

# 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}{2022-01-12}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2022-01-12}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2022-01-12}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2022-01-12}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2022-01-12}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2022-01-12}{ }
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:n
28   {
29     \_kernel_dependency_version_check:n {2021-02-18}
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`.)

`\_kernel_backend_first_shipout:n` We need to write at first shipout in a few places. As we want to use the most up-to-date method,

```

50 \cs_if_exist:NTF \@ifl@t@r
51 {
52   \@ifl@t@r \fmtversion { 2020-10-01 }
53   {
54     \cs_new_protected:Npn __kernel_backend_first_shipout:n #1
55     { \hook_gput_code:nnn { shipout / firstpage } { l3backend } {#1} }
56   }
57   { \cs_new_eq:NN __kernel_backend_first_shipout:n \AtBeginDvi }
58 }
59 { \cs_new_eq:NN __kernel_backend_first_shipout:n \use:n }

```

(End definition for `__kernel_backend_first_shipout:n`.)

## 1.1 dvips backend

```

60 <*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.

```

61 \cs_new_protected:Npn __kernel_backend_literal_postscript:n #1
62 { __kernel_backend_literal:n { ps:: #1 } }
63 \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 `"`).

`\_kernel_backend_postscript:x`

```

64 \cs_new_protected:Npn \_kernel_backend_postscript:n #1
65   { \_kernel_backend_literal:n { ps: SDict ~ begin ~ #1 ~ end } }
66 \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.

```

67 \bool_if:NT \g__kernel_backend_header_bool
68   {
69     \_kernel_backend_first_shipout:n
70     { \_kernel_backend_literal:n { header = l3backend-dvips.pro } }
71   }

```

`\_kernel_backend_align_begin:`

`\_kernel_backend_align_end:`

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.

```

72 \cs_new_protected:Npn \_kernel_backend_align_begin:
73   {
74     \_kernel_backend_literal:n { ps::[begin] }
75     \_kernel_backend_literal_postscript:n { currentpoint }
76     \_kernel_backend_literal_postscript:n { currentpoint-translate }
77   }
78 \cs_new_protected:Npn \_kernel_backend_align_end:
79   {
80     \_kernel_backend_literal_postscript:n { neg-exch-neg-exch-translate }
81     \_kernel_backend_literal:n { ps::[end] }
82   }

```

(End definition for `\_kernel_backend_align_begin:` and `\_kernel_backend_align_end:.`)

`\_kernel_backend_scope_begin:`

`\_kernel_backend_scope_end:`

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.

```

83 \cs_new_protected:Npn \_kernel_backend_scope_begin:
84   { \_kernel_backend_literal:n { ps:gsave } }
85 \cs_new_protected:Npn \_kernel_backend_scope_end:
86   { \_kernel_backend_literal:n { ps:grestore } }

```

(End definition for `\_kernel_backend_scope_begin:` and `\_kernel_backend_scope_end:.`)

```

87 </dvips>

```

## 1.2 LuaTeX and pdfTeX backends

88 `<*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  
\\_kernel\_backend\_literal\_pdf:x

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).

```
89 \cs_new_protected:Npn \_kernel_backend_literal_pdf:n #1
90 {
91   <*luatex>
92     \tex_pdfextension:D literal
93   </luatex>
94   <*pdftex>
95     \tex_pdfliteral:D
96   </pdftex>
97   { \exp_not:n {#1} }
98 }
99 \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.

```
100 \cs_new_protected:Npn \_kernel_backend_literal_page:n #1
101 {
102   <*luatex>
103     \tex_pdfextension:D literal ~
104   </luatex>
105   <*pdftex>
106     \tex_pdfliteral:D
107   </pdftex>
108     page { \exp_not:n {#1} }
109 }
```

(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:

```
110 \cs_new_protected:Npn \_kernel_backend_scope_begin:
111 {
112   <*luatex>
113     \tex_pdfextension:D save \scan_stop:
114   </luatex>
115   <*pdftex>
116     \tex_pdfsave:D
117   </pdftex>
118 }
119 \cs_new_protected:Npn \_kernel_backend_scope_end:
120 {
121   <*luatex>
122     \tex_pdfextension:D restore \scan_stop:
123   </luatex>
124   <*pdftex>
125     \tex_pdfrestore:D
```

```

126 </pdfTeX>
127 }

```

(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.

```

128 \cs_new_protected:Npn \_kernel_backend_matrix:n #1
129 {
130   <*luatex>
131   \tex_pdfextension:D setmatrix
132 </luatex>
133 <*pdfTeX>
134   \tex_pdfsetmatrix:D
135 </pdfTeX>
136   { \exp_not:n {#1} }
137 }
138 \cs_generate_variant:Nn \_kernel_backend_matrix:n { x }

```

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

```

139 </luatex | pdfTeX>

```

### 1.3 dvipdfmx backend

```

140 <*dvipdfmx | xetex>

```

The dvipdfmx shares code with the PDF mode one (using the common section to this file) but also with XeTeX. 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 XeTeX 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

```

```

141 \cs_new_protected:Npn \_kernel_backend_literal_pdf:n #1
142 { \_kernel_backend_literal:n { pdf:literal~ #1 } }
143 \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!

```

144 \cs_new_protected:Npn \_kernel_backend_literal_page:n #1
145 { \_kernel_backend_literal:n { pdf:literal~direct~ #1 } }

