function:nN {#1} \_draw_backend_dash:n }
1383     ] ~
1384     \dim_to_decimal_in_bp:n {#2} ~ d
1385   }
1386 }
1387 \cs_new:Npn \_draw_backend_dash:n #1
1388 { ~ \dim_to_decimal_in_bp:n {#1} }
1389 \cs_new_protected:Npn \_draw_backend_linewidth:n #1
1390 {
1391   \_draw_backend_literal:x
1392   { \dim_to_decimal_in_bp:n {#1} ~ w }
1393 }
1394 \cs_new_protected:Npn \_draw_backend_miterlimit:n #1
1395 { \_draw_backend_literal:x { #1 ~ M } }
1396 \cs_new_protected:Npn \_draw_backend_cap_butt:
1397 { \_draw_backend_literal:n { 0 ~ J } }
1398 \cs_new_protected:Npn \_draw_backend_cap_round:
1399 { \_draw_backend_literal:n { 1 ~ J } }
1400 \cs_new_protected:Npn \_draw_backend_cap_rectangle:
1401 { \_draw_backend_literal:n { 2 ~ J } }
1402 \cs_new_protected:Npn \_draw_backend_join_miter:
1403 { \_draw_backend_literal:n { 0 ~ j } }
1404 \cs_new_protected:Npn \_draw_backend_join_round:
1405 { \_draw_backend_literal:n { 1 ~ j } }

```

```

1406 \cs_new_protected:Npn \__draw_backend_join_bevel:
1407 { \__draw_backend_literal:n { 2 ~ j } }

```

(End definition for \\_\_draw\_backend\_dash\_pattern:nn and others.)

```

\__draw_backend_cm:nnnn
\__draw_backend_cm_aux:nnnn

```

Another split here between LuaTeX/pdfTeX and dvipdfmx/X<sub>Y</sub>TeX. In the former, we have a direct method to maintain alignment: the backend can use a matrix itself. For dvipdfmx/X<sub>Y</sub>TeX, we can to decompose the matrix into rotations and a scaling, then use those operations as they are handled by the backend. (There is backend support for matrix operations in dvipdfmx/X<sub>Y</sub>TeX, but as a matched pair so not suitable for the “stand alone” transformation set up here.) The specials used here are from xdvipdfmx originally: they are well-tested, but probably equivalent to the pdf: versions!

```

1408 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1409 {
1410   <*luatex | pdftex>
1411   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1412   </luatex | pdftex>
1413   <*dvipdfmx | xetex>
1414   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1415   \__draw_backend_cm_aux:nnnn
1416   </dvipdfmx | xetex>
1417 }
1418 <*dvipdfmx | xetex>
1419 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1420 {
1421   \__kernel_backend_literal:x
1422   {
1423     x:rotate~
1424     \fp_compare:nNnTF {#1} = \c_zero_fp
1425     { 0 }
1426     { \fp_eval:n { round ( -#1 , 5 ) } }
1427   }
1428   \__kernel_backend_literal:x
1429   {
1430     x:scale~
1431     \fp_eval:n { round ( #2 , 5 ) } ~
1432     \fp_eval:n { round ( #3 , 5 ) }
1433   }
1434   \__kernel_backend_literal:x
1435   {
1436     x:rotate~
1437     \fp_compare:nNnTF {#4} = \c_zero_fp
1438     { 0 }
1439     { \fp_eval:n { round ( -#4 , 5 ) } }
1440   }
1441 }
1442 </dvipdfmx | xetex>

```

(End definition for \\_\_draw\_backend\_cm:nnnn and \\_\_draw\_backend\_cm\_aux:nnnn.)

```

\__draw_backend_cm_decompose:nnnnN
\__draw_backend_cm_decompose_auxi:nnnnN
\__draw_backend_cm_decompose_auxii:nnnnN
\__draw_backend_cm_decompose_auxiii:nnnnN

```

Internally, transformations for drawing are tracked as a matrix. Not all engines provide a way of dealing with this: if we use a raw matrix, the engine loses track of positions (for example for hyperlinks), and this is not desirable. They do, however, allow us to

track rotations and scalings. Luckily, we can decompose any (two-dimensional) matrix into two rotations and a single scaling:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} \cos \beta & \sin \beta \\ -\sin \beta & \cos \beta \end{bmatrix} \begin{bmatrix} w_1 & 0 \\ 0 & w_2 \end{bmatrix} \begin{bmatrix} \cos \gamma & \sin \gamma \\ -\sin \gamma & \cos \gamma \end{bmatrix}$$

The parent matrix can be converted to

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} E & H \\ -H & E \end{bmatrix} + \begin{bmatrix} F & G \\ G & -F \end{bmatrix}$$

From these, we can find that

$$\begin{aligned} \frac{w_1 + w_2}{2} &= \sqrt{E^2 + H^2} \\ \frac{w_1 - w_2}{2} &= \sqrt{F^2 + G^2} \\ \gamma - \beta &= \tan^{-1}(G/F) \\ \gamma + \beta &= \tan^{-1}(H/E) \end{aligned}$$

at which point we just have to do various pieces of re-arrangement to get all of the values. (See J. Blinn, *IEEE Comput. Graph. Appl.*, 1996, **16**, 82–88.) There is one wrinkle: the PostScript (and PDF) way of specifying a transformation matrix exchanges where one would normally expect  $B$  and  $C$  to be.

```

1443 <*dvipdfmx | xetex>
1444 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1445 {
1446   \use:x
1447   {
1448     \__draw_backend_cm_decompose_auxi:nnnnN
1449     { \fp_eval:n { (#1 + #4) / 2 } }
1450     { \fp_eval:n { (#1 - #4) / 2 } }
1451     { \fp_eval:n { (#3 + #2) / 2 } }
1452     { \fp_eval:n { (#3 - #2) / 2 } }
1453   }
1454   #5
1455 }
1456 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1457 {
1458   \use:x
1459   {
1460     \__draw_backend_cm_decompose_auxii:nnnnN
1461     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1462     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1463     { \fp_eval:n { atand ( #3 , #2 ) } }
1464     { \fp_eval:n { atand ( #4 , #1 ) } }
1465   }
1466   #5
1467 }
1468 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1469 {
1470   \use:x
1471   {

```

```

1472 \__draw_backend_cm_decompose_auxiii:nnnnN
1473 { \fp_eval:n { ( #4 - #3 ) / 2 } }
1474 { \fp_eval:n { ( #1 + #2 ) / 2 } }
1475 { \fp_eval:n { ( #1 - #2 ) / 2 } }
1476 { \fp_eval:n { ( #4 + #3 ) / 2 } }
1477 }
1478 #5
1479 }
1480 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1481 {
1482 \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1483 { #5 {#1} {#2} {#3} {#4} }
1484 { #5 {#1} {#3} {#2} {#4} }
1485 }
1486 </dviptdpmx | xetex>

```

(End definition for \\_\_draw\_backend\_cm\_decompose:nnnnN and others.)

\\_\_draw\_backend\_box\_use:Nnnnn

Inserting a T<sub>E</sub>X box transformed to the requested position and using the current matrix is done using a mixture of T<sub>E</sub>X and low-level manipulation. The offset can be handled by T<sub>E</sub>X, so only any rotation/skew/scaling component needs to be done using the matrix operation. As this operation can never be cached, the scope is set directly not using the draw version.

```

1487 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1488 {
1489 \__kernel_backend_scope_begin:
1490 <*luatex | pdftex>
1491 \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1492 </luatex | pdftex>
1493 <*dviptdpmx | xetex>
1494 \__kernel_backend_literal:n
1495 { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1496 </dviptdpmx | xetex>
1497 \hbox_overlap_right:n { \box_use:N #1 }
1498 <*dviptdpmx | xetex>
1499 \__kernel_backend_literal:n { pdf:etrans }
1500 </dviptdpmx | xetex>
1501 \__kernel_backend_scope_end:
1502 }

```

(End definition for \\_\_draw\_backend\_box\_use:Nnnnn.)

```

1503 </dviptdpmx | luatex | pdftex | xetex>

```

### 4.3 dvisvgm backend

```

1504 <*dvisvgm>

```

\\_\_draw\_backend\_literal:n

The same as the more general literal call.

\\_\_draw\_backend\_literal:x

```

1505 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1506 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for \\_\_draw\_backend\_literal:n.)



`\__draw_backend_begin:` A drawing needs to be set up such that the co-ordinate system is translated. That is  
`\__draw_backend_end:` done inside a scope, which as described below

```

1507 \cs_new_protected:Npn \__draw_backend_begin:
1508 {
1509   \__kernel_backend_scope_begin:
1510   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1511 }
1512 \cs_new_eq:NN \__draw_backend_end: \__kernel_backend_scope_end:

```

(End definition for `\__draw_backend_begin:` and `\__draw_backend_end:.`)

`\__draw_backend_moveto:nn` Once again, some work is needed to get path constructs correct. Rather than write the  
`\__draw_backend_lineto:nn` values as they are given, the entire path needs to be collected up before being output  
`\__draw_backend_rectangle:nnnn` in one go. For that we use a dedicated storage routine, which adds spaces as required.  
`\__draw_backend_curveto:nnnnnn` Since paths should be fully expanded there is no need to worry about the internal x-type  
`\__draw_backend_add_to_path:n` expansion.  
`\g__draw_draw_path_tl`

```

1513 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1514 {
1515   \__draw_backend_add_to_path:n
1516   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1517 }
1518 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1519 {
1520   \__draw_backend_add_to_path:n
1521   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1522 }
1523 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1524 {
1525   \__draw_backend_add_to_path:n
1526   {
1527     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1528     h ~ \dim_to_decimal:n {#3} ~
1529     v ~ \dim_to_decimal:n {#4} ~
1530     h ~ \dim_to_decimal:n { -#3 } ~
1531     Z
1532   }
1533 }
1534 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1535 {
1536   \__draw_backend_add_to_path:n
1537   {
1538     C ~
1539     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1540     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1541     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1542   }
1543 }
1544 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1545 {
1546   \tl_gset:Nx \g__draw_draw_path_tl
1547   {
1548     \g__draw_draw_path_tl
1549     \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1550     #1

```

```

1551     }
1552   }
1553   \tl_new:N \g__draw_draw_path_tl

```

(End definition for \\_draw\_backend\_moveto:nn and others.)

```

\_draw_backend_evenodd_rule: The fill rules here have to be handled as scopes.
\_draw_backend_nonzero_rule:
1554 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1555   { \_draw_backend_scope:n { fill-rule="evenodd" } }
1556 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1557   { \_draw_backend_scope:n { fill-rule="nonzero" } }

```

(End definition for \\_draw\_backend\_evenodd\_rule: and \\_draw\_backend\_nonzero\_rule:.)

```

\_draw_backend_path:n Setting fill and stroke effects and doing clipping all has to be done using scopes. This
\_draw_backend_closepath: means setting up the various requirements in a shared auxiliary which deals with the
\_draw_backend_stroke: bits and pieces. Clipping paths are reused for path drawing: not essential but avoids
\_draw_backend_closestroke: constructing them twice. Discarding a path needs a separate function as it's not quite
\_draw_backend_fill: the same.
\_draw_backend_fillstroke:
1558 \cs_new_protected:Npn \_draw_backend_closepath:
\_draw_backend_clip: 1559   { \_draw_backend_add_to_path:n { Z } }
\_draw_backend_discardpath: 1560 \cs_new_protected:Npn \_draw_backend_path:n #1
\g__draw_draw_clip_bool 1561   {
\g__draw_draw_path_int 1562     \bool_if:NTF \g__draw_draw_clip_bool
1563     {
1564       \int_gincr:N \g__draw_clip_path_int
1565       \_draw_backend_literal:x
1566       {
1567         < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1568         { ?nl }
1569         <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1570         < /clipPath > { ? nl }
1571         <
1572         use~xlink:href =
1573         "\c_hash_str l3path \int_use:N \g__draw_path_int " ~
1574         #1
1575         />
1576       }
1577       \_draw_backend_scope:x
1578       {
1579         clip-path =
1580         "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int )"
1581       }
1582     }
1583   {
1584     \_draw_backend_literal:x
1585     { <path ~ d=" \g__draw_draw_path_tl " ~ #1 /> }
1586   }
1587   \tl_gclear:N \g__draw_draw_path_tl
1588   \bool_gset_false:N \g__draw_draw_clip_bool
1589 }
1590 \int_new:N \g__draw_path_int
1591 \cs_new_protected:Npn \_draw_backend_stroke:
1592   { \_draw_backend_path:n { style="fill:none" } }

```

```

1593 \cs_new_protected:Npn \__draw_backend_closestroke:
1594 {
1595   \__draw_backend_closepath:
1596   \__draw_backend_stroke:
1597 }
1598 \cs_new_protected:Npn \__draw_backend_fill:
1599 { \__draw_backend_path:n { style="stroke:none" } }
1600 \cs_new_protected:Npn \__draw_backend_fillstroke:
1601 { \__draw_backend_path:n { } }
1602 \cs_new_protected:Npn \__draw_backend_clip:
1603 { \bool_gset_true:N \g__draw_draw_clip_bool }
1604 \bool_new:N \g__draw_draw_clip_bool
1605 \cs_new_protected:Npn \__draw_backend_discardpath:
1606 {
1607   \bool_if:NT \g__draw_draw_clip_bool
1608   {
1609     \int_gincr:N \g__draw_clip_path_int
1610     \__draw_backend_literal:x
1611     {
1612       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1613       { ?nl }
1614       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1615       < /clipPath >
1616     }
1617     \__draw_backend_scope:x
1618     {
1619       clip-path =
1620       "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int )"
1621     }
1622   }
1623   \tl_gclear:N \g__draw_draw_path_tl
1624   \bool_gset_false:N \g__draw_draw_clip_bool
1625 }

```

(End definition for \\_\_draw\_backend\_path:n and others.)

All of these ideas are properties of scopes in SVG. The only slight complexity is converting the dash array properly (doing any required maths).

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_dash_aux:nn
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1626 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1627 {
1628   \use:x
1629   {
1630     \__draw_backend_dash_aux:nn
1631     { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1632     { \dim_to_decimal:n {#2} }
1633   }
1634 }
1635 \cs_new:Npn \__draw_backend_dash:n #1
1636 { , \dim_to_decimal_in_bp:n {#1} }
1637 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1638 {
1639   \__draw_backend_scope:x
1640   {
1641     stroke-dasharray =

```

```

1642         "
1643         \tl_if_empty:oTF { \use_none:n #1 }
1644         { none }
1645         { \use_none:n #1 }
1646     " ~
1647     stroke-offset=" #2 "
1648 }
1649 }
1650 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1651 { \__draw_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1652 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1653 { \__draw_backend_scope:x { stroke-miterlimit=" #1 " } }
1654 \cs_new_protected:Npn \__draw_backend_cap_but:
1655 { \__draw_backend_scope:n { stroke-linecap="butt" } }
1656 \cs_new_protected:Npn \__draw_backend_cap_round:
1657 { \__draw_backend_scope:n { stroke-linecap="round" } }
1658 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1659 { \__draw_backend_scope:n { stroke-linecap="square" } }
1660 \cs_new_protected:Npn \__draw_backend_join_miter:
1661 { \__draw_backend_scope:n { stroke-linejoin="miter" } }
1662 \cs_new_protected:Npn \__draw_backend_join_round:
1663 { \__draw_backend_scope:n { stroke-linejoin="round" } }
1664 \cs_new_protected:Npn \__draw_backend_join_bevel:
1665 { \__draw_backend_scope:n { stroke-linejoin="bevel" } }

```

(End definition for \\_\_draw\_backend\_dash\_pattern:nn and others.)

\\_\_draw\_backend\_cm:nnnn The four arguments here are floats (the affine matrix), the last two are a displacement vector.

```

1666 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1667 {
1668     \__draw_backend_scope:n
1669     {
1670         transform =
1671         " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1672     }
1673 }

```

(End definition for \\_\_draw\_backend\_cm:nnnn.)

\\_\_draw\_backend\_box\_use:Nnnnn No special savings can be made here: simply displace the box inside a scope. As there is nothing to re-box, just make the box passed of zero size.

```

1674 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1675 {
1676     \__kernel_backend_scope_begin:
1677     \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1678     \__kernel_backend_literal_svg:n
1679     {
1680         < g~
1681         stroke="none"~
1682         transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1683         >
1684     }
1685     \box_set_wd:Nn #1 { Opt }

```

```

1686 \box_set_ht:Nn #1 { Opt }
1687 \box_set_dp:Nn #1 { Opt }
1688 \box_use:N #1
1689 \__kernel_backend_literal_svg:n { </g> }
1690 \__kernel_backend_scope_end:
1691 }

```

(End definition for \\_\_draw\_backend\_box\_use:Nnnnn.)

```
1692 </dvisvgm>
```

```
1693 </package>
```

## 5 l3backend-graphics Implementation

```

1694 <*package>
1695 <@@=graphics>

```

### 5.1 dvips backend

```
1696 <*dvips>
```

\\_graphics\_backend\_getbb\_eps:n Simply use the generic function.

```
1697 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n
```

(End definition for \\_\_graphics\_backend\_getbb\_eps:n.)

\\_graphics\_backend\_include\_eps:n The special syntax is relatively clear here: remember we need PostScript sizes here.

```

1698 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1699 {
1700   \__kernel_backend_literal:x
1701   {
1702     PSfile = #1 \c_space_tl
1703     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1704     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1705     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1706     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1707   }
1708 }

```

(End definition for \\_\_graphics\_backend\_include\_eps:n.)

```
1709 </dvips>
```

### 5.2 LuaTeX and pdfTeX backends

```
1710 <*luatex | pdftex>
```

\l\_graphics\_graphics\_attr\_tl In PDF mode, additional attributes of an graphic (such as page number) are needed both to obtain the bounding box and when inserting the graphic: this occurs as the graphic dictionary approach means they are read as part of the bounding box operation. As such, it is easier to track additional attributes using a dedicated `tl` rather than build up the same data twice.

```
1711 \tl_new:N \l_graphics_graphics_attr_tl
```

(End definition for \l\_graphics\_graphics\_attr\_tl.)

`\_graphics_backend_getbb_jpg:n`  
`\_graphics_backend_getbb_pdf:n`  
`\_graphics_backend_getbb_png:n`  
`\_graphics_backend_getbb_auxi:n`  
`\_graphics_backend_getbb_auxii:n`

Getting the bounding box here requires us to box up the graphic and measure it. To deal with the difference in feature support in bitmap and vector graphics but keeping the common parts, there is a little work to do in terms of auxiliaries. The key here is to notice that we need two forms of the attributes: a “short” set to allow us to track for caching, and the full form to pass to the primitive.

```

1712 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1713 {
1714   \int_zero:N \l_graphics_page_int
1715   \tl_clear:N \l_graphics_pagebox_tl
1716   \tl_set:Nx \l__graphics_graphics_attr_tl
1717   {
1718     \tl_if_empty:NF \l_graphics_decodearray_tl
1719     { :D \l_graphics_decodearray_tl }
1720     \bool_if:NT \l_graphics_interpolate_bool
1721     { :I }
1722   }
1723   \tl_clear:N \l__graphics_graphics_attr_tl
1724   \_graphics_backend_getbb_auxi:n {#1}
1725 }
1726 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1727 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1728 {
1729   \tl_clear:N \l_graphics_decodearray_tl
1730   \bool_set_false:N \l_graphics_interpolate_bool
1731   \tl_set:Nx \l__graphics_graphics_attr_tl
1732   {
1733     : \l_graphics_pagebox_tl
1734     \int_compare:nNnT \l_graphics_page_int > 1
1735     { :P \int_use:N \l_graphics_page_int }
1736   }
1737   \_graphics_backend_getbb_auxi:n {#1}
1738 }
1739 \cs_new_protected:Npn \_graphics_backend_getbb_auxi:n #1
1740 {
1741   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1742   { \_graphics_backend_getbb_auxii:n {#1} }
1743 }

```

Measuring the graphic is done by boxing up: for PDF graphics we could use `\tex_pdfximagebbox:D`, but if doesn’t work for other types. As the box always starts at (0,0) there is no need to worry about the lower-left position.

```

1744 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:n #1
1745 {
1746   \tex_immediate:D \tex_pdfximage:D
1747   \bool_lazy_or:nnT
1748   { \l_graphics_interpolate_bool }
1749   { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1750   {
1751     attr ~
1752     {
1753       \tl_if_empty:NF \l_graphics_decodearray_tl
1754       { /Decode~[ \l_graphics_decodearray_tl ] }
1755       \bool_if:NT \l_graphics_interpolate_bool
1756       { /Interpolate~true }

```

```

1757     }
1758   }
1759   \int_compare:nNnT \l_graphics_page_int > 0
1760     { page ~ \int_use:N \l_graphics_page_int }
1761   \tl_if_empty:NF \l_graphics_pagebox_tl
1762     { \l_graphics_pagebox_tl }
1763   {#1}
1764   \hbox_set:Nn \l__graphics_internal_box
1765     { \tex_pdfrefximage:D \tex_pdflastximage:D }
1766   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1767   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1768   \int_const:cn { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1769     { \tex_the:D \tex_pdflastximage:D }
1770   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1771 }

```

(End definition for `\__graphics_backend_getbb_jpg:n` and others.)

`\__graphics_backend_include_jpg:n`  
`\__graphics_backend_include_pdf:n`  
`\__graphics_backend_include_png:n`

Images are already loaded for the measurement part of the code, so inclusion is straightforward, with only any attributes to worry about. The latter carry through from determination of the bounding box.

```

1772 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1773 {
1774   \tex_pdfrefximage:D
1775   \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1776 }
1777 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1778 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

(End definition for `\__graphics_backend_include_jpg:n`, `\__graphics_backend_include_pdf:n`, and `\__graphics_backend_include_png:n`.)

`\__graphics_backend_getbb_eps:n`  
`\__graphics_backend_getbb_eps:nm`  
`\__graphics_backend_include_eps:n`

EPS graphics may be included in LuaTeX/pdfTeX by conversion to PDF: this requires restricted shell escape. Modelled on the `epstopdf` L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> package, but simplified, conversion takes place here if we have shell access.

```

\l__graphics_backend_dir_str 1779 \sys_if_shell:T
\l__graphics_backend_name_str 1780 {
\l__graphics_backend_ext_str 1781   \str_new:N \l__graphics_backend_dir_str
1782   \str_new:N \l__graphics_backend_name_str
1783   \str_new:N \l__graphics_backend_ext_str
1784   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1785   {
1786     \file_parse_full_name:nNNN {#1}
1787     \l__graphics_backend_dir_str
1788     \l__graphics_backend_name_str
1789     \l__graphics_backend_ext_str
1790     \exp_args:Nx \__graphics_backend_getbb_eps:nn
1791     {
1792       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1793       -converted-to.pdf
1794     }
1795     {#1}
1796   }
1797   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2

```

```

1798     {
1799       \file_compare_timestamp:nNnT {#2} > {#1}
1800       {
1801         \sys_shell_now:n
1802         { repstopdf ~ #2 ~ #1 }
1803       }
1804       \tl_set:Nn \l_graphics_name_tl {#1}
1805       \__graphics_backend_getbb_pdf:n {#1}
1806     }
1807     \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1808     {
1809       \file_parse_full_name:nNNN {#1}
1810       \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1811       \exp_args:Nx \__graphics_backend_include_pdf:n
1812       {
1813         \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1814         -converted-to.pdf
1815       }
1816     }
1817   }

```

(End definition for \\_\_graphics\_backend\_getbb\_eps:n and others.)

```

1818 </luatex | pdftex>

```

### 5.3 dvipdfmx backend

```

1819 <*dvipdfmx | xetex>

```

\\_\_graphics\_backend\_getbb\_eps:n Simply use the generic functions: only for dvipdfmx in the extraction cases.

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
1820 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n
1821 <*dvipdfmx>
1822 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1823 {
1824   \int_zero:N \l_graphics_page_int
1825   \tl_clear:N \l_graphics_pagebox_tl
1826   \graphics_extract_bb:n {#1}
1827 }
1828 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1829 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1830 {
1831   \tl_clear:N \l_graphics_decodearray_tl
1832   \bool_set_false:N \l_graphics_interpolate_bool
1833   \graphics_extract_bb:n {#1}
1834 }
1835 </dvipdfmx>

```

(End definition for \\_\_graphics\_backend\_getbb\_eps:n and others.)

\g\_\_graphics\_track\_int Used to track the object number associated with each graphic.

```

1836 \int_new:N \g__graphics_track_int

```

(End definition for \g\_\_graphics\_track\_int.)



`\_graphics_backend_include_eps:n`  
`\_graphics_backend_include_jpg:n`  
`\_graphics_backend_include_pdf:n`  
`\_graphics_backend_include_png:n`  
`\_graphics_backend_include_auxi:nn`  
`\_graphics_backend_include_auxii:nnn`  
`\_graphics_backend_include_auxiii:nnn`

The special syntax depends on the file type. There is a difference in how PDF graphics are best handled between dvipdfmx and Xe<sub>La</sub>TeX: for the latter it is better to use the primitive route. The relevant code for that is included later in this file.

```

1837 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1838 {
1839   \_kernel_backend_literal:x
1840   {
1841     PSfile = #1 \c_space_tl
1842     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1843     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1844     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1845     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1846   }
1847 }
1848 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1849 { \_graphics_backend_include_auxi:nn {#1} { image } }
1850 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1851 <*/dvipdfmx>
1852 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1853 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1854 </dvipdfmx>

```

Graphic inclusion is set up to use the fact that each image is stored in the PDF as an XObject. This means that we can include repeated images only once and refer to them. To allow that, track the nature of each image: much the same as for the direct PDF mode case.

```

1855 \cs_new_protected:Npn \_graphics_backend_include_auxi:nn #1#2
1856 {
1857   \_graphics_backend_include_auxii:xnn
1858   {
1859     \tl_if_empty:NF \l_graphics_pagebox_tl
1860     { : \l_graphics_pagebox_tl }
1861     \int_compare:nNnT \l_graphics_page_int > 1
1862     { :P \int_use:N \l_graphics_page_int }
1863     \tl_if_empty:NF \l_graphics_decodearray_tl
1864     { :D \l_graphics_decodearray_tl }
1865     \bool_if:NT \l_graphics_interpolate_bool
1866     { :I }
1867   }
1868   {#1} {#2}
1869 }
1870 \cs_new_protected:Npn \_graphics_backend_include_auxii:nnn #1#2#3
1871 {
1872   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1873   {
1874     \_kernel_backend_literal:x
1875     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1876   }
1877   { \_graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1878 }
1879 \cs_generate_variant:Nn \_graphics_backend_include_auxii:nnn { x }

```

Inclusion using the specials is relatively straight-forward, but there is one wrinkle. To get the pagebox correct for PDF graphics in all cases, it is necessary to provide both

that information and the bbox argument: odd things happen otherwise!

```

1880 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1881 {
1882   \int_gincr:N \g__graphics_track_int
1883   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1884   \__kernel_backend_literal:x
1885   {
1886     pdf:#3~
1887     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1888     \int_compare:nNnT \l_graphics_page_int > 1
1889     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1890     \tl_if_empty:NF \l_graphics_pagebox_tl
1891     {
1892       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1893       bbox ~
1894         \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1895         \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1896         \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1897         \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1898     }
1899     (#1)
1900     \bool_lazy_or:nnT
1901     { \l_graphics_interpolate_bool }
1902     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1903     {
1904       <<
1905       \tl_if_empty:NF \l_graphics_decodearray_tl
1906       { /Decode~[ \l_graphics_decodearray_tl ] }
1907       \bool_if:NT \l_graphics_interpolate_bool
1908       { /Interpolate~true> }
1909     }
1910     }
1911   }
1912 }

```

(End definition for \\_\_graphics\_backend\_include\_eps:n and others.)

```

1913 </dvipdfmx | xetex>

```

## 5.4 Xe<sub>La</sub>TeX backend

```

1914 <*xetex>

```

### 5.4.1 Images

For Xe<sub>La</sub>TeX, there are two primitives that allow us to obtain the bounding box without needing `extractbb`. The only complexity is passing the various minor variations to a common core process. The Xe<sub>La</sub>TeX primitive omits the text box from the page box specification, so there is also some “trimming” to do here.

```

1915 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1916 {
1917   \int_zero:N \l_graphics_page_int
1918   \tl_clear:N \l_graphics_pagebox_tl
1919   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1920 }

```

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
\__graphics_backend_getbb_auxi:nN
\__graphics_backend_getbb_auxii:nnN
\__graphics_backend_getbb_auxiii:nNnn
\__graphics_backend_getbb_auxiv:nnNnn
\__graphics_backend_getbb_auxiv:NnnN
\__graphics_backend_getbb_auxv:nNnn
\__graphics_backend_getbb_auxv:NnnN
\__graphics_backend_getbb_pagebox:w

```

```

1921 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1922 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1923 {
1924   \tl_clear:N \l_graphics_decodearray_tl
1925   \bool_set_false:N \l_graphics_interpolate_bool
1926   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1927 }
1928 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
1929 {
1930   \int_compare:nNnTF \l_graphics_page_int > 1
1931     { \__graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
1932     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1933 }
1934 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
1935 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1936 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
1937 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1938 {
1939   \tl_if_empty:NTF \l_graphics_pagebox_tl
1940     { \__graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1941     { \__graphics_backend_getbb_auxv:nNnn }
1942     {#1} #2 {#3} {#4}
1943 }
1944 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
1945 {
1946   \use:x
1947   {
1948     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1949     { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1950   }
1951 }
1952 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1953 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1954 {
1955   \graphics_bb_restore:nF {#1#3}
1956   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1957 }
1958 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1959 {
1960   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1961   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1962   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1963   \graphics_bb_save:n {#1#3}
1964 }
1965 \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End definition for \\_\_graphics\_backend\_getbb\_jpg:n and others.)

\\_\_graphics\_backend\_include\_pdf:n For PDF graphics, properly supporting the pagebox concept in Xe<sub>La</sub>TeX is best done using  
\\_\_graphics\_backend\_include\_bitmap\_quote:w the \tex\_XeTeXpdffile:D primitive. The syntax here is the same as for the graphic  
measurement part, although we know at this stage that there must be some valid setting  
for \l\_graphics\_pagebox\_tl.

```

1966 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1967 {

```

```

1968 \tex_XeTeXpdffile:D
1969 \__graphics_backend_include_pdf_quote:w #1 "#1" \s__graphics_stop \c_space_tl
1970 \int_compare:nNnT \l_graphics_page_int > 0
1971 { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1972 \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1973 }
1974 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \s__graphics_stop
1975 { " #2 " }

```

(End definition for \\_\_graphics\_backend\_include\_pdf:n and \\_\_graphics\_backend\_include\_bitmap\_quote:w.)

```

1976 </xetex>

```

## 5.5 dvisvgm backend

```

1977 <*dvisvgm>

```

\\_\_graphics\_backend\_getbb\_eps:n Simply use the generic function.

```

1978 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n

```

(End definition for \\_\_graphics\_backend\_getbb\_eps:n.)

\\_\_graphics\_backend\_getbb\_png:n These can be included by extracting the bounding box data.

\\_\_graphics\_backend\_getbb\_jpg:n

```

1979 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1980 {
1981   \int_zero:N \l_graphics_page_int
1982   \tl_clear:N \l_graphics_pagebox_tl
1983   \graphics_extract_bb:n {#1}
1984 }
1985 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n

```

(End definition for \\_\_graphics\_backend\_getbb\_png:n and \\_\_graphics\_backend\_getbb\_jpg:n.)

\\_\_graphics\_backend\_getbb\_pdf:n Same as for dvipdfmx: use the generic function

```

1986 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1987 {
1988   \tl_clear:N \l_graphics_decodearray_tl
1989   \bool_set_false:N \l_graphics_interpolate_bool
1990   \graphics_extract_bb:n {#1}
1991 }

```

(End definition for \\_\_graphics\_backend\_getbb\_pdf:n.)

\\_\_graphics\_backend\_include\_eps:n The special syntax is relatively clear here: remember we need PostScript sizes here. (This is the same as the dvips code.)

\\_\_graphics\_backend\_include\_pdf:n

\\_\_graphics\_backend\_include:nn

```

1992 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1993 { __graphics_backend_include:nn { PSfile } {#1} }
1994 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1995 { __graphics_backend_include:nn { pdffile } {#1} }
1996 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
1997 {
1998   \__kernel_backend_literal:x
1999   {
2000     #1 = #2 \c_space_tl
2001     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl

```

```

2002         lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2003         urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2004         ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
2005     }
2006 }

```

(End definition for `\_graphics_backend_include_eps:n`, `\_graphics_backend_include_pdf:n`, and `\_graphics_backend_include:nn`.)

```

\_graphics_backend_include_png:n
\_graphics_backend_include_jpg:n
\_graphics_backend_include_bitmap_quote:w

```

The backend here has built-in support for basic graphic inclusion (see `dvisvgm.def` for a more complex approach, needed if clipping, *etc.*, is covered at the graphic backend level). The only issue is that `#1` must be quote-corrected. The `dvisvgm:img` operation quotes the file name, but if it is already quoted (contains spaces) then we have an issue: we simply strip off any quotes as a result.

```

2007 \cs_new_protected:Npn \_graphics_backend_include_png:n #1
2008 {
2009     \_kernel_backend_literal:x
2010     {
2011         dvisvgm:img~
2012         \dim_to_decimal:n { \l_graphics_ury_dim } ~
2013         \dim_to_decimal:n { \l_graphics_ury_dim } ~
2014         \_graphics_backend_include_bitmap_quote:w #1 " #1 " \s__graphics_stop
2015     }
2016 }
2017 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_png:n
2018 \cs_new:Npn \_graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \s__graphics_stop
2019 { " #2 " }

```

(End definition for `\_graphics_backend_include_png:n`, `\_graphics_backend_include_jpg:n`, and `\_graphics_backend_include_bitmap_quote:w`.)

```

2020 </dvisvgm>
2021 </package>

```

## 6 l3backend-pdf Implementation

```

2022 <*package>
2023 <@@=pdf>

```

Setting up PDF resources is a complex area with only limited documentation in the engine manuals. The following code builds heavily on existing ideas from `hyperref` work by Sebastian Rahtz and Heiko Oberdiek, and significant contributions by Alexander Grahn, in addition to the specific code referenced a various points.

### 6.1 Shared code

A very small number of items that belong at the backend level but which are common to all backends.

```

\l__pdf_internal_box

2024 \box_new:N \l__pdf_internal_box

(End definition for \l__pdf_internal_box.)

```

## 6.2 dvips backend

2025 `<*dvips>`

`\__pdf_backend_pdfmark:n`  
`\__pdf_backend_pdfmark:x`

Used often enough it should be a separate function.

2026 `\cs_new_protected:Npn \__pdf_backend_pdfmark:n #1`  
 2027 `{ \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }`  
 2028 `\cs_generate_variant:Nn \__pdf_backend_pdfmark:n { x }`

(End definition for `\__pdf_backend_pdfmark:n`.)

### 6.2.1 Catalogue entries

`\__pdf_backend_catalog_gput:nn`  
`\__pdf_backend_info_gput:nn`

2029 `\cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2`  
 2030 `{ \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }`  
 2031 `\cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2`  
 2032 `{ \__pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }`

(End definition for `\__pdf_backend_catalog_gput:nn` and `\__pdf_backend_info_gput:nn`.)

### 6.2.2 Objects

`\g__pdf_backend_object_int`  
`\g__pdf_backend_object_prop`

For tracking objects to allow finalisation.

2033 `\int_new:N \g__pdf_backend_object_int`  
 2034 `\prop_new:N \g__pdf_backend_object_prop`

(End definition for `\g__pdf_backend_object_int` and `\g__pdf_backend_object_prop`.)

`\__pdf_backend_object_new:nn`  
`\__pdf_backend_object_ref:n`

Tracking objects is similar to `dvipdfmx`.

2035 `\cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2`  
 2036 `{`  
 2037 `\int_gincr:N \g__pdf_backend_object_int`  
 2038 `\int_const:cn`  
 2039 `{ c__pdf_backend_object_ \tl_to_str:n {#1} _int }`  
 2040 `{ \g__pdf_backend_object_int }`  
 2041 `\prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}`  
 2042 `}`  
 2043 `\cs_new:Npn \__pdf_backend_object_ref:n #1`  
 2044 `{ { pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } } }`

(End definition for `\__pdf_backend_object_new:nn` and `\__pdf_backend_object_ref:n`.)

`\__pdf_backend_object_write:nn`  
`\__pdf_backend_object_write:nx`  
`\__pdf_backend_object_write_array:nn`  
`\__pdf_backend_object_write_dict:nn`  
`\__pdf_backend_object_write_fstream:nn`  
`\__pdf_backend_object_write_stream:nn`  
`\__pdf_backend_object_write_stream:nmm`

This is where we choose the actual type: some work to get things right.

2045 `\cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2`  
 2046 `{`  
 2047 `\__pdf_backend_pdfmark:x`  
 2048 `{`  
 2049 `/_objdef ~ \__pdf_backend_object_ref:n {#1}`  
 2050 `/type`  
 2051 `\str_case_e:nn`  
 2052 `{ \prop_item:Nn \g__pdf_backend_object_prop {#1} }`  
 2053 `{`  
 2054 `{ array }   { /array }`  
 2055 `{ dict }    { /dict }`

```

2056         { fstream } { /stream }
2057         { stream } { /stream }
2058     }
2059     /OBJ
2060 }
2061 \use:c
2062 { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2063 { \__pdf_backend_object_ref:n {#1} } {#2}
2064 }
2065 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2066 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2067 {
2068     \__pdf_backend_pdfmark:x
2069     { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2070 }
2071 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2072 {
2073     \__pdf_backend_pdfmark:x
2074     { #1 << \exp_not:n {#2} >> /PUT }
2075 }
2076 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2077 {
2078     \exp_args:Nx
2079     \__pdf_backend_object_write_fstream:nnn {#1} #2
2080 }
2081 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2082 {
2083     \__kernel_backend_postscript:n
2084     {
2085         SDict ~ begin ~
2086         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2087         mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2088         end
2089     }
2090 }
2091 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2092 {
2093     \exp_args:Nx
2094     \__pdf_backend_object_write_stream:nnn {#1} #2
2095 }
2096 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2097 {
2098     \__kernel_backend_postscript:n
2099     {
2100         mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2101         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2102     }
2103 }

```

(End definition for \\_\_pdf\_backend\_object\_write:nn and others.)

\\_\_pdf\_backend\_object\_now:nn  
\\_\_pdf\_backend\_object\_now:nx

No anonymous objects, so things are done manually.

```

2104 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2105 {

```

```

2106 \int_gincr:N \g__pdf_backend_object_int
2107 \__pdf_backend_pdfmark:x
2108 {
2109   /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2110   /type
2111   \str_case:nn
2112     {#1}
2113     {
2114       { array } { /array }
2115       { dict } { /dict }
2116       { fstream } { /stream }
2117       { stream } { /stream }
2118     }
2119   /OBJ
2120 }
2121 \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2122 { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2123 }
2124 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for \\_\_pdf\_backend\_object\_now:nn.)

\\_\_pdf\_backend\_object\_last: Much like the annotation version.

```

2125 \cs_new:Npn \__pdf_backend_object_last:
2126 { { pdf.obj \int_use:N \g__pdf_backend_object_int } }

```

(End definition for \\_\_pdf\_backend\_object\_last:.)

\\_pdf\_backend\_pageobject\_ref:n Page references are easy in dvips.

```

2127 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2128 { { Page #1 } }

```

(End definition for \\_pdf\_backend\_pageobject\_ref:n.)

### 6.2.3 Annotations

In dvips, annotations have to be constructed manually. As such, we need the object code above for some definitions.

\l\_\_pdf\_backend\_content\_box The content of an annotation.

```

2129 \box_new:N \l__pdf_backend_content_box

```

(End definition for \l\_\_pdf\_backend\_content\_box.)

\l\_\_pdf\_backend\_model\_box For creating model sizing for links.

```

2130 \box_new:N \l__pdf_backend_model_box

```

(End definition for \l\_\_pdf\_backend\_model\_box.)

\g\_\_pdf\_backend\_annotation\_int Needed as objects which are not annotations could be created.

```

2131 \int_new:N \g__pdf_backend_annotation_int

```

(End definition for \g\_\_pdf\_backend\_annotation\_int.)