```

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

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

```

146 \cs_new_protected:Npn \_kernel_backend_scope_begin:
147 { \_kernel_backend_literal:n { x:gsave } }
148 \cs_new_protected:Npn \_kernel_backend_scope_end:
149 { \_kernel_backend_literal:n { x:grestore } }

```

(End definition for `\_kernel_backend_scope_begin:` and `\_kernel_backend_scope_end:.`)

```

150 <@@=sys>

```

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

```

151 \group_begin:
152 \cs_set:Npn \__sys_tmp:w #1 Version ~ #2 ~ #3 \q_stop {#2}
153 \sys_get_shell:nnNTF { extractbb~--version }
154 { \char_set_catcode_space:n { '\ } }
155 \l__sys_internal_tl
156 {
157   \int_const:Nn \c__kernel_sys_dvipdfmx_version_int
158   {
159     \exp_after:wN \__sys_tmp:w \l__sys_internal_tl
160     \q_stop
161   }
162 }
163 { \int_const:Nn \c__kernel_sys_dvipdfmx_version_int { 0 } }
164 \group_end:

```

(End definition for `\c__kernel_sys_dvipdfmx_version_int`.)

```

165 <@@=
166 </dvipdfmx | xetex>

```

## 1.4 dvisvgm backend

```

167 <*dvisvgm>

```

`\__kernel_backend_literal_svg:n` 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.

`\__kernel_backend_literal_svg:x`

```

168 \cs_new_protected:Npn \__kernel_backend_literal_svg:n #1
169 { \__kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }
170 \cs_generate_variant:Nn \__kernel_backend_literal_svg:n { x }

```

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

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

`\l__kernel_backend_scope_int`

```

171 \int_new:N \g__kernel_backend_scope_int
172 \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:` 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.

`\__kernel_backend_scope_end:`

`\__kernel_backend_scope_begin:n`

`\__kernel_backend_scope_begin:x`

```

173 \cs_new_protected:Npn \__kernel_backend_scope_begin:
174 {
175   \__kernel_backend_literal_svg:n { <g> }
176   \int_set_eq:NN
177   \l__kernel_backend_scope_int
178   \g__kernel_backend_scope_int
179   \group_begin:
180   \int_gset:Nn \g__kernel_backend_scope_int { 1 }

```

```

181 }
182 \cs_new_protected:Npn \__kernel_backend_scope_end:
183 {
184     \prg_replicate:nn
185     { \g__kernel_backend_scope_int }
186     { \__kernel_backend_literal_svg:n { </g> } }
187     \group_end:
188     \int_gset_eq:NN
189     \g__kernel_backend_scope_int
190     \l__kernel_backend_scope_int
191 }
192 \cs_new_protected:Npn \__kernel_backend_scope_begin:n #1
193 {
194     \__kernel_backend_literal_svg:n { <g ~ #1 > }
195     \int_set_eq:NN
196     \l__kernel_backend_scope_int
197     \g__kernel_backend_scope_int
198     \group_begin:
199     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
200 }
201 \cs_generate_variant:Nn \__kernel_backend_scope_begin:n { x }
202 \cs_new_protected:Npn \__kernel_backend_scope:n #1
203 {
204     \__kernel_backend_literal_svg:n { <g ~ #1 > }
205     \int_gincr:N \g__kernel_backend_scope_int
206 }
207 \cs_generate_variant:Nn \__kernel_backend_scope:n { x }

```

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

```

208 </dvisvgm>
209 </package>

```

## 2 l3backend-box Implementation

```

210 <*package>
211 <@@=box>

```

### 2.1 dvips backend

```

212 <*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.

```

213 \cs_new_protected:Npn \__box_backend_clip:N #1
214 {
215     \__kernel_backend_scope_begin:
216     \__kernel_backend_align_begin:
217     \__kernel_backend_literal_postscript:n { matrix~currentmatrix }
218     \__kernel_backend_literal_postscript:n
219     { Resolution~72~div~VResolution~72~div~scale }

```

```

220 \__kernel_backend_literal_postscript:n { DVImag-dup~scale }
221 \__kernel_backend_literal_postscript:x
222 {
223   0 ~
224   \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
225   \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
226   \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
227   rectclip
228 }
229 \__kernel_backend_literal_postscript:n { setmatrix }
230 \__kernel_backend_align_end:
231 \hbox_overlap_right:n { \box_use:N #1 }
232 \__kernel_backend_scope_end:
233 \skip_horizontal:n { \box_wd:N #1 }
234 }

```

(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.

```

235 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
236 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
237 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
238 {
239   \__kernel_backend_scope_begin:
240   \__kernel_backend_align_begin:
241   \__kernel_backend_literal_postscript:x
242   {
243     \fp_compare:nNnTF {#2} = \c_zero_fp
244     { 0 }
245     { \fp_eval:n { round ( -(#2) , 5 ) } } ~
246     rotate
247   }
248   \__kernel_backend_align_end:
249   \box_use:N #1
250   \__kernel_backend_scope_end:
251 }

```

(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.

```

252 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
253 {
254   \__kernel_backend_scope_begin:
255   \__kernel_backend_align_begin:
256   \__kernel_backend_literal_postscript:x
257   {
258     \fp_eval:n { round ( #2 , 5 ) } ~
259     \fp_eval:n { round ( #3 , 5 ) } ~
260     scale
261   }
262   \__kernel_backend_align_end:
263   \hbox_overlap_right:n { \box_use:N #1 }