`\_pdf_backend_annotation:nnnn`

Annotations are objects, but we track them separately. Notably, they are not in the object data lists. Here, to get the co-ordinates of the annotation, we need to have the data collected at the PostScript level. That requires a bit of box trickery (effectively a L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> picture of zero size). Once the data is collected, use it to set up the annotation border.

```
2132 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2133 {
2134   \exp_args:Nf \_pdf_backend_annotation_aux:nnnn
2135   { \dim_eval:n {#1} } {#2} {#3} {#4}
2136 }
2137 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2138 {
2139   \box_move_down:nn {#3}
2140   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2141   \box_move_up:nn {#2}
2142   {
2143     \hbox:n
2144     {
2145       \_kernel_kern:n {#1}
2146       \_kernel_backend_postscript:n { pdf.save.ur }
2147       \_kernel_kern:n { -#1 }
2148     }
2149   }
2150   \int_gincr:N \g__pdf_backend_object_int
2151   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2152   \_pdf_backend_pdfmark:x
2153   {
2154     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2155     pdf.rect
2156     #4 ~
2157     /ANN
2158   }
2159 }
```

(End definition for `\_pdf_backend_annotation:nnnn`.)

`\_pdf_backend_annotation_last:`

Provide the last annotation we created: could get tricky of course if other packages are loaded.

```
2160 \cs_new:Npn \_pdf_backend_annotation_last:
2161 { { pdf.obj \int_use:N \g__pdf_backend_annotation_int } }
```

(End definition for `\_pdf_backend_annotation_last:`.)

`\g__pdf_backend_link_int`

To track annotations which are links.

```
2162 \int_new:N \g__pdf_backend_link_int
```

(End definition for `\g__pdf_backend_link_int`.)

`\g__pdf_backend_link_dict_tl`

To pass information to the end-of-link function.

```
2163 \tl_new:N \g__pdf_backend_link_dict_tl
```

(End definition for `\g__pdf_backend_link_dict_tl`.)

`\g__pdf_backend_link_sf_int`

Needed to save/restore space factor, which is needed to deal with the face we need a box.

```
2164 \int_new:N \g__pdf_backend_link_sf_int
```

(End definition for `\g__pdf_backend_link_sf_int`.)

`\g__pdf_backend_link_math_bool` Needed to save/restore math mode.

2165 `\bool_new:N \g__pdf_backend_link_math_bool`

(End definition for `\g__pdf_backend_link_math_bool`.)

`\g__pdf_backend_link_bool` Track link formation: we cannot nest at all.

2166 `\bool_new:N \g__pdf_backend_link_bool`

(End definition for `\g__pdf_backend_link_bool`.)

`\l__pdf_breaklink_pdfmark_tl` Swappable content for link breaking.

2167 `\tl_new:N \l__pdf_breaklink_pdfmark_tl`

2168 `\tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdfmark }`

(End definition for `\l__pdf_breaklink_pdfmark_tl`.)

`\__pdf_breaklink_postscript:n` To allow dropping material unless link breaking is active.

2169 `\cs_new_protected:Npn \__pdf_breaklink_postscript:n #1 { }`

(End definition for `\__pdf_breaklink_postscript:n`.)

`\__pdf_breaklink_usebox:N` Swappable box unpacking or use.

2170 `\cs_new_eq:NN \__pdf_breaklink_usebox:N \box_use:N`

(End definition for `\__pdf_breaklink_usebox:N`.)

`\__pdf_backend_link_begin_goto:nnw`

`\__pdf_backend_link_begin_user:nnw`

`\__pdf_backend_link:nw`

`\__pdf_backend_link_aux:nw`

`\__pdf_backend_link_end:`

`\__pdf_backend_link_end_aux:`

`\__pdf_backend_link_minima:`

`\__pdf_backend_link_outerbox:n`

`\__pdf_backend_link_sf_save:`

`\__pdf_backend_link_sf_restore:`

`pdf.linkdp.pad`

`pdf.linkht.pad`

`pdf.llx`

`pdf.lly`

`pdf.ury`

`pdf.link.dict`

`pdf.outerbox`

`pdf.baselineskip`

Links are crated like annotations but with dedicated code to allow for adjusting the size of the rectangle. In contrast to `hyperref`, we grab the link content as a box which can then unbox: this allows the same interface as for `pdfTeX`.

Taking the idea of `evenboxes` from `hypdvips`, we implement a minimum box height and depth for link placement. This means that “underlining” with a hyperlink will generally give an even appearance. However, to ensure that the full content is always above the link border, we do not allow this to be negative (contrast `hypdvips` approach). The result should be similar to `pdfTeX` in the vast majority of foreseeable cases.

The object number for a link is saved separately from the rest of the dictionary as this allows us to insert it just once, at either an unbroken link or only in the first line of a broken one. That makes the code clearer but also avoids a low-level PostScript error with the code as taken from `hypdvips`.

Getting the outer dimensions of the text area may be better using a two-pass approach and `\tex_savepos:D`. That plus format mode are still to re-examine.

2171 `\cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2`

2172 `{ \__pdf_backend_link_begin:nw { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }`

2173 `\cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2`

2174 `{ \__pdf_backend_link_begin:nw {#1#2} }`

2175 `\cs_new_protected:Npn \__pdf_backend_link_begin:nw #1`

2176 `{`

2177 `\bool_if:NF \g__pdf_backend_link_bool`

2178 `{ \__pdf_backend_link_begin_aux:nw {#1} }`

2179 `}`

2180 `\cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1`

2181 `{`

2182 `\bool_gset_true:N \g__pdf_backend_link_bool`

```

2183 \__kernel_backend_postscript:n
2184 { /pdf.link.dict ( #1 ) def }
2185 \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2186 \__pdf_backend_link_sf_save:
2187 \mode_if_math:TF
2188 { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2189 { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2190 \hbox_set:Nw \l__pdf_backend_content_box
2191 \__pdf_backend_link_sf_restore:
2192 \bool_if:NT \g__pdf_backend_link_math_bool
2193 { \c_math_toggle_token }
2194 }
2195 \cs_new_protected:Npn \__pdf_backend_link_end:
2196 {
2197   \bool_if:NT \g__pdf_backend_link_bool
2198   { \__pdf_backend_link_end_aux: }
2199 }
2200 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2201 {
2202   \bool_if:NT \g__pdf_backend_link_math_bool
2203   { \c_math_toggle_token }
2204   \__pdf_backend_link_sf_save:
2205   \hbox_set_end:
2206   \__pdf_backend_link_minima:
2207   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2208   \exp_args:Nx \__pdf_backend_link_outerbox:n
2209   {
2210     \int_if_odd:nTF { \value { page } }
2211     { \oddsidemargin }
2212     { \evensidemargin }
2213   }
2214   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2215   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2216   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2217   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2218   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2219   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2220   {
2221     \hbox:n
2222     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2223   }
2224   \int_gincr:N \g__pdf_backend_object_int
2225   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2226   \__kernel_backend_postscript:x
2227   {
2228     mark
2229     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2230     \g__pdf_backend_link_dict_tl \c_space_tl
2231     pdf.rect
2232     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2233   }
2234   \__pdf_backend_link_sf_restore:
2235   \bool_gset_false:N \g__pdf_backend_link_bool
2236 }

```

```

2237 \cs_new_protected:Npn \__pdf_backend_link_minima:
2238 {
2239   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2240   \__kernel_backend_postscript:x
2241   {
2242     /pdf.linkdp.pad ~
2243     \dim_to_decimal:n
2244     {
2245       \dim_max:nn
2246       {
2247         \box_dp:N \l__pdf_backend_model_box
2248         - \box_dp:N \l__pdf_backend_content_box
2249       }
2250       { Opt }
2251     } ~
2252     pdf.pt.dvi ~ def
2253   /pdf.linkht.pad ~
2254   \dim_to_decimal:n
2255   {
2256     \dim_max:nn
2257     {
2258       \box_ht:N \l__pdf_backend_model_box
2259       - \box_ht:N \l__pdf_backend_content_box
2260     }
2261     { Opt }
2262   } ~
2263   pdf.pt.dvi ~ def
2264 }
2265 }
2266 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2267 {
2268   \__kernel_backend_postscript:x
2269   {
2270     /pdf.outerbox
2271     [
2272       \dim_to_decimal:n {#1} ~
2273       \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2274       \dim_to_decimal:n { #1 + \textwidth } ~
2275       \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2276     ]
2277     [ exch { pdf.pt.dvi } forall ] def
2278   /pdf.baselineskip ~
2279   \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2280   { pdf.pt.dvi ~ def }
2281   { pop ~ pop }
2282   ifelse
2283 }
2284 }
2285 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2286 {
2287   \int_gset:Nn \g__pdf_backend_link_sf_int
2288   {
2289     \mode_if_horizontal:TF
2290     { \tex_spacefactor:D }

```

```

2291         { 0 }
2292     }
2293 }
2294 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2295 {
2296     \mode_if_horizontal:T
2297     {
2298         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2299         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2300     }
2301 }

```

(End definition for \\_\_pdf\_backend\_link\_begin\_goto:nw and others. These functions are documented on page ??.)

**\@makecol@hook** Hooks to allow link breaking: something will be needed in format mode at some stage. At present this code is disabled as there is an open question about the name of the hook: to be resolved at the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> end.

```

2302 \use_none:n
2303 {
2304     \cs_if_exist:NT \@makecol@hook
2305     {
2306         \tl_put_right:Nn \@makecol@hook
2307         {
2308             \box_if_empty:NF \@cclv
2309             {
2310                 \vbox_set:Nn \@cclv
2311                 {
2312                     \__kernel_backend_postscript:n
2313                     {
2314                         pdf.globaldict /pdf.brokenlink.rect ~ known
2315                         { pdf.bordertracking.continue }
2316                     if
2317                 }
2318                 \vbox_unpack_drop:N \@cclv
2319                 \__kernel_backend_postscript:n
2320                 { pdf.bordertracking.endpage }
2321             }
2322         }
2323     }
2324     \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2325     \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2326     \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2327 }
2328 }

```

(End definition for \@makecol@hook. This function is documented on page ??.)

**\\_\_pdf\_backend\_link\_last:** The same as annotations, but with a custom integer.

```

2329 \cs_new:Npn \__pdf_backend_link_last:
2330 { { pdf.obj \int_use:N \g__pdf_backend_link_int } }

```

(End definition for \\_\_pdf\_backend\_link\_last:.)

`\_pdf_backend_link_margin:n` Convert to big points and pass to PostScript.

```

2331 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2332 {
2333   \_kernel_backend_postscript:x
2334   {
2335     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2336   }
2337 }

```

(End definition for `\_pdf_backend_link_margin:n`.)

`\_pdf_backend_destination:nn` Here, we need to turn the zoom into a scale. We also need to know where the current  
`\_pdf_backend_destination:nmmn` anchor point actually is: worked out in PostScript. For the rectangle version, we have a  
`\_pdf_backend_destination_aux:nmmn` bit more PostScript: we need two points. `fitr` without rule spec doesn't work, so it falls  
back to `/Fit` here.

```

2338 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2339 {
2340   \_kernel_backend_postscript:n { pdf.dest.anchor }
2341   \_pdf_backend_pdfmark:x
2342   {
2343     /View
2344     [
2345       \str_case:nnF {#2}
2346       {
2347         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2348         { fit } { /Fit }
2349         { fitb } { /FitB }
2350         { fitbh } { /FitBH ~ pdf.dest.y }
2351         { fitbv } { /FitBV ~ pdf.dest.x }
2352         { fith } { /FitH ~ pdf.dest.y }
2353         { fitv } { /FitV ~ pdf.dest.x }
2354         { fitr } { /Fit }
2355       }
2356       {
2357         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2358       }
2359     ]
2360     /Dest ( \exp_not:n {#1} ) cvn
2361     /DEST
2362   }
2363 }
2364 \cs_new_protected:Npn \_pdf_backend_destination:nmmn #1#2#3#4
2365 {
2366   \exp_args:Ne \_pdf_backend_destination_aux:nmmn
2367   { \dim_eval:n {#2} } {#1} {#3} {#4}
2368 }
2369 \cs_new_protected:Npn \_pdf_backend_destination_aux:nmmn #1#2#3#4
2370 {
2371   \vbox_to_zero:n
2372   {
2373     \_kernel_kern:n {#4}
2374     \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } }
2375     \tex_vss:D
2376   }

```

```

2377 \__kernel_kern:n {#1}
2378 \vbox_to_zero:n
2379 {
2380   \__kernel_kern:n { -#3 }
2381   \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2382   \tex_vss:D
2383 }
2384 \__kernel_kern:n { -#1 }
2385 \__pdf_backend_pdfmark:n
2386 {
2387   /View
2388   [
2389     /FitR ~
2390     pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2391     pdf.urx ~ pdf.ury ~ pdf.dest2device
2392   ]
2393   /Dest ( #2 ) cvn
2394   /DEST
2395 }
2396 }

```

(End definition for \\_\_pdf\_backend\_destination:nn, \\_\_pdf\_backend\_destination:nnnn, and \\_\_pdf\_backend\_destination\_aux:nnnn.)

#### 6.2.4 Structure

\\_\_pdf\_backend\_compresslevel:n Doable for the usual ps2pdf method.

```

\__pdf_backend_compress_objects:n
2397 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2398 {
2399   \int_compare:nNnT {#1} = 0
2400   {
2401     \__kernel_backend_literal_postscript:n
2402     {
2403       /setdistillerparams ~ where
2404       { pop << /CompressPages ~ false >> setdistillerparams }
2405       if
2406     }
2407   }
2408 }
2409 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2410 {
2411   \bool_if:nF {#1}
2412   {
2413     \__kernel_backend_literal_postscript:n
2414     {
2415       /setdistillerparams ~ where
2416       { pop << /CompressStreams ~ false >> setdistillerparams }
2417       if
2418     }
2419   }
2420 }

```

(End definition for \\_\_pdf\_backend\_compresslevel:n and \\_\_pdf\_backend\_compress\_objects:n.)

```

\_pdf_backend_version_major_gset:n Data not available!
\_pdf_backend_version_minor_gset:n
2421 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2422 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1 { }

(End definition for \__pdf_backend_version_major_gset:n and \__pdf_backend_version_minor_gset:n.)

```

```

\_pdf_backend_version_major: Data not available!
\_pdf_backend_version_minor:
2423 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2424 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

(End definition for \__pdf_backend_version_major: and \__pdf_backend_version_minor:.)

```

### 6.2.5 Marked content

```

\_pdf_backend_bdc:nn Simple wrappers.
\_pdf_backend_emc:
2425 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2426 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2427 \cs_new_protected:Npn \__pdf_backend_emc:
2428 { \__pdf_backend_pdfmark:n { /EMC } }

(End definition for \__pdf_backend_bdc:nn and \__pdf_backend_emc:.)
2429 </dvips>

```

## 6.3 LuaTeX and pdfTeX backend

```
2430 <*luatex | pdftex>
```

### 6.3.1 Annotations

\\_pdf\_backend\_annotation:nnnn Simply pass the raw data through, just dealing with evaluation of dimensions.

```

2431 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2432 {
2433 <*luatex>
2434 \tex_pdfextension:D annot ~
2435 </luatex>
2436 <*pdftex>
2437 \tex_pdfannot:D
2438 </pdftex>
2439 width ~ \dim_eval:n {#1} ~
2440 height ~ \dim_eval:n {#2} ~
2441 depth ~ \dim_eval:n {#3} ~
2442 {#4}
2443 }

```

(End definition for \\_\_pdf\_backend\_annotation:nnnn.)

\\_pdf\_backend\_annotation\_last: A tiny amount of extra data gets added here; we use x-type expansion to get the space in the right place and form. The “extra” space in the LuaTeX version is *required* as it is consumed in finding the end of the keyword.

```

2444 \cs_new:Npx \__pdf_backend_annotation_last:
2445 {
2446 \exp_not:N \int_value:w
2447 <*luatex>
2448 \exp_not:N \tex_pdffeedback:D lastannot ~

```



```

2449 </luatex>
2450 <*pdftex>
2451     \exp_not:N \tex_pdflastannot:D
2452 </pdftex>
2453     \c_space_tl 0 ~ R
2454 }

(End definition for \_pdf_backend_annotation_last:.)

```

\\_pdf\_backend\_link\_begin\_goto:nnw Links are all created using the same internals.

```

\_pdf_backend_link_begin_user:nnw
\_pdf_backend_link_begin:nnnw
\_pdf_backend_link_end:
2455 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2456 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2457 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2458 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2459 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2460 {
2461 <*luatex>
2462     \tex_pdfextension:D startlink ~
2463 </luatex>
2464 <*pdftex>
2465     \tex_pdfstartlink:D
2466 </pdftex>
2467     attr {#1}
2468     #2 {#3}
2469 }
2470 \cs_new_protected:Npn \_pdf_backend_link_end:
2471 {
2472 <*luatex>
2473     \tex_pdfextension:D endlink \scan_stop:
2474 </luatex>
2475 <*pdftex>
2476     \tex_pdfendlink:D
2477 </pdftex>
2478 }

(End definition for \_pdf_backend_link_begin_goto:nnw and others.)

```

\\_pdf\_backend\_link\_last: Formatted for direct use.

```

2479 \cs_new:Npx \_pdf_backend_link_last:
2480 {
2481     \exp_not:N \int_value:w
2482 <*luatex>
2483     \exp_not:N \tex_pdffeedback:D lastlink ~
2484 </luatex>
2485 <*pdftex>
2486     \exp_not:N \tex_pdflastlink:D
2487 </pdftex>
2488     \c_space_tl 0 ~ R
2489 }

(End definition for \_pdf_backend_link_last:.)

```

\\_pdf\_backend\_link\_margin:n A simple task: pass the data to the primitive.

```

2490 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2491 {

```

```

2492 <*luatex>
2493     \tex_pdfvariable:D linkmargin
2494 </luatex>
2495 <*pdftex>
2496     \tex_pdflinkmargin:D
2497 </pdftex>
2498     \dim_eval:n {#1} \scan_stop:
2499 }

```

(End definition for \\_pdf\_backend\_link\_margin:n.)

\\_pdf\_backend\_destination:nn  
\\_pdf\_backend\_destination:nnnn

A simple task: pass the data to the primitive. The \scan\_stop: deals with the danger of an unterminated keyword. The zoom given here is a percentage, but we need to pass it as *per mille*. The rectangle version is also easy as everything is build in.

```

2500 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2501 {
2502 <*luatex>
2503     \tex_pdfextension:D dest ~
2504 </luatex>
2505 <*pdftex>
2506     \tex_pdfdest:D
2507 </pdftex>
2508     name {#1}
2509     \str_case:nnF {#2}
2510     {
2511         { xyz } { xyz }
2512         { fit } { fit }
2513         { fitb } { fitb }
2514         { fitbh } { fitbh }
2515         { fitbv } { fitbv }
2516         { fith } { fith }
2517         { fitv } { fitv }
2518         { fitr } { fitr }
2519     }
2520     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2521     \scan_stop:
2522 }
2523 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2524 {
2525 <*luatex>
2526     \tex_pdfextension:D dest ~
2527 </luatex>
2528 <*pdftex>
2529     \tex_pdfdest:D
2530 </pdftex>
2531     name {#1}
2532     fitr ~
2533     width \dim_eval:n {#2} ~
2534     height \dim_eval:n {#3} ~
2535     depth \dim_eval:n {#4} \scan_stop:
2536 }

```

(End definition for \\_pdf\_backend\_destination:nn and \\_pdf\_backend\_destination:nnnn.)

### 6.3.2 Catalogue entries

```

\__pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
2537 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2538 {
2539   \*luatex
2540     \tex_pdfextension:D catalog
2541   \*pdftex
2542     \tex_pdfcatalog:D
2543   \*pdftex
2544     { / #1 ~ #2 }
2545   }
2546 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2547 {
2548   \*luatex
2549     \tex_pdfextension:D info
2550   \*pdftex
2551     \tex_pdfinfo:D
2552   \*pdftex
2553     { / #1 ~ #2 }
2554   }
2555 }
2556 }

(End definition for \__pdf_backend_catalog_gput:nn and \__pdf_backend_info_gput:nn.)

```

### 6.3.3 Objects

```

\g__pdf_backend_object_prop For tracking objects to allow finalisation.
2557 \prop_new:N \g__pdf_backend_object_prop

(End definition for \g__pdf_backend_object_prop.)

\__pdf_backend_object_new:nn Declaring objects means reserving at the PDF level plus starting tracking.
\__pdf_backend_object_ref:n
2558 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2559 {
2560   \*luatex
2561     \tex_pdfextension:D obj ~
2562   \*pdftex
2563     \tex_pdfobj:D
2564     reserveobjnum ~
2565     \int_const:cn
2566     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2567   \*luatex
2568     { \tex_pdffeedback:D lastobj }
2569   \*pdftex
2570     { \tex_pdflastobj:D }
2571   \*pdftex
2572     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2573   }
2574 \cs_new:Npn \__pdf_backend_object_ref:n #1
2575 { \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }
2576 }

```

(End definition for `\_pdf_backend_object_new:nn` and `\_pdf_backend_object_ref:n`.)

`\_pdf_backend_object_write:nn` Writing the data needs a little information about the structure of the object.

```

\__pdf_backend_object_write:nx 2579 \cs_new_protected:Npn \_pdf_backend_object_write:nn #1#2
\__pdf_exp_not_i:nn           2580 {
\__pdf_exp_not_ii:nn          2581 <*luatex>
                                2582 \tex_immediate:D \tex_pdfextension:D obj ~
                                2583 </luatex>
                                2584 <*pdfTeX>
                                2585 \tex_immediate:D \tex_pdfobj:D
                                2586 </pdfTeX>
                                2587 useobjnum ~
                                2588 \int_use:c
                                2589 { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
                                2590 \str_case_e:nn
                                2591 { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
                                2592 {
                                2593   { array } { { [ ~ \exp_not:n {#2} ~ ] } }
                                2594   { dict } { { << ~ \exp_not:n {#2} ~ >> } }
                                2595   { fstream }
                                2596   {
                                2597     stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
                                2598     file ~ { \_pdf_exp_not_ii:nn #2 }
                                2599   }
                                2600   { stream }
                                2601   {
                                2602     stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
                                2603     { \_pdf_exp_not_ii:nn #2 }
                                2604   }
                                2605 }
                                2606 }
                                2607 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
                                2608 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
                                2609 \cs_new:Npn \_pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

```

(End definition for `\_pdf_backend_object_write:nn`, `\_pdf_exp_not_i:nn`, and `\_pdf_exp_not_ii:nn`.)

`\_pdf_backend_object_now:nn` Much like writing, but direct creation.

```

\__pdf_backend_object_now:nx 2610 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
                                2611 {
                                2612 <*luatex>
                                2613 \tex_immediate:D \tex_pdfextension:D obj ~
                                2614 </luatex>
                                2615 <*pdfTeX>
                                2616 \tex_immediate:D \tex_pdfobj:D
                                2617 </pdfTeX>
                                2618 \str_case:nn
                                2619 {#1}
                                2620 {
                                2621   { array } { { [ ~ \exp_not:n {#2} ~ ] } }
                                2622   { dict } { { << ~ \exp_not:n {#2} ~ >> } }
                                2623   { fstream }
                                2624   {

```

```

2625         stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2626         file ~ { \_pdf_exp_not_ii:nn #2 }
2627     }
2628 { stream }
2629 {
2630     stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2631     { \_pdf_exp_not_ii:nn #2 }
2632 }
2633 }
2634 }
2635 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }

```

(End definition for \\_pdf\_backend\_object\_now:nn.)

\\_pdf\_backend\_object\_last: Much like annotation.

```

2636 \cs_new:Npx \_pdf_backend_object_last:
2637 {
2638     \exp_not:N \int_value:w
2639 <*luatex>
2640     \exp_not:N \tex_pdffeedback:D lastobj ~
2641 </luatex>
2642 <*pdftex>
2643     \exp_not:N \tex_pdflastobj:D
2644 </pdftex>
2645     \c_space_tl 0 ~ R
2646 }

```

(End definition for \\_pdf\_backend\_object\_last:.)

\\_pdf\_backend\_pageobject\_ref:n The usual wrapper situation; the three spaces here are essential.

```

2647 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2648 {
2649     \exp_not:N \int_value:w
2650 <*luatex>
2651     \exp_not:N \tex_pdffeedback:D pageref
2652 </luatex>
2653 <*pdftex>
2654     \exp_not:N \tex_pdfpageref:D
2655 </pdftex>
2656     \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2657 }

```

(End definition for \\_pdf\_backend\_pageobject\_ref:n.)

### 6.3.4 Structure

\\_pdf\_backend\_compresslevel:n Simply pass data to the engine.

```

\_pdf_backend_compress_objects:n 2658 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
\_pdf_backend_objcompresslevel:n 2659 {
2660     \tex_global:D
2661 <*luatex>
2662     \tex_pdfvariable:D compresslevel
2663 </luatex>
2664 <*pdftex>
2665     \tex_pdfcompresslevel:D

```

```

2666 </pdftex>
2667     \int_value:w \int_eval:n {#1} \scan_stop:
2668 }
2669 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2670 {
2671     \bool_if:nTF {#1}
2672     { \__pdf_backend_objcompresslevel:n { 2 } }
2673     { \__pdf_backend_objcompresslevel:n { 0 } }
2674 }
2675 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2676 {
2677     \tex_global:D
2678     <*luatex>
2679     \tex_pdfvariable:D objcompresslevel
2680 </luatex>
2681 <*pdftex>
2682     \tex_pdfobjcompresslevel:D
2683 </pdftex>
2684     #1 \scan_stop:
2685 }

```

(End definition for \\_\_pdf\_backend\_compresslevel:n, \\_\_pdf\_backend\_compress\_objects:n, and \\_\_pdf\_backend\_objcompresslevel:n.)

\\_\_pdf\_backend\_version\_major\_gset:n  
\\_\_pdf\_backend\_version\_minor\_gset:n

The availability of the primitive is not universal, so we have to test at load time.

```

2686 \cs_new_protected:Npx \__pdf_backend_version_major_gset:n #1
2687 {
2688 <*luatex>
2689     \int_compare:nNnT \tex_luatexversion:D > { 106 }
2690     {
2691         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2692         \exp_not:N \int_eval:n {#1} \scan_stop:
2693     }
2694 </luatex>
2695 <*pdftex>
2696     \cs_if_exist:NT \tex_pdfmajorversion:D
2697     {
2698         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2699         \exp_not:N \int_eval:n {#1} \scan_stop:
2700     }
2701 </pdftex>
2702 }
2703 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2704 {
2705     \tex_global:D
2706 <*luatex>
2707     \tex_pdfvariable:D minorversion
2708 </luatex>
2709 <*pdftex>
2710     \tex_pdfminorversion:D
2711 </pdftex>
2712     \int_eval:n {#1} \scan_stop:
2713 }

```

(End definition for \\_\_pdf\_backend\_version\_major\_gset:n and \\_\_pdf\_backend\_version\_minor\_gset:n.)

`\_pdf_backend_version_major:` As above.

```

\__pdf_backend_version_minor:
2714 \cs_new:Npx \__pdf_backend_version_major:
2715 {
2716 <*luatex>
2717   \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2718   { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2719   { 1 }
2720 </luatex>
2721 <*pdfTeX>
2722   \cs_if_exist:NTF \tex_pdfmajorversion:D
2723   { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2724   { 1 }
2725 </pdfTeX>
2726 }
2727 \cs_new:Npn \__pdf_backend_version_minor:
2728 {
2729   \tex_the:D
2730 <*luatex>
2731   \tex_pdfvariable:D minorversion
2732 </luatex>
2733 <*pdfTeX>
2734   \tex_pdfminorversion:D
2735 </pdfTeX>
2736 }

(End definition for \_pdf_backend_version_major: and \_pdf_backend_version_minor:.)

```

### 6.3.5 Marked content

`\_pdf_backend_bdc:nn` Simple wrappers. May need refinement: see <https://chat.stackexchange.com/transcript/message/49970158#49970158>.

```

\__pdf_backend_emc:
2737 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2738 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2739 \cs_new_protected:Npn \_pdf_backend_emc:
2740 { \__kernel_backend_literal_page:n { EMC } }

(End definition for \_pdf_backend_bdc:nn and \_pdf_backend_emc:.)
2741 </luatex | pdfTeX>

```

## 6.4 dvipdfmx backend

```

2742 <*dvipdfmx | xetex>

\_pdf_backend:n A generic function for the backend PDF specials: used where we can.
\_pdf_backend:x
2743 \cs_new_protected:Npx \_pdf_backend:n #1
2744 { \__kernel_backend_literal:n { pdf: #1 } }
2745 \cs_generate_variant:Nn \_pdf_backend:n { x }

(End definition for \_pdf_backend:n.)

```

### 6.4.1 Catalogue entries

```

    \_pdf_backend_catalog_gput:nn
\_pdf_backend_info_gput:nn
2746 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2747 { \_pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2748 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2749 { \_pdf_backend:n { docinfo << /#1 ~ #2 >> } }

(End definition for \_pdf_backend_catalog_gput:nn and \_pdf_backend_info_gput:nn.)

```

### 6.4.2 Objects

For tracking objects to allow finalisation.

```

\_pdf_backend_object_int
\_pdf_backend_object_prop
2750 \int_new:N \g__pdf_backend_object_int
2751 \prop_new:N \g__pdf_backend_object_prop

(End definition for \g__pdf_backend_object_int and \g__pdf_backend_object_prop.)

```

Objects are tracked at the macro level, but we don't have to do anything at this stage.

```

\_pdf_backend_object_new:nn
\_pdf_backend_object_ref:n
2752 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2
2753 {
2754   \int_gincr:N \g__pdf_backend_object_int
2755   \int_const:cn
2756     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2757     { \g__pdf_backend_object_int }
2758   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2759 }
2760 \cs_new:Npn \_pdf_backend_object_ref:n #1
2761 { @pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } }

(End definition for \_pdf_backend_object_new:nn and \_pdf_backend_object_ref:n.)

```

This is where we choose the actual type.

```

\_pdf_backend_object_write:nn
\_pdf_backend_object_write:nx
\_pdf_backend_object_write:nnn
\_pdf_backend_object_write_array:nn
\_pdf_backend_object_write_dict:nn
\_pdf_backend_object_write_fstream:nn
\_pdf_backend_object_write_stream:nn
\_pdf_backend_object_write_stream:nnnn
2762 \cs_new_protected:Npn \_pdf_backend_object_write:nn #1#2
2763 {
2764   \exp_args:Nx \_pdf_backend_object_write:nnn
2765   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2766 }
2767 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
2768 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
2769 {
2770   \use:c { __pdf_backend_object_write_ #1 :nn }
2771   { \_pdf_backend_object_ref:n {#2} } {#3}
2772 }
2773 \cs_new_protected:Npn \_pdf_backend_object_write_array:nn #1#2
2774 {
2775   \_pdf_backend:x
2776     { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2777 }
2778 \cs_new_protected:Npn \_pdf_backend_object_write_dict:nn #1#2
2779 {
2780   \_pdf_backend:x
2781     { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2782 }
2783 \cs_new_protected:Npn \_pdf_backend_object_write_fstream:nn #1#2

```



```

2784 { \_pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2785 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nn #1#2
2786 { \_pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2787 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nnnn #1#2#3#4
2788 {
2789   \_pdf_backend:x
2790   {
2791     #1 stream ~ #2 ~
2792     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2793   }
2794 }

```

(End definition for \\_pdf\_backend\_object\_write:nn and others.)

\\_pdf\_backend\_object\_now:nn No anonymous objects with dvipdfmx so we have to give an object name.  
\\_pdf\_backend\_object\_now:nx

```

2795 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2796 {
2797   \int_gincr:N \g__pdf_backend_object_int
2798   \exp_args:Nnx \use:c { \_pdf_backend_object_write_ #1 :nn }
2799   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2800   {#2}
2801 }
2802 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }

```

(End definition for \\_pdf\_backend\_object\_now:nn.)

\\_pdf\_backend\_object\_last:

```

2803 \cs_new:Npn \_pdf_backend_object_last:
2804 { @pdf.obj \int_use:N \g__pdf_backend_object_int }

```

(End definition for \\_pdf\_backend\_object\_last:.)

\\_pdf\_backend\_pageobject\_ref:n Page references are easy in dvipdfmx/X<sub>Y</sub>TeX.

```

2805 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2806 { @page #1 }

```

(End definition for \\_pdf\_backend\_pageobject\_ref:n.)

### 6.4.3 Annotations

\g\_\_pdf\_backend\_annotation\_int Needed as objects which are not annotations could be created.

```

2807 \int_new:N \g__pdf_backend_annotation_int

```

(End definition for \g\_\_pdf\_backend\_annotation\_int.)

\\_pdf\_backend\_annotation:nnnn Simply pass the raw data through, just dealing with evaluation of dimensions.

```

2808 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2809 {
2810   \int_gincr:N \g__pdf_backend_object_int
2811   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2812   \_pdf_backend:x
2813   {
2814     ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
2815     width ~ \dim_eval:n {#1} ~
2816     height ~ \dim_eval:n {#2} ~

```

```

2817         depth ~ \dim_eval:n {#3} ~
2818         <</Type/Annot #4 >>
2819     }
2820 }

```

(End definition for \\_pdf\_backend\_annotation:nnnn.)

\\_pdf\_backend\_annotation\_last:

```

2821 \cs_new:Npn \_pdf_backend_annotation_last:
2822 { @pdf.obj \int_use:N \g_pdf_backend_annotation_int }

```

(End definition for \\_pdf\_backend\_annotation\_last:.)

\g\_pdf\_backend\_link\_int To track annotations which are links.

```

2823 \int_new:N \g_pdf_backend_link_int

```

(End definition for \g\_pdf\_backend\_link\_int.)

\\_pdf\_backend\_link\_begin\_goto:nnw All created using the same internals.

\\_pdf\_backend\_link\_begin\_user:nnw

\\_pdf\_backend\_link\_begin:n

\\_pdf\_backend\_link\_end:

```

2824 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2825 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2826 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2827 { \_pdf_backend_link_begin:n {#1#2} }
2828 \cs_new_protected:Npx \_pdf_backend_link_begin:n #1
2829 {
2830     \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2831     {
2832         \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int
2833     }
2834     \_pdf_backend:x
2835     {
2836         bann ~
2837         \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2838         {
2839             @pdf.lnk
2840             \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int
2841             \c_space_tl
2842         }
2843         <<
2844         /Type /Annot
2845         #1
2846         >>
2847     }
2848 }
2849 \cs_new_protected:Npn \_pdf_backend_link_end:
2850 { \_pdf_backend:n { eann } }

```

(End definition for \\_pdf\_backend\_link\_begin\_goto:nnw and others.)

\\_pdf\_backend\_link\_last: Available using the backend mechanism with a suitably-recent version.

```

2851 \cs_new:Npx \_pdf_backend_link_last:
2852 {
2853     \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2854     {
2855         @pdf.lnk

```

```

2856         \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
2857     }
2858 }

```

(End definition for \\_\_pdf\_backend\_link\_last:.)

\\_\_pdf\_backend\_link\_margin:n Pass to dvipdfmx.

```

2859 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2860 { \__kernel_backend_literal:x { dvipdfmx:config~g~ \dim_eval:n {#1} } }

```

(End definition for \\_\_pdf\_backend\_link\_margin:n.)

```

\__pdf_backend_destination:nn
\__pdf_backend_destination:nnnn
\__pdf_backend_destination_aux:nnnn

```

Here, we need to turn the zoom into a scale. The method for FitR is from Alexander Grahn: the idea is to avoid needing to do any calculations in T<sub>E</sub>X by using the backend data for @xpos and @ypos. /FitR without rule spec doesn't work, so it falls back to /Fit here.

```

2861 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2862 {
2863     \__pdf_backend:x
2864     {
2865         dest ~ ( \exp_not:n {#1} )
2866         [
2867             @thispage
2868             \str_case:nnF {#2}
2869             {
2870                 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2871                 { fit } { /Fit }
2872                 { fitb } { /FitB }
2873                 { fitbh } { /FitBH }
2874                 { fitbv } { /FitBV ~ @xpos }
2875                 { fith } { /FitH ~ @ypos }
2876                 { fitv } { /FitV ~ @xpos }
2877                 { fitr } { /Fit }
2878             }
2879             { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2880         ]
2881     }
2882 }
2883 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2884 {
2885     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2886     { \dim_eval:n {#2} } {#1} {#3} {#4}
2887 }
2888 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2889 {
2890     \vbox_to_zero:n
2891     {
2892         \__kernel_kern:n {#4}
2893         \hbox:n
2894         {
2895             \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2896             \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2897         }
2898     }

```

```

2899     }
2900     \__kernel_kern:n {#1}
2901     \vbox_to_zero:n
2902     {
2903         \__kernel_kern:n { -#3 }
2904         \hbox:n
2905         {
2906             \__pdf_backend:n
2907             {
2908                 dest ~ (#2)
2909                 [
2910                     @thispage
2911                     /FitR ~
2912                     @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2913                     @xpos ~ @ypos
2914                 ]
2915             }
2916         }
2917         \tex_vss:D
2918     }
2919     \__kernel_kern:n { -#1 }
2920 }

```

(End definition for \\_\_pdf\_backend\_destination:nn, \\_\_pdf\_backend\_destination:nnnn, and \\_\_pdf\_backend\_destination\_aux:nnnn.)

#### 6.4.4 Structure

\\_pdf\_backend\_compresslevel:n  
\\_pdf\_backend\_compress\_objects:n

Pass data to the backend: these are a one-shot.

```

2921 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2922 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2923 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2924 {
2925     \bool_if:nF {#1}
2926     { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2927 }

```

(End definition for \\_pdf\_backend\_compresslevel:n and \\_pdf\_backend\_compress\_objects:n.)

\\_pdf\_backend\_version\_major\_gset:n  
\\_pdf\_backend\_version\_minor\_gset:n

We start with the assumption that the default is active.

```

2928 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2929 {
2930     \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
2931     \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
2932 }
2933 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2934 {
2935     \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
2936     \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
2937 }

```

(End definition for \\_pdf\_backend\_version\_major\_gset:n and \\_pdf\_backend\_version\_minor\_gset:n.)

\\_pdf\_backend\_version\_major:  
\\_pdf\_backend\_version\_minor:

We start with the assumption that the default is active.

```

2938 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2939 \cs_new:Npn \__pdf_backend_version_minor: { 5 }

```

(End definition for `\_pdf_backend_version_major:` and `\_pdf_backend_version_minor:.`)

### 6.4.5 Marked content

`\_pdf_backend_bdc:nn` Simple wrappers. May need refinement: see <https://chat.stackexchange.com/transcript/message/49970158#49970158>.  
`\_pdf_backend_emc:`

```
2940 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2941 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2942 \cs_new_protected:Npn \_pdf_backend_emc:
2943 { \__kernel_backend_literal_page:n { EMC } }
```

(End definition for `\_pdf_backend_bdc:nn` and `\_pdf_backend_emc:.`)

```
2944 </dvipdfmx | xetex>
```

## 6.5 dvisvgm backend

```
2945 <*dvisvgm>
```

### 6.5.1 Catalogue entries

`\_pdf_backend_catalog_gput:nn`  
`\_pdf_backend_info_gput:nn`

No-op.

```
2946 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2 { }
2947 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2 { }
```

(End definition for `\_pdf_backend_catalog_gput:nn` and `\_pdf_backend_info_gput:nn`.)

### 6.5.2 Objects

`\_pdf_backend_object_new:nn`  
`\_pdf_backend_object_ref:n`  
`\_pdf_backend_object_write:nn`  
`\_pdf_backend_object_write:nx`  
`\_pdf_backend_object_now:nn`  
`\_pdf_backend_object_now:nx`  
`\_pdf_backend_object_last:`  
`\_pdf_backend_pageobject_ref:n`

All no-ops here.

```
2948 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2 { }
2949 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
2950 \cs_new_protected:Npn \_pdf_backend_object_write:nn #1#2 { }
2951 \cs_new_protected:Npn \_pdf_backend_object_write:nx #1#2 { }
2952 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
2953 \cs_new_protected:Npn \_pdf_backend_object_now:nx #1#2 { }
2954 \cs_new:Npn \_pdf_backend_object_last: { }
2955 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1 { }
```

(End definition for `\_pdf_backend_object_new:nn` and others.)

### 6.5.3 Structure

`\_pdf_backend_compresslevel:n`  
`\_pdf_backend_compress_objects:n`

These are all no-ops.

```
2956 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1 { }
2957 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1 { }
```

(End definition for `\_pdf_backend_compresslevel:n` and `\_pdf_backend_compress_objects:n`.)