```

```

264     \__kernel_backend_scope_end:
265 }

(End definition for \__box_backend_scale:Nnn.)

266 </dvips>

```

## 2.2 LuaTeX and pdfTeX backends

```

267 <*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.

```

268 \cs_new_protected:Npn \__box_backend_clip:N #1
269 {
270     \__kernel_backend_scope_begin:
271     \__kernel_backend_literal_pdf:x
272     {
273         0~
274         \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
275         \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
276         \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
277         re~W~n
278     }
279     \hbox_overlap_right:n { \box_use:N #1 }
280     \__kernel_backend_scope_end:
281     \skip_horizontal:n { \box_wd:N #1 }
282 }

```

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

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

```

283 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
284 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
285 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
286 {
287     \__kernel_backend_scope_begin:
288     \box_set_wd:Nn #1 { Opt }
289     \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
290     \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
291     { \fp_zero:N \l__box_backend_cos_fp }
292     \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
293     \__kernel_backend_matrix:x
294     {
295         \fp_use:N \l__box_backend_cos_fp \c_space_tl
296         \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp

```

```

297         { 0~0 }
298         {
299             \fp_use:N \l__box_backend_sin_fp
300             \c_space_tl
301             \fp_eval:n { -\l__box_backend_sin_fp }
302         }
303         \c_space_tl
304         \fp_use:N \l__box_backend_cos_fp
305     }
306     \box_use:N #1
307     \__kernel_backend_scope_end:
308 }
309 \fp_new:N \l__box_backend_cos_fp
310 \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.

```

311 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
312 {
313     \__kernel_backend_scope_begin:
314     \__kernel_backend_matrix:x
315     {
316         \fp_eval:n { round ( #2 , 5 ) } ~
317         0~0~
318         \fp_eval:n { round ( #3 , 5 ) }
319     }
320     \hbox_overlap_right:n { \box_use:N #1 }
321     \__kernel_backend_scope_end:
322 }

```

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

323 </luatex | pdftex>

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

324 <\*dvipdfmx | xetex>

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

```

325 \cs_new_protected:Npn \__box_backend_clip:N #1
326 {
327     \__kernel_backend_scope_begin:
328     \__kernel_backend_literal_pdf:x
329     {
330         0~
331         \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
332         \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
333         \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
334         re~W~n
335     }
336     \hbox_overlap_right:n { \box_use:N #1 }
337     \__kernel_backend_scope_end:
338     \skip_horizontal:n { \box_wd:N #1 }
339 }

```

(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.

`\_box_backend_rotate_aux:Nn`

```

340 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
341 { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
342 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
343 {
344   \_kernel_backend_scope_begin:
345   \_kernel_backend_literal:x
346   {
347     x:rotate~
348     \fp_compare:nNnTF {#2} = \c_zero_fp
349     { 0 }
350     { \fp_eval:n { round ( #2 , 5 ) } }
351   }
352   \box_use:N #1
353   \_kernel_backend_scope_end:
354 }

```

(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.

```

355 \cs_new_protected:Npn \_box_backend_scale:Nnn #1#2#3
356 {
357   \_kernel_backend_scope_begin:
358   \_kernel_backend_literal:x
359   {
360     x:scale~
361     \fp_eval:n { round ( #2 , 5 ) } ~
362     \fp_eval:n { round ( #3 , 5 ) }
363   }
364   \hbox_overlap_right:n { \box_use:N #1 }
365   \_kernel_backend_scope_end:
366 }

```

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

367 `</dvipdfmx | xetex>`

## 2.4 dvisvgm backend

368 `<*dvisvgm>`

`\_box_backend_clip:N`  
`\g_kernel_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  $\text{\TeX}$  box and keep the reference point the same!

```

369 \cs_new_protected:Npn \__box_backend_clip:N #1
370 {
371   \int_gincr:N \g__kernel_clip_path_int
372   \__kernel_backend_literal_svg:x
373   { < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " > }
374   \__kernel_backend_literal_svg:x
375   {
376     <
377     path ~ d =
378     "
379       M ~ 0 ~
380       \dim_to_decimal:n { -\box_dp:N #1 } ~
381       L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
382       \dim_to_decimal:n { -\box_dp:N #1 } ~
383       L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
384       \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
385       L ~ 0 ~
386       \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
387       Z
388     "
389     />
390   }
391   \__kernel_backend_literal_svg:n
392   { < /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.

```

393 \__kernel_backend_scope_begin:n
394 {
395   transform =
396   "
397     translate ( { ?x } , { ?y } ) ~
398     scale ( 1 , -1 )
399   "
400 }
401 \__kernel_backend_scope:x
402 {
403   clip-path =
404   "url ( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int ) "
405 }
406 \__kernel_backend_scope:n
407 {
408   transform =
409   "
410     scale ( -1 , 1 ) ~
411     translate ( { ?x } , { ?y } ) ~
412     scale ( -1 , -1 )
413   "
414 }

```

```

415     \box_use:N #1
416     \__kernel_backend_scope_end:
417   }
418   \int_new:N \g__kernel_clip_path_int
(End definition for \__box_backend_clip:N and \g__kernel_clip_path_int.)

```

`\__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.