`\_pdf_backend_version_major_gset:n`  
`\_pdf_backend_version_minor_gset:n`

Data not available!

```
2958 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
2959 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1 { }
```

(End definition for `\_pdf_backend_version_major_gset:n` and `\_pdf_backend_version_minor_gset:n`.)

```

\__pdf_backend_version_major: Data not available!
\__pdf_backend_version_minor:
2960 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2961 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

(End definition for \__pdf_backend_version_major: and \__pdf_backend_version_minor:.)

\__pdf_backend_bdc:nn More no-ops.
\__pdf_backend_emc:
2962 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2963 \cs_new_protected:Npn \__pdf_backend_emc: { }

(End definition for \__pdf_backend_bdc:nn and \__pdf_backend_emc:.)

2964 </dvisvgm>
2965 </package>

```

## 7 l3backend-opacity Implementation

```

2966 <*package>
2967 <@@=opacity>

```

Although opacity is not color, it needs to be managed in a somewhat similar way: using a dedicated stack if possible. Depending on the backend, that may not be possible. There is also the need to cover fill/stroke setting as well as more general running opacity. It is easiest to describe the value used in terms of opacity, although commonly this is referred to as transparency.

```

2968 <*dvips>

```

```

\__opacity_backend_select:n No stack so set values directly.
\__opacity_backend_select_aux:n
2969 \cs_new_protected:Npn \__opacity_backend_select:n #1
2970 {
2971   \exp_args:Nx \__opacity_backend_select_aux:n
2972   { \fp_eval:n { min(max(0,#1),1) } }
2973 }
2974 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
2975 {
2976   \__kernel_backend_postscript:n
2977   { #1 ~ .setfillconstantalpha ~ #1 ~ .setstrokeconstantalpha }
2978 }

(End definition for \__opacity_backend_select:n and \__opacity_backend_select_aux:n.)

```

```

\__opacity_backend_fill:n Similar to the above but with no stack and only adding to one or other of the entries.
\__opacity_backend_stroke:n
\__opacity_backend:nn
\__opacity_backend:xn
2979 \cs_new_protected:Npn \__opacity_backend_fill:n #1
2980 { \__opacity_backend:xn { \fp_eval:n { min(max(0,#1),1) } } { fill } }
2981 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
2982 { \__opacity_backend:xn { \fp_eval:n { min(max(0,#1),1) } } { stroke } }
2983 \cs_new_protected:Npn \__opacity_backend:nn #1#2
2984 {
2985   \__kernel_backend_postscript:n { #1 ~ .set #2 constantalpha }
2986 }
2987 \cs_generate_variant:Nn \__opacity_backend:nn { x }

(End definition for \__opacity_backend_fill:n, \__opacity_backend_stroke:n, and \__opacity_backend:nn.)

2988 </dvips>

```

2989  $\langle$ \*dvi $\rangle$ pdfmx | luatex | pdftex | xetex $\rangle$

$\backslash$ c\_ opacity\_backend\_stack\_int Set up a stack.

```
2990 \cs_if_exist:NT \pdfmanagement_add:nnn
2991 {
2992   \__kernel_color_backend_stack_init:Nnn \c_ opacity_backend_stack_int
2993   { page ~ direct } { /opacity 1 ~ gs }
2994   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
2995   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
2996 }
```

(End definition for  $\backslash$ c\_ opacity\_backend\_stack\_int.)

$\backslash$ l\_ opacity\_backend\_fill\_tl We use tl here for speed: at the backend, this should be reasonable.

```
\l_ opacity_backend_stroke_tl
2997 \tl_new:N \l_ opacity_backend_fill_tl
2998 \tl_new:N \l_ opacity_backend_stroke_tl
```

(End definition for  $\backslash$ l\_ opacity\_backend\_fill\_tl and  $\backslash$ l\_ opacity\_backend\_stroke\_tl.)

$\backslash$ \_ opacity\_backend\_select:n Other than the need to evaluate the opacity as an fp, much the same as color.

```
\_ opacity_backend_select_aux:n
\_ opacity_backend_reset:
2999 \cs_new_protected:Npn \_ opacity_backend_select:n #1
3000 {
3001   \exp_args:Nx \_ opacity_backend_select_aux:n
3002   { \fp_eval:n { min(max(0,#1),1) } }
3003 }
3004 \cs_new_protected:Npn \_ opacity_backend_select_aux:n #1
3005 {
3006   \tl_set:Nn \l_ opacity_backend_fill_tl {#1}
3007   \tl_set:Nn \l_ opacity_backend_stroke_tl {#1}
3008   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3009   { opacity #1 }
3010   { << /ca ~ #1 /CA ~ #1 >> }
3011   \__kernel_color_backend_stack_push:nn \c_ opacity_backend_stack_int
3012   { /opacity #1 ~ gs }
3013   \group_insert_after:N \_ opacity_backend_reset:
3014 }
3015 \cs_if_exist:NF \pdfmanagement_add:nnn
3016 {
3017   \cs_gset_protected:Npn \_ opacity_backend_select_aux:n #1 { }
3018 }
3019 \cs_new_protected:Npn \_ opacity_backend_reset:
3020 { \__kernel_color_backend_stack_pop:n \c_ opacity_backend_stack_int }
```

(End definition for  $\backslash$ \_ opacity\_backend\_select:n,  $\backslash$ \_ opacity\_backend\_select\_aux:n, and  $\backslash$ \_ opacity\_backend\_reset:.)

$\backslash$ \_ opacity\_backend\_fill:n For separate fill and stroke, we need to work out if we need to do more work or if we can stick to a single setting.

```
\_ opacity_backend_stroke:n
\_ opacity_backend_fillstroke:nn
\_ opacity_backend_fillstroke:xx
3021 \cs_new_protected:Npn \_ opacity_backend_fill:n #1
3022 {
3023   \_ opacity_backend_fill_stroke:xx
3024   { \fp_eval:n { min(max(0,#1),1) } }
3025   \l_ opacity_backend_stroke_tl
3026 }
3027 \cs_new_protected:Npn \_ opacity_backend_stroke:n #1
```

```

3028 {
3029   \_opacity_backend_fill_stroke:xx
3030   \l\_opacity_backend_fill_tl
3031   { \fp_eval:n { min(max(0,#1),1) } }
3032 }
3033 \cs_new_protected:Npn \_opacity_backend_fill_stroke:nn #1#2
3034 {
3035   \str_if_eq:nnTF {#1} {#2}
3036   { \_opacity_backend_select_aux:n {#1} }
3037   {
3038     \tl_set:Nn \l\_opacity_backend_fill_tl {#1}
3039     \tl_set:Nn \l\_opacity_backend_stroke_tl {#2}
3040     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3041     { opacity.fill #1 }
3042     { << /ca ~ #1 >> }
3043     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3044     { opacity.stroke #1 }
3045     { << /CA ~ #2 >> }
3046     \_kernel_color_backend_stack_push:nn \c\_opacity_backend_stack_int
3047     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3048     \group_insert_after:N \_opacity_backend_reset:
3049   }
3050 }
3051 \cs_generate_variant:Nn \_opacity_backend_fill_stroke:nn { xx }

```

(End definition for \\_opacity\_backend\_fill:n, \\_opacity\_backend\_stroke:n, and \\_opacity\_backend\_fillstroke:nn.)

```

3052 </dvipdfmx | luatex | pdftex | xetex>
3053 < *dvipdfmx | xdvipdfmx>

```

\\_opacity\_backend\_select:n Older backends have no stack support, so everything is done directly.

```

3054 \int_compare:nNnT \c\_kernel_sys_dvipdfmx_version_int < { 20201111 }
3055 {
3056   \cs_gset_protected:Npn \_opacity_backend_select_aux:n #1
3057   {
3058     \tl_set:Nn \l\_opacity_backend_fill_tl {#1}
3059     \tl_set:Nn \l\_opacity_backend_stroke_tl {#1}
3060     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3061     { opacity #1 }
3062     { << /ca ~ #1 /CA ~ #1 >> }
3063     \_kernel_backend_literal_pdf:n { /opacity #1 ~ gs }
3064   }
3065   \cs_gset_protected:Npn \_opacity_backend_fill_stroke:nn #1#2
3066   {
3067     \str_if_eq:nnTF {#1} {#2}
3068     { \_opacity_backend_select_aux:n {#1} }
3069     {
3070       \tl_set:Nn \l\_opacity_backend_fill_tl {#1}
3071       \tl_set:Nn \l\_opacity_backend_stroke_tl {#2}
3072       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3073       { opacity.fill #1 }
3074       { << /ca ~ #1 >> }
3075       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3076       { opacity.stroke #1 }

```



```

3077         { << /CA ~ #2 >> }
3078         \__kernel_backend_literal_pdf:n
3079         { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3080     }
3081 }
3082 }

(End definition for \__opacity_backend_select:n.)

3083 </dvipdfmx | xdvipdfmx>
3084 <*dvisvgm>

```

\\_\_opacity\_backend\_select:n Once again, we use a scope here. There is a general opacity function for SVG, but that is of course not set up using the stack.

```

\__opacity_backend_fill:n
\__opacity_backend_stroke:n
\__opacity_backend:nn
3085 \cs_new_protected:Npn \__opacity_backend_select:n #1
3086 { \__opacity_backend:nn {#1} { } }
3087 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3088 { \__opacity_backend:nn {#1} { fill- } }
3089 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3090 { \__opacity_backend:nn { {#1} } { stroke- } }
3091 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3092 { \__kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

(End definition for \__opacity_backend_select:n and others.)

3093 </dvisvgm>
3094 </package>

```

## 8 l3backend-header Implementation

```

3095 <*dvips & header>

color.sc Empty definition for color at the top level.
3096 /color.sc { } def

(End definition for color.sc. This function is documented on page ??.)

```

TeXcolorseparation Support for separation/spot colors: this strange naming is so things work with the color stack.

```

3097 TeXDict begin
3098 /TeXcolorseparation { setcolor } def
3099 end

(End definition for TeXcolorseparation and separation. These functions are documented on page ??.)

```

pdf.globaldict A small global dictionary for backend use.

```

3100 true setglobal
3101 /pdf.globaldict 4 dict def
3102 false setglobal

(End definition for pdf.globaldict. This function is documented on page ??.)

```

pdf.cvs	Small utilities for PostScript manipulations. Conversion to DVI dimensions is done here
pdf.dvi.pt	to allow for Resolution. The total height of a rectangle (an array) needs a little maths,
pdf.pt.dvi	in contrast to simply extracting a value.
pdf.rect.ht	<pre> 3103 /pdf.cvs { 65534 string cvs } def 3104 /pdf.dvi.pt { 72.27 mul Resolution div } def 3105 /pdf.pt.dvi { 72.27 div Resolution mul } def 3106 /pdf.rect.ht { dup 1 get neg exch 3 get add } def </pre>
	<i>(End definition for pdf.cvs and others. These functions are documented on page ??.)</i>
pdf.linkmargin	Settings which are defined up-front in SDict.
pdf.linkdp.pad	<pre> 3107 /pdf.linkmargin { 1 pdf.pt.dvi } def </pre>
pdf.linkht.pad	<pre> 3108 /pdf.linkdp.pad { 0 } def 3109 /pdf.linkht.pad { 0 } def </pre>
	<i>(End definition for pdf.linkmargin, pdf.linkdp.pad, and pdf.linkht.pad. These functions are documented on page ??.)</i>
pdf.rect	Functions for marking the limits of an annotation/link, plus drawing the border. We
pdf.save.ll	separate links for generic annotations to support adding a margin and setting a minimal
pdf.save.ur	size.
pdf.save.linkll	<pre> 3110 /pdf.rect </pre>
pdf.save.linkur	<pre> 3111 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def </pre>
pdf.llx	<pre> 3112 /pdf.save.ll </pre>
pdf.lly	<pre> 3113 { </pre>
pdf.urx	<pre> 3114     currentpoint </pre>
pdf.ury	<pre> 3115     /pdf.lly exch def 3116     /pdf.llx exch def 3117 } </pre>
	<pre> 3118     def 3119 /pdf.save.ur </pre>
	<pre> 3120 { </pre>
	<pre> 3121     currentpoint </pre>
	<pre> 3122     /pdf.ury exch def 3123     /pdf.urx exch def 3124 } </pre>
	<pre> 3125     def 3126 /pdf.save.linkll </pre>
	<pre> 3127 { </pre>
	<pre> 3128     currentpoint </pre>
	<pre> 3129     pdf.linkmargin add 3130     pdf.linkdp.pad add 3131     /pdf.lly exch def 3132     pdf.linkmargin sub 3133     /pdf.llx exch def 3134 } </pre>
	<pre> 3135     def 3136 /pdf.save.linkur </pre>
	<pre> 3137 { </pre>
	<pre> 3138     currentpoint </pre>
	<pre> 3139     pdf.linkmargin sub 3140     pdf.linkht.pad sub 3141     /pdf.ury exch def 3142     pdf.linkmargin add </pre>

```

3143     /pdf.urx exch def
3144   }
3145   def

```

*(End definition for pdf.rect and others. These functions are documented on page ??.)*

pdf.dest.anchor For finding the anchor point of a destination link. We make the use case a separate function as it comes up a lot, and as this makes it easier to adjust if we need additional effects. We also need a more complex approach to convert a co-ordinate pair correctly when defining a rectangle: this can otherwise be out when using a landscape page. (Thanks to Alexander Grahn for the approach here.)

```

pdf.dev.x 3146 /pdf.dest.anchor
pdf.dev.y 3147 {
pdf.tmpa 3148   currentpoint exch
pdf.tmpb 3149   pdf.dvi.pt 72 add
pdf.tmpc 3150   /pdf.dest.x exch def
pdf.tmpd 3151   pdf.dvi.pt
3152   vsize 72 sub exch sub
3153   /pdf.dest.y exch def
3154 }
3155 def
3156 /pdf.dest.point
3157 { pdf.dest.x pdf.dest.y } def
3158 /pdf.dest2device
3159 {
3160   /pdf.dest.y exch def
3161   /pdf.dest.x exch def
3162   matrix currentmatrix
3163   matrix defaultmatrix
3164   matrix invertmatrix
3165   matrix concatmatrix
3166   cvx exec
3167   /pdf.dev.y exch def
3168   /pdf.dev.x exch def
3169   /pdf.tmpd exch def
3170   /pdf.tmpc exch def
3171   /pdf.tmpb exch def
3172   /pdf.tmpa exch def
3173   pdf.dest.x pdf.tmpa mul
3174   pdf.dest.y pdf.tmpc mul add
3175   pdf.dev.x add
3176   pdf.dest.x pdf.tmpb mul
3177   pdf.dest.y pdf.tmpd mul add
3178   pdf.dev.y add
3179 }
3180 def

```

*(End definition for pdf.dest.anchor and others. These functions are documented on page ??.)*

pdf.bordertracking To know where a breakable link can go, we need to track the boundary rectangle. That can be done by hooking into a and x operations: those names have to be retained. The boundary is stored at the end of the operation. Special effort is needed at the start and end of pages (or rather galleys), such that everything works properly.

```

pdf.bordertracking.begin 3181 /pdf.bordertracking false def
pdf.bordertracking.end
pdf.leftboundary
pdf.rightboundary
pdf.brokenlink.rect
pdf.brokenlink.skip
pdf.brokenlink.dict
pdf.bordertracking.endpage
pdf.bordertracking.continue
pdf.originx
pdf.originy

```

```

3182 /pdf.bordertracking.begin
3183 {
3184     SDict /pdf.bordertracking true put
3185     SDict /pdf.leftboundary undef
3186     SDict /pdf.rightboundary undef
3187     /a where
3188     {
3189         /a
3190         {
3191             currentpoint pop
3192             SDict /pdf.rightboundary known dup
3193             {
3194                 SDict /pdf.rightboundary get 2 index lt
3195                 { not }
3196                 if
3197             }
3198             if
3199             { pop }
3200             { SDict exch /pdf.rightboundary exch put }
3201             ifelse
3202             moveto
3203             currentpoint pop
3204             SDict /pdf.leftboundary known dup
3205             {
3206                 SDict /pdf.leftboundary get 2 index gt
3207                 { not }
3208                 if
3209             }
3210             if
3211             { pop }
3212             { SDict exch /pdf.leftboundary exch put }
3213             ifelse
3214         }
3215         put
3216     }
3217     if
3218 }
3219 def
3220 /pdf.bordertracking.end
3221 {
3222     /a where { /a { moveto } put } if
3223     /x where { /x { 0 exch rmoveto } put } if
3224     SDict /pdf.leftboundary known
3225     { pdf.outerbox 0 pdf.leftboundary put }
3226     if
3227     SDict /pdf.rightboundary known
3228     { pdf.outerbox 2 pdf.rightboundary put }
3229     if
3230     SDict /pdf.bordertracking false put
3231 }
3232 def
3233 /pdf.bordertracking.endpage
3234 {
3235     pdf.bordertracking

```

```

3236 {
3237     pdf.bordertracking.end
3238     true setglobal
3239     pdf.globaldict
3240     /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3241     pdf.globaldict
3242     /pdf.brokenlink.skip pdf.baselineskip put
3243     pdf.globaldict
3244     /pdf.brokenlink.dict
3245     pdf.link.dict pdf.cvs put
3246     false setglobal
3247     mark pdf.link.dict cvx exec /Rect
3248     [
3249         pdf.llx
3250         pdf.lly
3251         pdf.outerbox 2 get pdf.linkmargin add
3252         currentpoint exch pop
3253         pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3254     ]
3255     /ANN pdf.pdfmark
3256 }
3257 if
3258 }
3259 def
3260 /pdf.bordertracking.continue
3261 {
3262     /pdf.link.dict pdf.globaldict
3263     /pdf.brokenlink.dict get def
3264     /pdf.outerbox pdf.globaldict
3265     /pdf.brokenlink.rect get def
3266     /pdf.baselineskip pdf.globaldict
3267     /pdf.brokenlink.skip get def
3268     pdf.globaldict dup dup
3269     /pdf.brokenlink.dict undef
3270     /pdf.brokenlink.skip undef
3271     /pdf.brokenlink.rect undef
3272     currentpoint
3273     /pdf.originy exch def
3274     /pdf.originx exch def
3275     /a where
3276     {
3277         /a
3278         {
3279             moveto
3280             SDict
3281             begin
3282                 currentpoint pdf.originy ne exch
3283                 pdf.originx ne or
3284                 {
3285                     pdf.save.linkll
3286                     /pdf.lly
3287                     pdf.lly pdf.outerbox 1 get sub def
3288                     pdf.bordertracking.begin
3289                 }

```

```

3290         if
3291         end
3292     }
3293     put
3294 }
3295 if
3296 /x where
3297 {
3298     /x
3299     {
3300         0 exch rmoveto
3301         SDict
3302         begin
3303             currentpoint
3304             pdf.originy ne exch pdf.originx ne or
3305             {
3306                 pdf.save.link11
3307                 /pdf.lly
3308                 pdf.lly pdf.outerbox 1 get sub def
3309                 pdf.bordertracking.begin
3310             }
3311             if
3312             end
3313         }
3314         put
3315     }
3316     if
3317 }
3318 def

```

(End definition for pdf.bordertracking and others. These functions are documented on page ??.)