```

419   \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
420   {
421     \__kernel_backend_scope_begin:x
422     {
423       transform =
424       "
425         rotate
426         ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
427       "
428     }
429     \box_use:N #1
430     \__kernel_backend_scope_end:
431   }
(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).

```

432   \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
433   {
434     \__kernel_backend_scope_begin:x
435     {
436       transform =
437       "
438         translate ( { ?x } , { ?y } ) ~
439         scale
440         (
441           \fp_eval:n { round ( -#2 , 5 ) } ,
442           \fp_eval:n { round ( -#3 , 5 ) }
443         ) ~
444         translate ( { ?x } , { ?y } ) ~
445         scale ( -1 )
446       "
447     }
448     \hbox_overlap_right:n { \box_use:N #1 }
449     \__kernel_backend_scope_end:
450   }
(End definition for \__box_backend_scale:Nnn.)

```

```

451   </dvisvgm>
452   </package>

```

### 3 l3backend-color Implementation

```

453 <*package>
454 <@@=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

```

455 <*dvisvgm | dvipdfmx | dvips | xetex>

```

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). The x-type expansion is there to cover the case where xcolor is in use.

```

456 \cs_new_protected:Npn \__color_backend_pickup:N #1 { }
457 \cs_if_exist:cT { ver@color.sty }
458 {
459   \cs_set_protected:Npn \__color_backend_pickup:N #1
460   {
461     \exp_args:NW \tl_if_head_is_space:nTF \current@color
462     {
463       \tl_set:Nx #1
464       {
465         { \exp_after:wN \use:n \current@color }
466         { 1 }
467       }
468     }
469     {
470       \exp_last_unbraced:Nx \__color_backend_pickup:w
471       { \current@color } \s__color_stop #1
472     }
473   }
474   \cs_new_protected:Npn \__color_backend_pickup:w #1 ~ #2 \s__color_stop #3
475   { \tl_set:Nn #3 { {#1} {#2} } }
476 }

```

(End definition for \\_\_color\_backend\_pickup:N and \\_\_color\_backend\_pickup:w.)

```

477 </dvisvgm | dvipdfmx | dvips | xetex>

```

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

```

478 <*luatex | pdftex>

```

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

```

479 \cs_new_protected:Npn \__color_backend_pickup:N #1 { }
480 \cs_if_exist:cT { ver@color.sty }

```

```

481 {
482   \cs_set_protected:Npn \__color_backend_pickup:N #1
483   {
484     \exp_last_unbraced:Nx \__color_backend_pickup:w
485     { \current@color } ~ 0 ~ 0 ~ 0 \s__color_stop #1
486   }
487   \cs_new_protected:Npn \__color_backend_pickup:w
488   #1 ~ #2 ~ #3 ~ #4 ~ #5 ~ #6 \s__color_stop #7
489   {
490     \str_if_eq:nnTF {#2} { g }
491     { \tl_set:Nn #7 { { gray } {#1} } }
492     {
493       \str_if_eq:nnTF {#4} { rg }
494       { \tl_set:Nn #7 { { rgb } { #1 ~ #2 ~ #3 } } }
495       {
496         \str_if_eq:nnTF {#5} { k }
497         { \tl_set:Nn #7 { { cmyk } { #1 ~ #2 ~ #3 ~ #4 } } }
498         {
499           \str_if_eq:nnTF {#2} { cs }
500           {
501             \tl_set:Nx #7 { { \use:n #1 } { #5 } }
502           }
503           {
504             \tl_set:Nn #7 { { gray } { 0 } }
505           }
506         }
507       }
508     }
509   }
510 }

```

(End definition for \\_\_color\_backend\_pickup:N and \\_\_color\_backend\_pickup:w.)

511 </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/X<sub>Y</sub>TeX the backend version.

### 3.2.1 Common code

512 <\*dvipdfmx | luatex | pdftex | xetex>

\l\_\_color\_backend\_stack\_int pdfTeX, LuaTeX and recent (x)dvipdfmx have multiple stacks available, and to track which one is in use a variable is required.

513 \int\_new:N \l\_\_color\_backend\_stack\_int

(End definition for \l\_\_color\_backend\_stack\_int.)

514 </dvipdfmx | luatex | pdftex | xetex>

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

515  $\langle *dvipdfmx | xetex \rangle$

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.

```

516 \int_compare:nNnTF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
517 { \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3 { } }
518 {
519   \int_new:N \g__color_backend_stack_int
520   \cs_new_protected:Npx \__kernel_color_backend_stack_init:Nnn #1#2#3
521   {
522     \int_gincr:N \exp_not:N \g__color_backend_stack_int
523     \int_const:Nn #1 { \exp_not:N \g__color_backend_stack_int }
524     \use:x
525     {
526       \__kernel_backend_first_shipout:n
527       {
528         \__kernel_backend_literal:n
529         {
530           pdfcolorstackinit ~
531           \exp_not:N \int_use:N \exp_not:N \g__color_backend_stack_int
532           \c_space_tl
533           \exp_not:N \tl_if_blank:nF {#2} { #2 ~ }
534           (#3)
535         }
536       }
537     }
538   }
539   \cs_if_exist:cTF { main@pdfcolorstack }
540   {
541     \int_set:Nn \l__color_backend_stack_int
542     { \int_use:c { main@pdfcolorstack } }
543   }
544   {
545     \__kernel_color_backend_stack_init:Nnn \c__color_backend_main_stack_int
546     { page ~ direct } { 0 ~ g ~ 0 ~ G }
547     \int_set_eq:NN \l__color_backend_stack_int
548     \c__color_backend_main_stack_int
549     \int_const:cn { main@pdfcolorstack } { \c__color_backend_main_stack_int }
550   }

```

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.

```

551   \cs_gset_protected:Npn \__kernel_backend_scope_end:
552   {
553     \__kernel_backend_literal:n { x:grestore }
554     \__kernel_backend_literal:n
555     { pdfcolorstack ~ \g__color_backend_stack_int current }
556   }
557 }

```

(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.

```

558 \int_compare:nNnF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
559 {
560   \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
561   {
562     \__kernel_backend_literal:x
563     {
564       pdfcolorstack ~
565       \int_eval:n {#1} ~
566       push ~ (#2)
567     }
568   }
569   \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
570   \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
571   {
572     \__kernel_backend_literal:x
573     {
574       pdfcolorstack ~
575       \int_eval:n {#1} ~
576       pop
577     }
578   }
579 }

```

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

```

580 </dvipdfmx | xetex>

```

### 3.2.3 LuaTeX and pdfTeX

```

581 <*luatex | pdftex>

```

```

\__kernel_color_backend_stack_init:Nnn

```

```

582 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
583 {
584   \int_const:Nn #1
585   {
586     <*luatex>
587     \tex_pdffeedback:D colorstackinit ~
588     </luatex>
589     <*pdftex>
590     \tex_pdfcolorstackinit:D
591     </pdftex>
592     \tl_if_blank:nF {#2} { #2 ~ }
593     {#3}
594   }
595 }

```

(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

```

```

596 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
597 {
598   <*luatex>

```

```

599     \tex_pdfextension:D colorstack ~
600   </luatex>
601   <*pdfTex>
602     \tex_pdfcolorstack:D
603   </pdfTex>
604     \int_eval:n {#1} ~ push ~ {#2}
605   }
606   \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
607   \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
608   {
609     <luatex>
610     \tex_pdfextension:D colorstack ~
611   </luatex>
612   <*pdfTex>
613     \tex_pdfcolorstack:D
614   </pdfTex>
615     \int_eval:n {#1} ~ pop \scan_stop:
616   }

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

617 </luatex | pdfTex>

```

### 3.3 General color

#### 3.3.1 dvips-style

```

618 <*dvips | dvisvgm>

\__color_backend_select_cmyk:n Push the data to the stack. In the case of dvips also saves the drawing color in raw
\__color_backend_select_gray:n PostScript.
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc
619 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
620 { \__color_backend_select:n { cmyk ~ #1 } }
621 \cs_new_protected:Npn \__color_backend_select_gray:n #1
622 { \__color_backend_select:n { gray ~ #1 } }
623 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
624 { \__color_backend_select:n { rgb ~ #1 } }
625 \cs_new_protected:Npn \__color_backend_select:n #1
626 {
627   \__kernel_backend_literal:n { color~push~ #1 }
628 <*dvips>
629   \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
630 </dvips>
631 }
632 \cs_new_protected:Npn \__color_backend_reset:
633 { \__kernel_backend_literal:n { color~pop } }

(End definition for \__color_backend_select_cmyk:n and others. This function is documented on page
??.)

634 </dvips | dvisvgm>

```

### 3.3.2 LuaTeX and pdfTeX

635  $\langle$ \*dvipdfmx | luatex | pdftex | xetex $\rangle$

\l\_\_color\_backend\_fill\_tl  
\l\_\_color\_backend\_stroke\_tl

636 \tl\_new:N \l\_\_color\_backend\_fill\_tl  
637 \tl\_new:N \l\_\_color\_backend\_stroke\_tl

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

\\_color\_backend\_select\_cmyk:n  
\\_color\_backend\_select\_gray:n  
\\_color\_backend\_select\_rgb:n  
\\_\_color\_backend\_select:nn  
\\_\_color\_backend\_reset:

Store the values then pass to the stack.

638 \cs\_new\_protected:Npn \\_color\_backend\_select\_cmyk:n #1  
639 { \\_color\_backend\_select:nn { #1 ~ k } { #1 ~ K } }  
640 \cs\_new\_protected:Npn \\_color\_backend\_select\_gray:n #1  
641 { \\_color\_backend\_select:nn { #1 ~ g } { #1 ~ G } }  
642 \cs\_new\_protected:Npn \\_color\_backend\_select\_rgb:n #1  
643 { \\_color\_backend\_select:nn { #1 ~ rg } { #1 ~ RG } }  
644 \cs\_new\_protected:Npn \\_color\_backend\_select:nn #1#2  
645 {  
646 \tl\_set:Nn \l\_\_color\_backend\_fill\_tl {#1}  
647 \tl\_set:Nn \l\_\_color\_backend\_stroke\_tl {#2}  
648 \\_\_kernel\_color\_backend\_stack\_push:nn \l\_\_color\_backend\_stack\_int { #1 ~ #2 }  
649 }  
650 \cs\_new\_protected:Npn \\_color\_backend\_reset:  
651 { \\_\_kernel\_color\_backend\_stack\_pop:n \l\_\_color\_backend\_stack\_int }

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

652  $\langle$ /dvipdfmx | luatex | pdftex | xetex $\rangle$

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

653  $\langle$ \*dvipdfmx | xetex $\rangle$

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).