**pdf.breaklink** Dealing with link breaking itself has multiple stage. The first step is to find the **Rect** entry in the dictionary, looping over key-value pairs. The first line is handled first, adjusting the rectangle to stay inside the text area. The second phase is a loop over the height of the bulk of the link area, done on the basis of a number of baselines. Finally, the end of the link area is tidied up, again from the boundary of the text area.

```

3319 /pdf.breaklink
3320 {
3321     pop
3322     counttomark 2 mod 0 eq
3323     {
3324         counttomark /pdf.count exch def
3325         {
3326             pdf.count 0 eq { exit } if
3327             counttomark 2 roll
3328             1 index /Rect eq
3329             {
3330                 dup 4 array copy
3331                 dup dup
3332                 1 get
3333                 pdf.outerbox pdf.rect.ht
3334                 pdf.linkmargin 2 mul add sub
3335                 3 exch put

```

```

3336     dup
3337     pdf.outerbox 2 get
3338     pdf.linkmargin add
3339     2 exch put
3340   dup dup
3341     3 get
3342     pdf.outerbox pdf.rect.ht
3343     pdf.linkmargin 2 mul add add
3344     1 exch put
3345   /pdf.currentrect exch def
3346   pdf.breaklink.write
3347   {
3348     pdf.currentrect
3349     dup
3350     pdf.outerbox 0 get
3351     pdf.linkmargin sub
3352     0 exch put
3353     dup
3354     pdf.outerbox 2 get
3355     pdf.linkmargin add
3356     2 exch put
3357     dup dup
3358     1 get
3359     pdf.baselineskip add
3360     1 exch put
3361     dup dup
3362     3 get
3363     pdf.baselineskip add
3364     3 exch put
3365     /pdf.currentrect exch def
3366     pdf.breaklink.write
3367   }
3368   1 index 3 get
3369   pdf.linkmargin 2 mul add
3370   pdf.outerbox pdf.rect.ht add
3371   2 index 1 get sub
3372   pdf.baselineskip div round cvi 1 sub
3373   exch
3374   repeat
3375   pdf.currentrect
3376   dup
3377     pdf.outerbox 0 get
3378     pdf.linkmargin sub
3379     0 exch put
3380   dup dup
3381     1 get
3382     pdf.baselineskip add
3383     1 exch put
3384   dup dup
3385     3 get
3386     pdf.baselineskip add
3387     3 exch put
3388   dup 2 index 2 get 2 exch put
3389   /pdf.currentrect exch def

```

```

3390         pdf.breaklink.write
3391         SDict /pdf.pdfmark.good false put
3392         exit
3393     }
3394     { pdf.count 2 sub /pdf.count exch def }
3395     ifelse
3396 }
3397 loop
3398 }
3399 if
3400 /ANN
3401 }
3402 def
3403 /pdf.breaklink.write
3404 {
3405     counttomark 1 sub
3406     index /_objdef eq
3407     {
3408         counttomark -2 roll
3409         dup wcheck
3410         {
3411             readonly
3412             counttomark 2 roll
3413         }
3414         { pop pop }
3415         ifelse
3416     }
3417     if
3418     counttomark 1 add copy
3419     pop pdf.currentrect
3420     /ANN pdfmark
3421 }
3422 def

```

*(End definition for pdf.breaklink and others. These functions are documented on page ??.)*

<p>pdf.pdfmark</p> <p>pdf.pdfmark.good</p> <p>pdf.outerbox</p> <p>pdf.baselineskip</p> <p>pdf.pdfmark.dict</p>	<p>The business end of breaking links starts by hooking into pdfmarks. Unlike hypdvips, we avoid altering any links we have not created by using a copy of the core pdfmarks function. Only mark types which are known are altered. At present, this is purely ANN marks, which are measured relative to the size of the baseline skip. If they are more than one apparent line high, breaking is applied.</p>
--	--

```

3423 /pdf.pdfmark
3424 {
3425     SDict /pdf.pdfmark.good true put
3426     dup /ANN eq
3427     {
3428         pdf.pdfmark.store
3429         pdf.pdfmark.dict
3430         begin
3431             Subtype /Link eq
3432             currentdict /Rect known and
3433             SDict /pdf.outerbox known and
3434             SDict /pdf.baselineskip known and
3435             {

```



```

3436             Rect 3 get
3437             pdf.linkmargin 2 mul add
3438             pdf.outerbox pdf.rect.ht add
3439             Rect 1 get sub
3440             pdf.baselineskip div round cvi 0 gt
3441             { pdf.breaklink }
3442             if
3443             }
3444             if
3445             end
3446             SDict /pdf.outerbox undef
3447             SDict /pdf.baselineskip undef
3448             currentdict /pdf.pdfmark.dict undef
3449         }
3450         if
3451         pdf.pdfmark.good
3452         { pdfmark }
3453         { cleartomark }
3454         ifelse
3455     }
3456     def
3457 /pdf.pdfmark.store
3458 {
3459     /pdf.pdfmark.dict 65534 dict def
3460     counttomark 1 add copy
3461     pop
3462     {
3463         dup mark eq
3464         {
3465             pop
3466             exit
3467         }
3468         {
3469             pdf.pdfmark.dict
3470             begin def end
3471         }
3472         ifelse
3473     }
3474     loop
3475 }
3476 def

```

*(End definition for pdf.pdfmark and others. These functions are documented on page ??.)*

```

3477 </dvips & header>

```

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

- \\_ ..... 146
- A**
- \AtBeginDvi ..... 59, 60, 518, 519, 671, 672
- B**
- bool commands:
  - \bool\_gset\_false:N ..... 1170, 1189, 1212, 1234, 1250, 1351, 1588, 1624, 2189, 2235
  - \bool\_gset\_true:N ..... 1168, 1237, 1349, 1603, 2182, 2188
  - \bool\_if:NTF ..... 57, 669, 1180, 1184, 1200, 1203, 1207, 1218, 1225, 1229, 1241, 1245, 1362, 1367, 1372, 1562, 1607, 1720, 1755, 1865, 1907, 2177, 2192, 2197, 2202
  - \bool\_if:nTF ..... 2411, 2671, 2925
  - \bool\_lazy\_or:nnTF ..... 1747, 1900
  - \bool\_new:N ..... 1171, 1238, 1352, 1604, 2165, 2166
  - \bool\_set\_false:N ..... 1730, 1832, 1925, 1989
- box commands:
  - \box\_dp:N ..... 216, 218, 266, 268, 323, 325, 372, 374, 376, 378, 2214, 2247, 2248, 2273
  - \box\_ht:N ..... 218, 268, 325, 376, 378, 1767, 1962, 2219, 2258, 2259, 2275
  - \box\_if\_empty:NTF ..... 2308
  - \box\_move\_down:nn ..... 2139, 2214
  - \box\_move\_up:nn ..... 2141, 2219
  - \box\_new:N ..... 2024, 2129, 2130
  - \box\_set\_dp:Nn ..... 1687
  - \box\_set\_ht:Nn ..... 1686
  - \box\_set\_wd:Nn ..... 280, 1685
  - \box\_use:N ..... 223, 241, 255, 271, 298, 312, 328, 344, 356, 407, 421, 440, 1302, 1497, 1688, 2170
  - \box\_wd:N ..... 217, 225, 267, 273, 324, 330, 373, 375, 1766, 1961
- box internal commands:
  - \\_\_box\_backend\_clip:N ..... 205, 260, 317, 361
  - \l\_box\_backend\_cos\_fp ..... 275
  - \\_\_box\_backend\_rotate:Nn ..... 227, 275, 332, 411
  - \\_\_box\_backend\_rotate\_aux:Nn ..... 227, 275, 332
  - \\_\_box\_backend\_scale:Nnn ..... 244, 303, 347, 424
  - \l\_box\_backend\_sin\_fp ..... 275
  - \g\_box\_clip\_path\_int ..... 361
- C**
- char commands:
  - \char\_set\_catcode\_space:n ..... 146
- clist commands:
  - \clist\_map\_function:nN ... 1258, 1382
  - \clist\_map\_function:nn ..... 1631
- color internal commands:
  - \\_\_color\_backend:nnn ..... 1056
  - \\_\_color\_backend\_cmyk:w ..... 1057
  - \\_\_color\_backend\_devicen\_init:n ..... 904
  - \\_\_color\_backend\_devicen\_init:nnn ..... 817, 904
  - \\_\_color\_backend\_devicen\_init:w ..... 904
  - \\_\_color\_backend\_fill:n ..... 963, 990, 1020, 1038, 1045
  - \\_\_color\_backend\_fill\_cmyk:n ..... 963, 997, 1020, 1045
  - \\_\_color\_backend\_fill\_devicen:nn ..... 989, 1011, 1037, 1107
  - \\_\_color\_backend\_fill\_gray:n ..... 963, 997, 1020, 1045
  - \\_\_color\_backend\_fill\_rgb:n ..... 963, 997, 1020, 1045
  - \\_\_color\_backend\_fill\_separation:nn ..... 989, 997, 1037, 1107
  - \l\_color\_backend\_fill\_tl ..... 631, 641, 971, 986
  - \c\_color\_backend\_main\_stack\_int ..... 508
  - \\_\_color\_backend\_pickup:N .. 448, 471
  - \\_\_color\_backend\_pickup:w 14, 448, 471
  - \\_\_color\_backend\_reset: ..... 613, 633, 650, 974, 987, 997, 1029, 1054
  - \\_\_color\_backend\_rgb:w ..... 1080
  - \\_\_color\_backend\_select:n .. 613, 665
  - \\_\_color\_backend\_select:nn .. 633, 844
  - \\_\_color\_backend\_select\_cmyk:n .. 613, 633, 650
  - \\_\_color\_backend\_select\_devicen:nn ..... 664, 837, 843, 955
  - \\_\_color\_backend\_select\_gray:n .. 613, 633, 650

\__color_backend_select_rgb:n ...	\l__color_backend_stroke_tl ....
..... <a href="#">613</a> , <a href="#">633</a> , <a href="#">650</a>	..... <a href="#">631</a> , <a href="#">642</a> , <a href="#">973</a> , <a href="#">984</a>
\__color_backend_select_separation:nn	\g__color_model_int <a href="#">687</a> , <a href="#">823</a> , <a href="#">868</a> , <a href="#">941</a>
..... <a href="#">664</a> , <a href="#">837</a> , <a href="#">843</a> , <a href="#">955</a>	\c__color_model_range_CIELAB_tl .
\__color_backend_separation-	..... <a href="#">778</a> , <a href="#">813</a> , <a href="#">888</a> , <a href="#">895</a>
init:n ..... <a href="#">667</a> , <a href="#">846</a> , <a href="#">928</a> , <a href="#">952</a>	color.sc ..... <a href="#">613</a> , <a href="#">3096</a>
\__color_backend_separation-	cs commands:
init:nnn ..... <a href="#">667</a>	\cs_generate_variant:Nn .. <a href="#">49</a> , <a href="#">53</a> ,
\__color_backend_separation-	<a href="#">56</a> , <a href="#">91</a> , <a href="#">130</a> , <a href="#">135</a> , <a href="#">162</a> , <a href="#">193</a> , <a href="#">199</a> , <a href="#">563</a> ,
init:nnnn ..... <a href="#">667</a>	<a href="#">600</a> , <a href="#">680</a> , <a href="#">1117</a> , <a href="#">1312</a> , <a href="#">1506</a> , <a href="#">1879</a> ,
\__color_backend_separation-	<a href="#">1936</a> , <a href="#">1952</a> , <a href="#">2028</a> , <a href="#">2065</a> , <a href="#">2124</a> , <a href="#">2607</a> ,
init:nnnnn ..... <a href="#">667</a> , <a href="#">839</a> , <a href="#">846</a>	<a href="#">2635</a> , <a href="#">2745</a> , <a href="#">2767</a> , <a href="#">2802</a> , <a href="#">2987</a> , <a href="#">3051</a>
\__color_backend_separation-	\cs_gset:Npx ..... <a href="#">2930</a> , <a href="#">2935</a>
init:nw ..... <a href="#">667</a>	\cs_gset_eq:NN ..... <a href="#">657</a> ,
\__color_backend_separation-	<a href="#">658</a> , <a href="#">958</a> , <a href="#">1004</a> , <a href="#">1005</a> , <a href="#">1011</a> , <a href="#">1013</a> , <a href="#">1015</a>
init:w ..... <a href="#">667</a>	\cs_gset_protected:Npn .....
\__color_backend_separation-	..... <a href="#">545</a> , <a href="#">652</a> , <a href="#">659</a> , <a href="#">901</a> , <a href="#">957</a> ,
init_/DeviceCMYK:nnn ..... <a href="#">667</a>	<a href="#">999</a> , <a href="#">1006</a> , <a href="#">1008</a> , <a href="#">1010</a> , <a href="#">3017</a> , <a href="#">3056</a> , <a href="#">3065</a>
\__color_backend_separation-	\cs_if_exist:NTF ..... <a href="#">27</a> ,
init_/DeviceGray:nnn ..... <a href="#">667</a>	<a href="#">59</a> , <a href="#">449</a> , <a href="#">472</a> , <a href="#">518</a> , <a href="#">533</a> , <a href="#">671</a> , <a href="#">862</a> ,
\__color_backend_separation-	<a href="#">899</a> , <a href="#">935</a> , <a href="#">2304</a> , <a href="#">2696</a> , <a href="#">2722</a> , <a href="#">2990</a> , <a href="#">3015</a>
init_/DeviceRGB:nnn ..... <a href="#">667</a>	\cs_if_exist_use:NTF ..... <a href="#">38</a> , <a href="#">693</a>
\__color_backend_separation-	\cs_new:Npn ..... <a href="#">702</a> , <a href="#">704</a> , <a href="#">706</a> ,
init_aux:nnnnn ..... <a href="#">667</a>	<a href="#">708</a> , <a href="#">715</a> , <a href="#">721</a> , <a href="#">723</a> , <a href="#">729</a> , <a href="#">746</a> , <a href="#">753</a> ,
\__color_backend_separation-	<a href="#">755</a> , <a href="#">946</a> , <a href="#">1263</a> , <a href="#">1387</a> , <a href="#">1635</a> , <a href="#">1965</a> ,
init_CIELAB:nnn .... <a href="#">667</a> , <a href="#">839</a> , <a href="#">846</a>	<a href="#">1974</a> , <a href="#">2018</a> , <a href="#">2043</a> , <a href="#">2125</a> , <a href="#">2127</a> , <a href="#">2160</a> ,
\__color_backend_separation-	<a href="#">2329</a> , <a href="#">2423</a> , <a href="#">2424</a> , <a href="#">2577</a> , <a href="#">2608</a> , <a href="#">2609</a> ,
init_CIELAB:nnnnnn ..... <a href="#">840</a>	<a href="#">2727</a> , <a href="#">2760</a> , <a href="#">2803</a> , <a href="#">2805</a> , <a href="#">2821</a> , <a href="#">2938</a> ,
\__color_backend_separation-	<a href="#">2939</a> , <a href="#">2949</a> , <a href="#">2954</a> , <a href="#">2955</a> , <a href="#">2960</a> , <a href="#">2961</a>
init_count:n ..... <a href="#">667</a>	\cs_new:Npx .....
\__color_backend_separation-	.. <a href="#">2444</a> , <a href="#">2479</a> , <a href="#">2636</a> , <a href="#">2647</a> , <a href="#">2714</a> , <a href="#">2851</a>
init_count:w ..... <a href="#">667</a>	\cs_new_eq:NN .. <a href="#">46</a> , <a href="#">666</a> , <a href="#">845</a> , <a href="#">952</a> ,
\__color_backend_separation-	<a href="#">993</a> , <a href="#">994</a> , <a href="#">1041</a> , <a href="#">1042</a> , <a href="#">1109</a> , <a href="#">1110</a> ,
init_Device:Nn ..... <a href="#">667</a>	<a href="#">1116</a> , <a href="#">1311</a> , <a href="#">1317</a> , <a href="#">1318</a> , <a href="#">1505</a> , <a href="#">1512</a> ,
\g__color_backend_stack_int ... <a href="#">508</a>	<a href="#">1697</a> , <a href="#">1726</a> , <a href="#">1777</a> , <a href="#">1778</a> , <a href="#">1820</a> , <a href="#">1828</a> ,
\l__color_backend_stack_int ....	<a href="#">1850</a> , <a href="#">1921</a> , <a href="#">1978</a> , <a href="#">1985</a> , <a href="#">2017</a> , <a href="#">2170</a>
..... <a href="#">505</a> , <a href="#">535</a> , <a href="#">541</a> , <a href="#">643</a> , <a href="#">647</a> , <a href="#">972</a> , <a href="#">985</a>	\cs_new_protected:Npn ..... <a href="#">47</a> ,
\__color_backend_stroke:n .....	<a href="#">51</a> , <a href="#">54</a> , <a href="#">64</a> , <a href="#">70</a> , <a href="#">75</a> , <a href="#">77</a> , <a href="#">81</a> , <a href="#">92</a> , <a href="#">102</a> ,
..... <a href="#">963</a> , <a href="#">992</a> , <a href="#">997</a>	<a href="#">111</a> , <a href="#">120</a> , <a href="#">133</a> , <a href="#">136</a> , <a href="#">138</a> , <a href="#">140</a> , <a href="#">160</a> ,
\__color_backend_stroke_cmyk:n ..	<a href="#">165</a> , <a href="#">174</a> , <a href="#">184</a> , <a href="#">194</a> , <a href="#">205</a> , <a href="#">227</a> , <a href="#">229</a> ,
..... <a href="#">963</a> , <a href="#">1020</a> , <a href="#">1056</a>	<a href="#">244</a> , <a href="#">260</a> , <a href="#">275</a> , <a href="#">277</a> , <a href="#">303</a> , <a href="#">317</a> , <a href="#">332</a> ,
\__color_backend_stroke_cmyk:w <a href="#">1056</a>	<a href="#">334</a> , <a href="#">347</a> , <a href="#">361</a> , <a href="#">411</a> , <a href="#">424</a> , <a href="#">448</a> , <a href="#">466</a> ,
\__color_backend_stroke_devicen:nn	<a href="#">471</a> , <a href="#">479</a> , <a href="#">509</a> , <a href="#">554</a> , <a href="#">564</a> , <a href="#">576</a> , <a href="#">590</a> ,
..... <a href="#">989</a> , <a href="#">1015</a> , <a href="#">1037</a> , <a href="#">1107</a>	<a href="#">601</a> , <a href="#">613</a> , <a href="#">615</a> , <a href="#">617</a> , <a href="#">619</a> , <a href="#">627</a> , <a href="#">633</a> ,
\__color_backend_stroke_gray:n ..	<a href="#">635</a> , <a href="#">637</a> , <a href="#">639</a> , <a href="#">646</a> , <a href="#">664</a> , <a href="#">681</a> , <a href="#">771</a> ,
..... <a href="#">963</a> , <a href="#">1020</a> , <a href="#">1056</a>	<a href="#">817</a> , <a href="#">837</a> , <a href="#">838</a> , <a href="#">839</a> , <a href="#">840</a> , <a href="#">843</a> , <a href="#">846</a> ,
\__color_backend_stroke_gray-	<a href="#">873</a> , <a href="#">877</a> , <a href="#">904</a> , <a href="#">963</a> , <a href="#">965</a> , <a href="#">967</a> , <a href="#">969</a> ,
aux:n ..... <a href="#">1056</a>	<a href="#">976</a> , <a href="#">978</a> , <a href="#">980</a> , <a href="#">982</a> , <a href="#">989</a> , <a href="#">991</a> , <a href="#">1020</a> ,
\__color_backend_stroke_rgb:n ...	<a href="#">1022</a> , <a href="#">1024</a> , <a href="#">1026</a> , <a href="#">1031</a> , <a href="#">1033</a> , <a href="#">1035</a> ,
..... <a href="#">963</a> , <a href="#">1020</a> , <a href="#">1056</a>	<a href="#">1037</a> , <a href="#">1039</a> , <a href="#">1045</a> , <a href="#">1047</a> , <a href="#">1049</a> , <a href="#">1051</a> ,
\__color_backend_stroke_rgb:w . <a href="#">1056</a>	<a href="#">1056</a> , <a href="#">1058</a> , <a href="#">1069</a> , <a href="#">1077</a> , <a href="#">1079</a> , <a href="#">1081</a> ,
\__color_backend_stroke_separation:nn	<a href="#">1107</a> , <a href="#">1108</a> , <a href="#">1118</a> , <a href="#">1123</a> , <a href="#">1128</a> , <a href="#">1130</a> ,
..... <a href="#">989</a> , <a href="#">997</a> , <a href="#">1037</a> , <a href="#">1107</a>	<a href="#">1132</a> , <a href="#">1140</a> , <a href="#">1148</a> , <a href="#">1157</a> , <a href="#">1167</a> , <a href="#">1169</a> ,
	<a href="#">1172</a> , <a href="#">1174</a> , <a href="#">1191</a> , <a href="#">1196</a> , <a href="#">1214</a> , <a href="#">1236</a> ,

1239, 1252, 1265, 1270, 1272, 1274,  
 1276, 1278, 1280, 1282, 1284, 1289,  
 1313, 1315, 1319, 1324, 1329, 1339,  
 1348, 1350, 1353, 1355, 1357, 1359,  
 1364, 1369, 1374, 1376, 1389, 1394,  
 1396, 1398, 1400, 1402, 1404, 1406,  
 1408, 1419, 1444, 1456, 1468, 1480,  
 1487, 1507, 1513, 1518, 1523, 1534,  
 1544, 1554, 1556, 1558, 1560, 1591,  
 1593, 1598, 1600, 1602, 1605, 1626,  
 1637, 1650, 1652, 1654, 1656, 1658,  
 1660, 1662, 1664, 1666, 1674, 1698,  
 1712, 1727, 1739, 1744, 1772, 1784,  
 1797, 1807, 1822, 1829, 1837, 1848,  
 1852, 1855, 1870, 1880, 1915, 1922,  
 1928, 1934, 1937, 1944, 1953, 1958,  
 1966, 1979, 1986, 1992, 1994, 1996,  
 2007, 2026, 2029, 2031, 2035, 2045,  
 2066, 2071, 2076, 2081, 2091, 2096,  
 2104, 2132, 2137, 2169, 2171, 2173,  
 2175, 2180, 2195, 2200, 2237, 2266,  
 2285, 2294, 2331, 2338, 2364, 2369,  
 2397, 2409, 2421, 2422, 2425, 2427,  
 2431, 2455, 2457, 2459, 2470, 2490,  
 2500, 2523, 2537, 2547, 2558, 2579,  
 2610, 2658, 2669, 2675, 2703, 2737,  
 2739, 2746, 2748, 2752, 2762, 2768,  
 2773, 2778, 2783, 2785, 2787, 2795,  
 2808, 2824, 2826, 2849, 2859, 2861,  
 2883, 2888, 2921, 2923, 2928, 2933,  
 2940, 2942, 2946, 2947, 2948, 2950,  
 2951, 2952, 2953, 2956, 2957, 2958,  
 2959, 2962, 2963, 2969, 2974, 2979,  
 2981, 2983, 2999, 3004, 3019, 3021,  
 3027, 3033, 3085, 3087, 3089, 3091  
 \cs\_new\_protected:Npx .....  
 .... 512, 667, 1092, 2686, 2743, 2828  
 \cs\_set:Npn ..... 144  
 \cs\_set\_eq:NN ..... 2325, 2326  
 \cs\_set\_protected:Npn ..... 451, 474

## D

dim commands:

\dim\_eval:n ..... 2135, 2367,  
 2439, 2440, 2441, 2498, 2533, 2534,  
 2535, 2815, 2816, 2817, 2860, 2886  
 \dim\_max:nn ..... 2245, 2256  
 \dim\_set:Nn ... 1766, 1767, 1961, 1962  
 \dim\_to\_decimal:n .. 372, 373, 374,  
 375, 376, 378, 1516, 1521, 1527,  
 1528, 1529, 1530, 1539, 1540, 1541,  
 1632, 1651, 2012, 2013, 2243, 2254,  
 2272, 2273, 2274, 2275, 2279, 2335  
 \dim\_to\_decimal\_in\_bp:n .....

.... 216, 217, 218, 266, 267, 268,  
 323, 324, 325, 1136, 1137, 1144,  
 1145, 1152, 1153, 1161, 1162, 1163,  
 1260, 1264, 1268, 1322, 1327, 1333,  
 1334, 1335, 1343, 1344, 1384, 1388,  
 1392, 1636, 1703, 1704, 1705, 1706,  
 1842, 1843, 1844, 1845, 1894, 1895,  
 1896, 1897, 2001, 2002, 2003, 2004

draw internal commands:

\\_\_draw\_align\_currentpoint:... .. 35  
 \\_\_draw\_backend\_add\_to\_path:n ...  
 ..... 1513, 1559  
 \\_\_draw\_backend\_begin: .....  
 ..... 1118, 1313, 1507  
 \\_\_draw\_backend\_box\_use:Nnnnn ...  
 ..... 30, 1289, 1487, 1674  
 \\_\_draw\_backend\_cap\_but: .....  
 ..... 1252, 1376, 1626  
 \\_\_draw\_backend\_cap\_rectangle: ..  
 ..... 1252, 1376, 1626  
 \\_\_draw\_backend\_cap\_round: .....  
 ..... 1252, 1376, 1626  
 \\_\_draw\_backend\_clip: 1172, 1353, 1558  
 \\_\_draw\_backend\_closepath: .....  
 ..... 1172, 1353, 1558  
 \\_\_draw\_backend\_closestroke: ...  
 ..... 1172, 1353, 1558  
 \\_\_draw\_backend\_cm:nnnn 1284, 1297,  
 1298, 1299, 1408, 1491, 1666, 1677  
 \\_\_draw\_backend\_cm\_aux:nnnn .. 1408  
 \\_\_draw\_backend\_cm\_decompose:nnnnN  
 ..... 1414, 1443  
 \\_\_draw\_backend\_cm\_decompose\_-  
 aux:nnnnN ..... 1443  
 \\_\_draw\_backend\_cm\_decompose\_-  
 auxii:nnnnN ..... 1443  
 \\_\_draw\_backend\_cm\_decompose\_-  
 auxiii:nnnnN ..... 1443  
 \\_\_draw\_backend\_curveto:nnnnnn ..  
 ..... 1132, 1319, 1513  
 \\_\_draw\_backend\_dash:n .....  
 ..... 1252, 1376, 1626  
 \\_\_draw\_backend\_dash\_aux:nn .. 1626  
 \\_\_draw\_backend\_dash\_pattern:nn ..  
 ..... 1252, 1376, 1626  
 \\_\_draw\_backend\_discardpath: ...  
 ..... 1172, 1353, 1558  
 \\_\_draw\_backend\_end: 1118, 1313, 1507  
 \\_\_draw\_backend\_evenodd\_rule: ...  
 ..... 1167, 1348, 1554  
 \\_\_draw\_backend\_fill: 1172, 1353, 1558  
 \\_\_draw\_backend\_fillstroke: .....  
 ..... 1172, 1353, 1558

<code>\__draw_backend_join_bevel:</code> . . . .	
1252, 1376, 1626	
<code>\__draw_backend_join_miter:</code> . . . .	
1252, 1376, 1626	
<code>\__draw_backend_join_round:</code> . . . .	
1252, 1376, 1626	
<code>\__draw_backend_lineto:nn</code> . . . . .	
1132, 1319, 1513	
<code>\__draw_backend_linewidth:n</code> . . . .	
1252, 1376, 1626	
<code>\__draw_backend_literal:n</code> . . . . .	
1116, 1121, 1125, 1129,	
1131, 1134, 1142, 1150, 1159, 1173,	
1176, 1177, 1178, 1179, 1182, 1188,	
1198, 1205, 1211, 1216, 1221, 1222,	
1223, 1224, 1227, 1233, 1243, 1249,	
1254, 1267, 1271, 1273, 1275, 1277,	
1279, 1281, 1283, 1286, 1291, 1292,	
1293, 1294, 1295, 1296, 1300, 1301,	
1303, 1304, 1305, 1306, 1307, 1311,	
1321, 1326, 1331, 1341, 1354, 1356,	
1358, 1361, 1366, 1371, 1375, 1378,	
1391, 1395, 1397, 1399, 1401, 1403,	
1405, 1407, 1505, 1565, 1584, 1610	
<code>\__draw_backend_miterlimit:n</code> . . .	
1252, 1376, 1626	
<code>\__draw_backend_moveto:nn</code> . . . . .	
1132, 1319, 1513	
<code>\__draw_backend_nonzero_rule:</code> . . .	
1167, 1348, 1554	
<code>\__draw_backend_path:n</code> . . . . .	1558
<code>\__draw_backend_rectangle:nnnn</code> . .	
1132, 1319, 1513	
<code>\__draw_backend_scope:n</code> 1555, 1557,	
1577, 1617, 1639, 1651, 1653, 1655,	
1657, 1659, 1661, 1663, 1665, 1668	
<code>\__draw_backend_scope_begin:</code> . . .	
1128, 1314, 1317	
<code>\__draw_backend_scope_end:</code> . . . . .	
1128, 1316, 1317	
<code>\__draw_backend_stroke:</code> . . . . .	
1172, 1353, 1558	
<code>\g__draw_clip_path_int</code> . . . . .	
1564, 1567, 1580, 1609, 1612, 1620	
<code>\g__draw_draw_clip_bool</code> . . . . .	1172, 1558
<code>\g__draw_draw_eor_bool</code> . . . . .	
1167, 1184, 1200, 1207, 1218,	
1229, 1245, 1348, 1362, 1367, 1372	
<code>\g__draw_draw_path_int</code> . . . . .	1558
<code>\g__draw_draw_path_tl</code> . . . . .	
1513, 1569, 1585, 1587, 1614, 1623	
<code>\g__draw_path_int</code> . . . . .	1573, 1590
	<b>E</b>
<code>\errmessage</code> . . . . .	38
<code>\evensidemargin</code> . . . . .	2212
exp commands:	
<code>\exp_after:wN</code> . . . . .	151, 457, 1972
<code>\exp_args:Ne</code> . . . . .	717, 2366, 2885
<code>\exp_args:Nf</code> . . . . .	1257, 1381, 2134
<code>\exp_args:NNf</code> . . . . .	228, 276, 333
<code>\exp_args:Nnx</code> . . . . .	2121, 2798
<code>\exp_args:NV</code> . . . . .	453
<code>\exp_args:Nx</code> . . . . .	1790, 1811,
2078, 2093, 2208, 2764, 2971, 3001	
<code>\exp_last_unbraced:Nx</code> . . . . .	462, 476
<code>\exp_not:N</code> . . . . .	514, 515, 519, 520, 525,
527, 672, 675, 2446, 2448, 2451,	
2481, 2483, 2486, 2638, 2640, 2643,	
2649, 2651, 2654, 2691, 2692, 2698,	
2699, 2718, 2723, 2832, 2840, 2856	
<code>\exp_not:n</code> . . . . .	48, 89, 100, 128, 2069,
2074, 2360, 2593, 2594, 2608, 2609,	
2621, 2622, 2776, 2781, 2792, 2865	
<code>\ExplBackendFileDate</code> . . . . .	1
	<b>F</b>
file commands:	
<code>\file_compare_timestamp:nNnTF</code> . . . . .	1799
<code>\file_parse_full_name:nNNN</code> . . . . .	1786, 1809
fp commands:	
<code>\fp_compare:nNnTF</code> . . . . .	
235, 282, 288, 340, 1424, 1437, 1482	
<code>\fp_eval:n</code> . . . . .	228, 237, 250, 251, 276,
293, 308, 310, 333, 342, 353, 354,	
418, 433, 434, 1064, 1065, 1066,	
1074, 1087, 1088, 1089, 1426, 1431,	
1432, 1439, 1449, 1450, 1451, 1452,	
1461, 1462, 1463, 1464, 1473, 1474,	
1475, 1476, 2357, 2520, 2879, 2972,	
2980, 2982, 3002, 3024, 3031, 3092	
<code>\fp_new:N</code> . . . . .	301, 302
<code>\fp_set:Nn</code> . . . . .	281, 284
<code>\fp_use:N</code> . . . . .	287, 291, 296
<code>\fp_zero:N</code> . . . . .	283
<code>\c_zero_fp</code> . . . . .	235, 282, 288, 340, 1424, 1437
	<b>G</b>
graphics commands:	
<code>\graphics_bb_restore:nTF</code> . . . . .	1741, 1955
<code>\graphics_bb_save:n</code> . . . . .	1770, 1963
<code>\l_graphics_decodearray_tl</code> . . . . .	
1718, 1719,	
1729, 1749, 1753, 1754, 1831, 1863,	
1864, 1902, 1905, 1906, 1924, 1988	
<code>\graphics_extract_bb:n</code> . . . . .	
1826, 1833, 1983, 1990	

<code>\l_graphics_interpolate_bool</code> . . .	<code>\__graphics_backend_include:nn</code> <a href="#">1992</a>
. . . . . <a href="#">1720, 1730, 1748, 1755,</a>	<code>\__graphics_backend_include_</code>
<a href="#">1832, 1865, 1901, 1907, 1925, 1989</a>	auxi:nn . . . . . <a href="#">1837</a>
<code>\l_graphics_llx_dim</code> . . . . .	<code>\__graphics_backend_include_</code>
. . . . . <a href="#">1703, 1842, 1894, 2001</a>	auxii:nnn . . . . . <a href="#">1837</a>
<code>\l_graphics_lly_dim</code> . . . . .	<code>\__graphics_backend_include_</code>
. . . . . <a href="#">1704, 1843, 1895, 2002</a>	auxiii:nnn . . . . . <a href="#">1837</a>
<code>\l_graphics_name_tl</code> . . . . . <a href="#">1804</a>	<code>\__graphics_backend_include_</code>
<code>\l_graphics_page_int</code> . . . . .	bitmap_quote:w . . . . . <a href="#">1966, 2007</a>
. . . . . <a href="#">1714, 1734, 1735, 1759,</a>	<code>\__graphics_backend_include_</code>
<a href="#">1760, 1824, 1861, 1862, 1888, 1889,</a>	eps:n . . . . . <a href="#">1698, 1779, 1837, 1992</a>
<a href="#">1917, 1930, 1931, 1970, 1971, 1981</a>	<code>\__graphics_backend_include_</code>
<code>\l_graphics_pagebox_tl</code> . . . . .	jpg:n . . . . . <a href="#">1772, 1837, 2007</a>
. . . . . <a href="#">51, 1715, 1733,</a>	<code>\__graphics_backend_include_</code>
<a href="#">1761, 1762, 1825, 1859, 1860, 1890,</a>	pdf:n . . <a href="#">1772, 1811, 1837, 1966, 1992</a>
<a href="#">1892, 1918, 1939, 1940, 1972, 1982</a>	<code>\__graphics_backend_include_pdf_</code>
<code>\graphics_read_bb:n</code> . <a href="#">1697, 1820, 1978</a>	quote:w . . . . . <a href="#">1969, 1974</a>
<code>\l_graphics_urx_dim</code> . . . . .	<code>\__graphics_backend_include_</code>
. . <a href="#">1705, 1766, 1844, 1896, 1961, 2003</a>	png:n . . . . . <a href="#">1772, 1837, 2007</a>
<code>\l_graphics_ury_dim</code> . . <a href="#">1706, 1767,</a>	<code>\l_graphics_backend_name_str</code> . <a href="#">1779</a>
<a href="#">1845, 1897, 1962, 2004, 2012, 2013</a>	<code>\l_graphics_graphics_attr_tl</code> . . .
graphics internal commands:	. . . . . <a href="#">1711, 1716,</a>
<code>\l_graphics_backend_dir_str</code> . <a href="#">1779</a>	<a href="#">1723, 1731, 1741, 1768, 1770, 1775</a>
<code>\l_graphics_backend_ext_str</code> . <a href="#">1779</a>	<code>\l_graphics_internal_box</code> . . . . .
<code>\__graphics_backend_getbb_auxi:n</code>	. . <a href="#">1764, 1766, 1767, 1960, 1961, 1962</a>
. . . . . <a href="#">1712</a>	<code>\g_graphics_track_int</code> . . . . .
<code>\__graphics_backend_getbb_</code>	. . . . . <a href="#">1836, 1882, 1883</a>
auxi:nN . . . . . <a href="#">1915</a>	group commands:
<code>\__graphics_backend_getbb_</code>	<code>\group_begin:</code> . . . . . <a href="#">143, 171, 190</a>
auxii:n . . . . . <a href="#">1712</a>	<code>\group_end:</code> . . . . . <a href="#">156, 179</a>
<code>\__graphics_backend_getbb_</code>	<code>\group_insert_after:N</code> <a href="#">625, 644, 655,</a>
auxii:nnN . . . . . <a href="#">1915</a>	<a href="#">974, 987, 1002, 1029, 1054, 3013, 3048</a>
<code>\__graphics_backend_getbb_</code>	
auxiii:nNnn . . . . . <a href="#">1915</a>	<b>H</b>
<code>\__graphics_backend_getbb_</code>	hbox commands:
auxiv:nnNnn . . . . . <a href="#">1915</a>	<code>\hbox:n</code> . . . . . <a href="#">2140, 2143,</a>
<code>\__graphics_backend_getbb_</code>	<a href="#">2215, 2221, 2374, 2381, 2893, 2904</a>
auxv:nNnn . . . . . <a href="#">1915</a>	<code>\hbox_overlap_right:n</code> . . . . . <a href="#">223,</a>
<code>\__graphics_backend_getbb_</code>	<a href="#">255, 271, 312, 328, 356, 440, 1302, 1497</a>
auxvi:nNnn . . . . . <a href="#">1956, 1958</a>	<code>\hbox_set:Nn</code> . . <a href="#">1764, 1960, 2207, 2239</a>
<code>\__graphics_backend_getbb_eps:n</code> .	<code>\hbox_set:Nw</code> . . . . . <a href="#">2190</a>
. . . . . <a href="#">1697, 1779, 1820, 1978</a>	<code>\hbox_set_end:</code> . . . . . <a href="#">2205</a>
<code>\__graphics_backend_getbb_eps:nm</code>	<code>\hbox_unpack:N</code> . . . . . <a href="#">2326</a>
. . . . . <a href="#">1779</a>	
<code>\__graphics_backend_getbb_eps:nn</code>	<b>I</b>
. . . . . <a href="#">1790, 1797</a>	int commands:
<code>\__graphics_backend_getbb_jpg:n</code> .	<code>\int_compare:nNnTF</code> . . . . .
. . . . . <a href="#">1712, 1820, 1915, 1979</a>	<a href="#">508, 552, 650, 955, 997, 1734, 1759,</a>
<code>\__graphics_backend_getbb_</code>	<a href="#">1861, 1888, 1930, 1970, 2298, 2399,</a>
pagebox:w . . . . . <a href="#">1915, 1972</a>	<a href="#">2689, 2717, 2830, 2837, 2853, 3054</a>
<code>\__graphics_backend_getbb_pdf:n</code> .	<code>\int_const:Nn</code> . . . . . <a href="#">149, 155, 515,</a>
. . . . . <a href="#">1712, 1805, 1820, 1915, 1986</a>	<a href="#">543, 578, 1768, 1883, 2038, 2567, 2755</a>
<code>\__graphics_backend_getbb_png:n</code> .	<code>\int_eval:n</code> . . . . . <a href="#">559, 569, 598, 609,</a>
. . . . . <a href="#">1712, 1820, 1915, 1979</a>	<a href="#">713, 722, 735, 737, 741, 754, 2667,</a>
	<a href="#">2692, 2699, 2712, 2922, 2930, 2935</a>

`\int_gincr:N` ..... 197, 363,  
 514, 1564, 1609, 1882, 2037, 2106,  
 2150, 2224, 2754, 2797, 2810, 2832  
`\int_gset:Nn` ..... 172, 191, 2287  
`\int_gset_eq:NN` 180, 2151, 2225, 2811  
`\int_if_exist:NTF` ..... 1872  
`\int_if_odd:nTF` ..... 2210  
`\int_new:N` ..... 163, 164,  
 410, 505, 511, 1590, 1836, 2033,  
 2131, 2162, 2164, 2750, 2807, 2823  
`\int_set:Nn` ..... 535  
`\int_set_eq:NN` ... 168, 187, 541, 2299  
`\int_step_function:nnnN` ..... 739  
`\int_use:N` ..... 365,  
 396, 525, 536, 687, 823, 868, 941,  
 1567, 1573, 1580, 1612, 1620, 1735,  
 1760, 1775, 1862, 1875, 1887, 1889,  
 1971, 2044, 2109, 2122, 2126, 2154,  
 2161, 2229, 2330, 2578, 2588, 2761,  
 2799, 2804, 2814, 2822, 2840, 2856  
`\int_value:w` .....  
 ..... 2446, 2481, 2638, 2649, 2667  
`\int_zero:N` ... 1714, 1824, 1917, 1981

## K

kernel internal commands:

`\__kernel_backend_align_begin:` ..  
 ..... 64, 208, 232, 247  
`\__kernel_backend_align_end:` ...  
 ..... 64, 222, 240, 254  
`\g_kernel_backend_header_bool` ..  
 ..... 57, 669  
`\__kernel_backend_literal:n` ....  
 ..... 46, 52, 55, 62,  
 66, 73, 76, 78, 134, 137, 139, 141,  
 161, 337, 350, 522, 547, 548, 556,  
 566, 621, 628, 654, 660, 683, 819,  
 1001, 1007, 1009, 1028, 1053, 1120,  
 1126, 1421, 1428, 1434, 1494, 1499,  
 1700, 1839, 1874, 1884, 1998, 2009,  
 2744, 2860, 2922, 2926, 2931, 2936  
`\__kernel_backend_literal_page:n`  
 ..... 92, 136, 2738, 2740, 2941, 2943  
`\__kernel_backend_literal_pdf:n` .  
 .. 81, 133, 263, 320, 1311, 3063, 3078  
`\__kernel_backend_literal_-  
 postscript:n` .....  
 ..... 51, 67, 68, 72, 209, 210, 212,  
 213, 221, 233, 248, 1116, 2401, 2413  
`\__kernel_backend_literal_svg:n` .  
 ..... 160, 167, 178, 186,  
 196, 364, 366, 383, 1505, 1678, 1689  
`\__kernel_backend_matrix:n` .....  
 ..... 120, 285, 306, 1411