\\_color\_backend\_select\_cmyk:n  
\\_color\_backend\_select\_gray:n  
\\_color\_backend\_select\_rgb:n  
\\_\_color\_backend\_reset:

Push the data to the stack.

654 \int\_compare:nNnT \c\_\_kernel\_sys\_dvipdfmx\_version\_int < { 20201111 }  
655 {  
656 \cs\_gset\_protected:Npn \\_color\_backend\_select\_cmyk:n #1  
657 { \\_\_kernel\_backend\_literal:n { pdf: bc ~ [#1] } }  
658 \cs\_gset\_eq:NN \\_color\_backend\_select\_gray:n \\_color\_backend\_select\_cmyk:n  
659 \cs\_gset\_eq:NN \\_color\_backend\_select\_rgb:n \\_color\_backend\_select\_cmyk:n  
660 \cs\_gset\_protected:Npn \\_color\_backend\_reset:  
661 { \\_\_kernel\_backend\_literal:n { pdf: ec } }  
662 }

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

663  $\langle$ /dvipdfmx | xetex $\rangle$

### 3.4 Separations

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

```
664 <*dvipdfmx | luatex | pdftex | xetex | dvips>
```

But we start with some functionality needed for both PostScript and PDF based backends.

```
\g_color_backend_colorant_prop
```

```
665 \prop_new:N \g__color_backend_colorant_prop
```

(End definition for \g\_\_color\_backend\_colorant\_prop.)

```
\_color_backend_devicen_colorants:n
```

```
\_color_backend_devicen_colorants:w
```

```
666 \cs_new:Npx \_color_backend_devicen_colorants:n #1
667 {
668   \exp_not:N \tl_if_blank:nF {#1}
669   {
670     \c_space_tl
671     << ~
672     /Colorants ~
673     << ~
674     \exp_not:N \_color_backend_devicen_colorants:w #1 ~
675     \exp_not:N \q_recursion_tail \c_space_tl
676     \exp_not:N \q_recursion_stop
677     >> ~
678   }
679 }
680 }
681 \cs_new:Npn \_color_backend_devicen_colorants:w #1 ~
682 {
683   \quark_if_recursion_tail_stop:n {#1}
684   \prop_if_in:NnT \g__color_backend_colorant_prop {#1}
685   {
686     #1 ~
687     \prop_item:Nn \g__color_backend_colorant_prop {#1} ~
688   }
689   \_color_backend_devicen_colorants:w
690 }
```

(End definition for \\_color\_backend\_devicen\_colorants:n and \\_color\_backend\_devicen\_colorants:w.)

```
691 </dvipdfmx | luatex | pdftex | xetex | dvips>
```

```
692 <*dvips>
```

```
\_color_backend_select_separation:nn
```

```
\_color_backend_select_devicen:nn
```

```
693 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
694 { \_color_backend_select:n { separation ~ #1 ~ #2 } }
695 \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_select_iccbased:nn
```

No support.

```
696 \cs_new_protected:Npn \_color_backend_select_iccbased:nn #1#2 { }
```

(End definition for `\_color_backend_select_iccbased:nn`.)

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.

```

\_color_backend_separation_init:nnnnn
\_color_backend_separation_init:nxxnn
\_color_backend_separation_init_aux:nnnnnn
\_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:nnnnn
    \_color_backend_separation_init:w
    \_color_backend_separation_init:n
    \_color_backend_separation_init:nw
\_color_backend_separation_init_CIELAB:nnn
697 \cs_new_protected:Npx \_color_backend_separation_init:nnnnn #1#2#3#4#5
698 {
699   \bool_if:NT \g__kernel_backend_header_bool
700   {
701     \exp_args:Nx \_kernel_backend_first_shipout:n
702     {
703       \exp_not:N \_color_backend_separation_init_aux:nnnnnn
704       { \exp_not:N \int_use:N \g__color_model_int }
705       {#1} {#2} {#3} {#4} {#5}
706     }
707     \prop_gput:Nxx \exp_not:N \g__color_backend_colorant_prop
708     { / \exp_not:N \str_convert_pdfname:n {#1} }
709     {
710       << ~
711       /setcolorspace ~ {} ~
712       >> ~ begin ~
713       color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
714       end
715     }
716   }
717 }
718 \cs_generate_variant:Nn \_color_backend_separation_init:nnnnn { nxx }
719 \cs_new_protected:Npn \_color_backend_separation_init_aux:nnnnnn #1#2#3#4#5#6
720 {
721   \_kernel_backend_literal:e
722   {
723     !
724     TeXDict ~ begin ~
725     /color #1
726     {
727       [ ~
728       /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
729       [ ~ #3 ~ ] ~
730       {
731         \cs_if_exist_use:cF { \_color_backend_separation_init_ #3 :nnn }
732         { \_color_backend_separation_init:nnn }
733         {#4} {#5} {#6}
734       }
735       ] ~ setcolorspace
736     } ~ def ~
737     end
738   }
739 }
740 \cs_new:cpn { \_color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
741 { \_color_backend_separation_init_Device:Nn 4 {#3} }
742 \cs_new:cpn { \_color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
743 { \_color_backend_separation_init_Device:Nn 1 {#3} }
744 \cs_new:cpn { \_color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3

```

```

745 { \_color_backend_separation_init_Device:Nn 2 {#3} }
746 \cs_new:Npn \_color_backend_separation_init_Device:Nn #1#2
747 {
748   #2 ~
749   \prg_replicate:nn {#1}
750   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
751   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
752 }

```

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.

```

753 \cs_new:Npn \_color_backend_separation_init:nnn #1#2#3
754 {
755   \exp_args:Ne \_color_backend_separation_init:nnnn
756   { \_color_backend_separation_init_count:n {#2} }
757   {#1} {#2} {#3}
758 }
759 \cs_new:Npn \_color_backend_separation_init_count:n #1
760 { \int_eval:n { 0 } \_color_backend_separation_init_count:w #1 ~ \s_color_stop } }
761 \cs_new:Npn \_color_backend_separation_init_count:w #1 ~ #2 \s_color_stop
762 {
763   +1
764   \tl_if_blank:nF {#2}
765   { \_color_backend_separation_init_count:w #2 \s_color_stop }
766 }

```

Now we implement the algorithm. In the terms in the PostScript manual, we have  $\mathbf{N} = 1$  and  $\mathbf{Domain} = [0 \ 1]$ , with  $\mathbf{Range}$  as #2,  $\mathbf{C0}$  as #3 and  $\mathbf{C1}$  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  $\mathbf{C0}$  and  $\mathbf{C1}$  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.

```

767 \cs_new:Npn \_color_backend_separation_init:nnnn #1#2#3#4
768 {
769   \_color_backend_separation_init:w #3 ~ \s_color_stop #4 ~ \s_color_stop
770   \prg_replicate:nn {#1}
771   {
772     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
773     \int_eval:n { 3 * #1 } ~ index ~ mul ~
774     2 ~ index ~ add ~
775     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
776   }
777   \int_step_function:nnnN {#1} { -1 } { 1 }
778   \_color_backend_separation_init:n
779   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
780   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
781   \tl_if_blank:nF {#2}

```

```

782     { \_color_backend_separation_init:nw {#1} #2 ~ \s_color_stop }
783   }
784   \cs_new:Npn \_color_backend_separation_init:w
785     #1 ~ #2 \s_color_stop #3 ~ #4 \s_color_stop
786   {
787     #1 ~ #3 ~ 0 ~
788     \tl_if_blank:nF {#2}
789     { \_color_backend_separation_init:w #2 \s_color_stop #4 \s_color_stop }
790   }
791   \cs_new:Npn \_color_backend_separation_init:n #1
792   { \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.

```

793   \cs_new:Npn \_color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s_color_stop
794   {
795     #2 ~ #3 ~
796     2 ~ index ~ 2 ~ index ~ lt ~
797     { ~ pop ~ excl ~ pop ~ } ~
798     { ~
799       2 ~ index ~ 1 ~ index ~ gt ~
800       { ~ excl ~ pop ~ excl ~ pop ~ } ~
801       { ~ pop ~ pop ~ } ~
802       ifelse ~
803     }
804     ifelse ~
805     #1 ~ 1 ~ roll ~
806     \tl_if_blank:nF {#4}
807     { \_color_backend_separation_init:nw {#1} #4 \s_color_stop }
808   }

```

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.

```

809   \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
810   {
811     \_color_backend_separation_init:nxxxnn
812     {#2}
813     {
814       /CIEBasedABC ~
815       << ~
816       /RangeABC ~ [ ~ \c_color_model_range_CIELAB_tl \c_space_tl ] ~
817       /DecodeABC ~
818       [ ~
819         { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
820         { ~ 500 ~ div ~ } ~ bind ~
821         { ~ 200 ~ div ~ } ~ bind ~
822       ] ~
823       /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
824       /DecodeLMN ~
825       [ ~
826         { ~
827           dup ~ 6 ~ 29 ~ div ~ ge ~
828           { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
829           { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~

```

```

830         ifelse ~
831         0.9505 ~ mul ~
832     } ~ bind ~
833     { ~
834         dup ~ 6 ~ 29 ~ div ~ ge ~
835         { ~ dup ~ dup ~ mul ~ mul ~ } ~
836         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
837         ifelse ~
838     } ~ bind ~
839     { ~
840         dup ~ 6 ~ 29 ~ div ~ ge ~
841         { ~ dup ~ dup ~ mul ~ mul ~ } ~
842         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
843         ifelse ~
844         1.0890 ~ mul ~
845     } ~ bind
846 ] ~
847 /WhitePoint ~
848 [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
849 >>
850 }
851 { \c__color_model_range_CIELAB_tl }
852 { 100 ~ 0 ~ 0 }
853 {#3}
854 }

```

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

`\_color_backend_devicen_init:nmn` Trivial as almost all of the work occurs in the shared code.

```

855 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
856 {
857     \_kernel_backend_literal:e
858     {
859         !
860         TeXDict ~ begin ~
861         /color \int_use:N \g__color_model_int
862         {
863             [ ~
864             /DeviceN ~
865             [ ~ #1 ~ ] ~
866             #2 ~
867             { ~ #3 ~ } ~
868             \_color_backend_devicen_colorants:n {#1}
869             ] ~ setcolorspace
870         } ~ def ~
871         end
872     }
873 }

```

(End definition for `\_color_backend_devicen_init:nnn`.)

`\_color_backend_iccbased_init:nmn` No support at present.