`\__kernel_backend_postscript:n` ..  
 ..... 54, 623,  
 1032, 1034, 1036, 1040, 2027, 2083,  
 2098, 2140, 2146, 2183, 2215, 2222,  
 2226, 2240, 2268, 2312, 2319, 2325,  
 2333, 2340, 2374, 2381, 2976, 2985  
`\__kernel_backend_scope:n` .....  
 ..... 165, 393, 398, 1094, 1510, 3092  
`\__kernel_backend_scope_begin:` ..  
 ..... 75, 102, 138,  
 165, 207, 231, 246, 262, 279, 305,  
 319, 336, 349, 1317, 1489, 1509, 1676  
`\__kernel_backend_scope_begin:n` .  
 ..... 165, 385, 413, 426  
`\__kernel_backend_scope_end:` ...  
 . 75, 102, 138, 165, 224, 242, 256,  
 272, 299, 313, 329, 345, 357, 408,  
 422, 441, 545, 1318, 1501, 1512, 1690  
`\g_kernel_backend_scope_int` ...  
 163, 170, 172, 177, 181, 189, 191, 197  
`\l_kernel_backend_scope_int` ...  
 ..... 163, 169, 182, 188  
`\__kernel_color_backend_stack_-  
 init:Nnn` ..... 508, 576, 2992  
`\__kernel_color_backend_stack_-  
 pop:n` ..... 552, 590, 647, 3020  
`\__kernel_color_backend_stack_-  
 push:nn` .....  
 .. 552, 590, 643, 972, 985, 3011, 3046  
`\__kernel_dependency_version_-  
 check:Nn` ..... 1  
`\__kernel_dependency_version_-  
 check:nn` ..... 27, 29  
`\__kernel_kern:n` .....  
 ..... 2145, 2147, 2373, 2377,  
 2380, 2384, 2892, 2900, 2903, 2919  
`\c_kernel_sys_dvipdfmx_version_-  
 int` ..... 143, 508, 552,  
 650, 955, 997, 2830, 2837, 2853, 3054

## M

math commands:

`\c_math_toggle_token` .... 2193, 2203  
`\MessageBreak` ..... 40

mode commands:

`\mode_if_horizontal:TF` ... 2289, 2296  
`\mode_if_math:TF` ..... 2187

## O

`\oddsidemargin` ..... 2211

opacity internal commands:

`\__opacity_backend:nn` ... 2979, 3085  
`\__opacity_backend_fill:n` .....  
 ..... 2979, 3021, 3085

\\_opacity\_backend\_fill\_stroke:nn  
     ..... [3023](#), [3029](#), [3033](#), [3051](#), [3065](#)  
 \l\_opacity\_backend\_fill\_tl ....  
     .. [2997](#), [3006](#), [3030](#), [3038](#), [3058](#), [3070](#)  
 \\_opacity\_backend\_fillstroke:nn  
     ..... [3021](#)  
 \\_opacity\_backend\_reset: [2999](#), [3048](#)  
 \\_opacity\_backend\_select:n ....  
     ..... [2969](#), [2999](#), [3054](#), [3085](#)  
 \\_opacity\_backend\_select\_aux:n .  
     ..... [2969](#), [2999](#), [3036](#), [3056](#), [3068](#)  
 \c\_opacity\_backend\_stack\_int ...  
     ..... [2990](#), [3011](#), [3020](#), [3046](#)  
 \\_opacity\_backend\_stroke:n ....  
     ..... [2979](#), [3021](#), [3085](#)  
 \l\_opacity\_backend\_stroke\_tl ...  
     .. [2997](#), [3007](#), [3025](#), [3039](#), [3059](#), [3071](#)

## P

pdf commands:

\pdf\_object\_if\_exist:nTF ..... [879](#)  
 \pdf\_object\_new:nn ..... [881](#)  
 \pdf\_object\_ref:n ..... [894](#)  
 \pdf\_object\_ref\_last: .....  
     ..... [860](#), [869](#), [933](#), [942](#)  
 \pdf\_object\_unnamed\_write:nn ...  
     ..... [848](#), [875](#), [899](#), [906](#)  
 \pdf\_object\_write:nn ..... [882](#)

pdf internal commands:

\\_pdf\_backend:n ..... [2743](#),  
     [2747](#), [2749](#), [2775](#), [2780](#), [2789](#), [2812](#),  
     [2834](#), [2850](#), [2863](#), [2895](#), [2896](#), [2906](#)  
 \\_pdf\_backend\_annotation:nnnn ..  
     ..... [2132](#), [2431](#), [2808](#)  
 \\_pdf\_backend\_annotation\_-  
     aux:nnnn ..... [2134](#), [2137](#)  
 \g\_pdf\_backend\_annotation\_int ..  
     .. [2131](#), [2151](#), [2161](#), [2807](#), [2811](#), [2822](#)  
 \\_pdf\_backend\_annotation\_last: .  
     ..... [2160](#), [2444](#), [2821](#)  
 \\_pdf\_backend\_bdc:nn .....  
     ..... [2425](#), [2737](#), [2940](#), [2962](#)  
 \\_pdf\_backend\_catalog\_gput:nn ..  
     ..... [2029](#), [2537](#), [2746](#), [2946](#)  
 \\_pdf\_backend\_compress\_objects:n  
     ..... [2397](#), [2658](#), [2921](#), [2956](#)  
 \\_pdf\_backend\_compresslevel:n ..  
     ..... [2397](#), [2658](#), [2921](#), [2956](#)  
 \l\_pdf\_backend\_content\_box [2129](#),  
     [2190](#), [2214](#), [2217](#), [2219](#), [2248](#), [2259](#)  
 \\_pdf\_backend\_destination:nn ...  
     ..... [2338](#), [2500](#), [2861](#)  
 \\_pdf\_backend\_destination:nnnn .  
     ..... [2338](#), [2500](#), [2861](#)

\\_pdf\_backend\_destination\_-  
     aux:nnnn ..... [2338](#), [2861](#)  
 \\_pdf\_backend\_emc: .....  
     ..... [2425](#), [2737](#), [2940](#), [2962](#)  
 \\_pdf\_backend\_info\_gput:nn ....  
     ..... [2029](#), [2537](#), [2746](#), [2946](#)  
 \\_pdf\_backend\_link:nw ..... [2171](#)  
 \\_pdf\_backend\_link\_aux:nw ... [2171](#)  
 \\_pdf\_backend\_link\_begin:n .. [2824](#)  
 \\_pdf\_backend\_link\_begin:nnnw [2455](#)  
 \\_pdf\_backend\_link\_begin:nw ...  
     ..... [2172](#), [2174](#), [2175](#)  
 \\_pdf\_backend\_link\_begin\_aux:nw  
     ..... [2178](#), [2180](#)  
 \\_pdf\_backend\_link\_begin\_-  
     goto:nnw ..... [2171](#), [2455](#), [2824](#)  
 \\_pdf\_backend\_link\_begin\_-  
     user:nnw ..... [2171](#), [2455](#), [2824](#)  
 \g\_pdf\_backend\_link\_bool .....  
     ..... [2166](#), [2177](#), [2182](#), [2197](#), [2235](#)  
 \g\_pdf\_backend\_link\_dict\_tl ...  
     ..... [2163](#), [2185](#), [2230](#)  
 \\_pdf\_backend\_link\_end: .....  
     ..... [2171](#), [2455](#), [2824](#)  
 \\_pdf\_backend\_link\_end\_aux: . [2171](#)  
 \g\_pdf\_backend\_link\_int .....  
     ..... [2162](#), [2225](#),  
     [2229](#), [2330](#), [2823](#), [2832](#), [2840](#), [2856](#)  
 \\_pdf\_backend\_link\_last: .....  
     ..... [2329](#), [2479](#), [2851](#)  
 \\_pdf\_backend\_link\_margin:n ...  
     ..... [2331](#), [2490](#), [2859](#)  
 \g\_pdf\_backend\_link\_math\_bool ..  
     ..... [2165](#), [2188](#), [2189](#), [2192](#), [2202](#)  
 \\_pdf\_backend\_link\_minima: .. [2171](#)  
 \\_pdf\_backend\_link\_outerbox:n [2171](#)  
 \g\_pdf\_backend\_link\_sf\_int ....  
     ..... [2164](#), [2287](#), [2298](#), [2299](#)  
 \\_pdf\_backend\_link\_sf\_restore: [2171](#)  
 \\_pdf\_backend\_link\_sf\_save: . [2171](#)  
 \l\_pdf\_backend\_model\_box . [2130](#),  
     [2207](#), [2239](#), [2247](#), [2258](#), [2273](#), [2275](#)  
 \\_pdf\_backend\_objcompresslevel:n  
     ..... [2658](#)  
 \g\_pdf\_backend\_object\_int .....  
     ..... [2033](#), [2037](#), [2040](#),  
     [2106](#), [2109](#), [2122](#), [2126](#), [2150](#), [2151](#),  
     [2154](#), [2224](#), [2225](#), [2750](#), [2754](#), [2757](#),  
     [2797](#), [2799](#), [2804](#), [2810](#), [2811](#), [2814](#)  
 \\_pdf\_backend\_object\_last: ....  
     ..... [2125](#), [2636](#), [2803](#), [2948](#)  
 \\_pdf\_backend\_object\_new:nn ...  
     ..... [2035](#), [2558](#), [2752](#), [2948](#)



\__pdf_backend_object_now:nn . . .	pdf.brokenlink.dict . . . . .	3181
..... 2104, 2610, 2795, 2948	pdf.brokenlink.rect . . . . .	3181
\g_pdf_backend_object_prop . . .	pdf.brokenlink.skip . . . . .	3181
..... 2033, 2041, 2052, 2062,	pdf.count . . . . .	3319
2557, 2575, 2591, 2750, 2758, 2765	pdf.currentrect . . . . .	3319
\__pdf_backend_object_ref:n 2035,	pdf.cvs . . . . .	3103
2049, 2063, 2558, 2752, 2771, 2948	pdf.dest.anchor . . . . .	3146
\__pdf_backend_object_write:nn . .	pdf.dest.point . . . . .	3146
..... 2045, 2579, 2762, 2948	pdf.dest.x . . . . .	3146
\__pdf_backend_object_write:nnn 2762	pdf.dest.y . . . . .	3146
\__pdf_backend_object_write_-	pdf.dest2device . . . . .	3146
array:nn . . . . . 2045, 2762	pdf.dev.x . . . . .	3146
\__pdf_backend_object_write_-	pdf.dev.y . . . . .	3146
dict:nn . . . . . 2045, 2762	pdf.dvi.pt . . . . .	3103
\__pdf_backend_object_write_-	pdf.globaldict . . . . .	3100
fstream:nn . . . . . 2045, 2762	pdf.leftboundary . . . . .	3181
\__pdf_backend_object_write_-	pdf.link.dict . . . . .	2171
fstream:nnn . . . . . 2079, 2081	pdf.linkdp.pad . . . . .	2171, 3107
\__pdf_backend_object_write_-	pdf.linkht.pad . . . . .	2171, 3107
stream:nn . . . . . 2045, 2762	pdf.linkmargin . . . . .	3107
\__pdf_backend_object_write_-	pdf.llx . . . . .	2171, 3110
stream:nnn . . . . . 2045	pdf.lly . . . . .	2171, 3110
\__pdf_backend_object_write_-	pdf.originx . . . . .	3181
stream:nnnn . . . . . 2762	pdf.originy . . . . .	3181
\__pdf_backend_pageobject_ref:n .	pdf.outerbox . . . . .	2171, 3423
..... 2127, 2647, 2805, 2948	pdf.pdfmark . . . . .	3423
\__pdf_backend_pdfmark:n . . . . .	pdf.pdfmark.dict . . . . .	3423
2026, 2030, 2032, 2047, 2068, 2073,	pdf.pdfmark.good . . . . .	3423
2107, 2152, 2341, 2385, 2426, 2428	pdf.pt.dvi . . . . .	3103
\__pdf_backend_version_major: . . .	pdf.rect . . . . .	3110
.. 2423, 2714, 2930, 2931, 2938, 2960	pdf.rect.ht . . . . .	3103
\__pdf_backend_version_major_-	pdf.rightboundary . . . . .	3181
gset:n . . . . . 2421, 2686, 2928, 2958	pdf.save.linkll . . . . .	3110
\__pdf_backend_version_minor: . . .	pdf.save.linkur . . . . .	3110
.. 2423, 2714, 2935, 2936, 2938, 2960	pdf.save.ll . . . . .	3110
\__pdf_backend_version_minor_-	pdf.save.ur . . . . .	3110
gset:n . . . . . 2421, 2686, 2928, 2958	pdf.tmpa . . . . .	3146
\l_pdf_breaklink_pdfmark_tl . . .	pdf.tmpb . . . . .	3146
..... 2167, 2232, 2324	pdf.tmpc . . . . .	3146
\__pdf_breaklink_postscript:n . . .	pdf.tmpd . . . . .	3146
..... 2169, 2216, 2218, 2325	pdf.urx . . . . .	3110
\__pdf_breaklink_usebox:N . . . . .	pdf.ury . . . . .	2171, 3110
..... 2170, 2217, 2326	pdfmanagement commands:	
\__pdf_exp_not_i:nn . 2579, 2625, 2630	\pdfmanagement_add:nnn . . . . .	862,
\__pdf_exp_not_ii:nn 2579, 2626, 2631	866, 935, 939, 2990, 2994, 3008,	
\l_pdf_internal_box . . . . . 2024	3015, 3040, 3043, 3060, 3072, 3075	
pdf.baselineskip . . . . .	prg commands:	
2171, 3423	\prg_replicate:nn . . . . .	
pdf.bordertracking . . . . .	..... 176, 711, 732, 742, 912	
3181	prop commands:	
pdf.bordertracking.begin . . . . .	\prop_gput:Nnn . . . . .	2041, 2575, 2758
3181	\prop_item:Nn . 2052, 2062, 2591, 2765	
pdf.bordertracking.continue . . . . .	\prop_new:N . . . . .	2034, 2557, 2751
3181	\ProvidesExplFile . . . . .	2
pdf.bordertracking.end . . . . .		
3181		
pdf.bordertracking.endpage . . . . .		
3181		
pdf.breaklink . . . . .		
3319		
pdf.breaklink.write . . . . .		
3319		

<b>Q</b>	
quark commands:	
\q_stop . . . . .	144, 152
quark internal commands:	
\s__color_stop . . . . .	463, 466, 477, 480, 722, 723, 727, 731, 744, 747, 751, 755, 769, 913, 946, 950, 1057, 1059, 1080, 1082
\s__graphics_stop . . . . .	1969, 1974, 2014, 2018
<b>S</b>	
scan commands:	
\scan_stop: . . . . .	105, 114, 609, 2473, 2498, 2521, 2535, 2667, 2684, 2692, 2699, 2712
separation . . . . .	3097
skip commands:	
\skip_horizontal:n . . . .	225, 273, 330
str commands:	
\c_hash_str . . . .	396, 1573, 1580, 1620
\c_percent_str . . . .	1100, 1101, 1102
\str_case:nn . . . . .	918, 2111, 2618
\str_case:nnTF . . . . .	2345, 2509, 2868
\str_case_e:nn . . . . .	2051, 2590
\str_convert_pdfname:n . . . .	690, 859
\str_if_eq:nnTF . . . . .	482, 485, 488, 491, 3035, 3067
\str_new:N . . . . .	1781, 1782, 1783
\str_tail:N . . . . .	1792, 1813
sys commands:	
\sys_get_shell:nnNTF . . . . .	145
\sys_if_shell:TF . . . . .	1779
\sys_shell_now:n . . . . .	1801
sys internal commands:	
\l_sys_internal_tl . . . . .	147, 151
\__sys_tmp:w . . . . .	144, 151
<b>T</b>	
TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
\@cclv . . . . .	2308, 2310, 2318
\@makecol@hook . . . . .	2302
\current@color . . . . .	14, 453, 457, 463, 477
\special . . . . .	2
tex commands:	
\tex_baselineskip:D . . . . .	2279
\tex_endinput:D . . . . .	44
\tex_global:D . . . . .	2660, 2677, 2691, 2698, 2705
\tex_immediate:D . . . . .	1746, 2582, 2585, 2613, 2616
\tex_lua <sup>te</sup> xversion:D . . . .	2689, 2717
\tex_pdfannot:D . . . . .	2437
\tex_pdfcatalog:D . . . . .	2543
\tex_pdfcolorstack:D . . . . .	596, 607
\tex_pdfcolorstackinit:D . . . . .	584
\tex_pdfcompresslevel:D . . . . .	2665
\tex_pdftest:D . . . . .	2506, 2529
\tex_pdfendlink:D . . . . .	2476
\tex_pdfextension:D . . . . .	84, 95, 105, 114, 123, 593, 604, 2434, 2462, 2473, 2503, 2526, 2540, 2550, 2561, 2582, 2613
\tex_pdffeedback:D . . . . .	581, 2448, 2483, 2570, 2640, 2651
\tex_pdfinfo:D . . . . .	2553
\tex_pdflastannot:D . . . . .	2451
\tex_pdflastlink:D . . . . .	2486
\tex_pdflastobj:D . . . . .	2573, 2643
\tex_pdflastximage:D . . . .	1765, 1769
\tex_pdflinkmargin:D . . . . .	2496
\tex_pdfliteral:D . . . . .	87, 98
\tex_pdfmajorversion:D . . . . .	2696, 2698, 2722, 2723
\tex_pdfminorversion:D . . . .	2710, 2734
\tex_pdfobj:D . . . . .	2564, 2585, 2616
\tex_pdfobjcompresslevel:D . . . .	2682
\tex_pdfpageref:D . . . . .	2654
\tex_pdfrefximage:D . . . .	1765, 1774
\tex_pdfrestore:D . . . . .	117
\tex_pdfsave:D . . . . .	108
\tex_pdfsetmatrix:D . . . . .	126
\tex_pdfstartlink:D . . . . .	2465
\tex_pdfvariable:D . . . . .	2493, 2662, 2679, 2691, 2707, 2718, 2731
\tex_pdfximage:D . . . . .	1746
\tex_spacefactor:D . . . . .	2290, 2299
\tex_special:D . . . . .	46
\tex_the:D . . . .	1769, 2718, 2723, 2729
\tex_vss:D . . . .	2375, 2382, 2898, 2917
\tex_XeTeXpdf <sup>file</sup> :D . . . . .	1926, 1968
\tex_XeTeXpic <sup>file</sup> :D . . . . .	1919
TeXcolorseparation . . . . .	3097
\textwidth . . . . .	2274
tl commands:	
\c_space_tl . . . .	287, 292, 295, 526, 778, 986, 1549, 1702, 1703, 1704, 1705, 1841, 1842, 1843, 1844, 1889, 1892, 1894, 1895, 1896, 1897, 1969, 1971, 2000, 2001, 2002, 2003, 2230, 2453, 2488, 2645, 2656, 2814, 2841
\tl_clear:N . . . .	1715, 1723, 1729, 1825, 1831, 1918, 1924, 1982, 1988
\tl_gclear:N . . . . .	1587, 1623
\tl_gset:Nn . . . . .	1546, 2185
\tl_if_blank:nTF . . . . .	527, 586, 726, 743, 750, 768, 852, 949

\tl_if_empty:NTF .	1549, 1718, 1753, 1761, 1859, 1863, 1890, 1905, 1939	
\tl_if_empty:nTF	1643	
\tl_if_empty_p:N	1749, 1902	
\tl_if_head_is_space:nTF	453	
\tl_new:N	631, 632, 1553, 1711, 2163, 2167, 2997, 2998	
\tl_put_right:Nn	2306	
\tl_set:Nn .	455, 467, 483, 486, 489, 493, 496, 641, 642, 971, 984, 1716, 1731, 1804, 2168, 2324, 3006, 3007, 3038, 3039, 3058, 3059, 3070, 3071	
\tl_to_str:n	2039, 2044, 2568, 2578, 2589, 2756, 2761	
\tl_use:N	810, 887	
		<b>U</b>
		use commands:
\use:N	43, 2061, 2121, 2770, 2798	
\use:n .	61, 457, 493, 516, 520, 673, 864, 937, 1061, 1071, 1084, 1257, 1381, 1446, 1458, 1470, 1628, 1946	
\use_none:n	1643, 1645, 2302	
		<b>V</b>
\value	2210	
		vbox commands:
\vbox_set:Nn	2310	
\vbox_to_zero:n	2371, 2378, 2890, 2901	
\vbox_unpack_drop:N	2318	