```

874 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3 { }

```

(End definition for `\_color_backend_iccbased_init:nnn`.)

875 `</dvips>`

876 `<*dvisvgm>`

`\_color_backend_select_separation:nn` No support at present.

`\_color_backend_select_devicen:nn` 877 `\cs_new_protected:Npn \_color_backend_select_separation:nn #1#2 { }`

878 `\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` No support at present.

`\_color_backend_separation_init_CIELAB:nnn` 879 `\cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }`

880 `\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`.)

`\_color_backend_select_iccbased:nn` As detailed in <https://www.w3.org/TR/css-color-4/#at-profile>, we can apply a color profile using CSS. As we have a local file, we use a relative URL.

881 `\cs_new_protected:Npn \_color_backend_select_iccbased:nn #1#2`

882 `{`

883 `\_kernel_backend_literal_svg:x`

884 `{`

885 `<style>`

886 `@color-profile ~`

887 `\str_if_eq:nnTF {#2} { cmyk }`

888 `{ device-cmyk }`

889 `{ --color \int_use:N \g_color_model_int }`

890 `\c_space_tl`

891 `{`

892 `src:("#1")`

893 `}`

894 `</style>`

895 `}`

896 `}`

(End definition for `\_color_backend_select_iccbased:nn`.)

897 `</dvisvgm>`

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

`\_color_backend_select_separation: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.

`\_color_backend_select_devicen:nn`

`\_color_backend_select_iccbased:nn`

899 `\cs_new_protected:Npn \_color_backend_select_separation:nn #1#2`

900 `{ \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }`

901 `\cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn`

902 `\cs_new_eq:NN \_color_backend_select_iccbased:nn \_color_backend_select_separation:nn`

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

```

__color_backend_separation_init:nnnn
__color_backend_separation_init:nn
__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.

```

903 \cs_new_protected:Npn __color_backend_separation_init:nnnn #1#2#3#4#5
904 {
905   \pdf_object_unnamed_write:nx { dict }
906   {
907     /FunctionType ~ 2
908     /Domain ~ [0 ~ 1]
909     \tl_if_blank:nF {#3} { /Range ~ [#3] }
910     /C0 ~ [#4] ~
911     /C1 ~ [#5] /N ~ 1
912   }
913   \exp_args:Nx __color_backend_separation_init:nn
914   { \str_convert_pdfname:n {#1} } {#2}
915   \bool_lazy_and:nnT
916   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
917   { \pdfmanagement_if_active_p: }
918   {
919     \use:x
920     {
921       \pdfmanagement_add:nnn
922       { Page / Resources / ColorSpace }
923       { color \int_use:N \g__color_model_int }
924       { \pdf_object_ref_last: }
925     }
926   }
927 }
928 \cs_new_protected:Npn __color_backend_separation_init:nn #1#2
929 {
930   \pdf_object_unnamed_write:nx { array }
931   { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
932   \prop_gput:Nnx \g__color_backend_colorant_prop { /#1 }
933   { \pdf_object_ref_last: }
934 }

```

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

```

935 \cs_new_protected:Npn __color_backend_separation_init_CIELAB:nnn #1#2#3
936 {
937   \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
938   {
939     \pdf_object_new:nn { __color_illuminant_CIELAB_ #1 } { array }
940     \pdf_object_write:nx { __color_illuminant_CIELAB_ #1 }
941     {
942       /Lab ~
943       <<
944       /WhitePoint ~
945       [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
946       /Range ~ [ \c__color_model_range_CIELAB_tl ]
947       >>
948     }
949   }
950   __color_backend_separation_init:nnnn

```

```

951     {#2}
952     { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
953     { \c__color_model_range_CIELAB_t1 }
954     { 100 ~ 0 ~ 0 }
955     {#3}
956 }

```

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

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

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

```

957 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
958 {
959   \pdf_object_unnamed_write:nx { stream }
960   {
961     {
962       /FunctionType ~ 4 ~
963       /Domain ~
964       [ ~
965         \prg_replicate:nn
966         { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
967         { 0 ~ 1 ~ }
968       ] ~
969       /Range ~
970       [ ~
971         \str_case:nn {#2}
972         {
973           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
974           { /DeviceGray } { 0 ~ 1 }
975           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
976         } ~
977       ]
978     }
979     { {#3} }
980   }
981   \pdf_object_unnamed_write:nx { array }
982   {
983     /DeviceN ~
984     [ ~ #1 ~ ] ~
985     #2 ~
986     \pdf_object_ref_last:
987     \__color_backend_devicen_colorants:n {#1}
988   }
989   \bool_lazy_and:nnT
990   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
991   { \pdfmanagement_if_active_p: }
992   {
993     \use:x
994     {
995       \pdfmanagement_add:nnn
996       { Page / Resources / ColorSpace }
997       { color \int_use:N \g__color_model_int }
998       { \pdf_object_ref_last: }

```

```

999         }
1000     }
1001 }
1002 \cs_new:Npn \__color_backend_devicen_init:w #1 ~ #2 \s__color_stop
1003 {
1004     + 1
1005     \tl_if_blank:nF {#2}
1006     { \__color_backend_devicen_init:w #2 \s__color_stop }
1007 }

(End definition for \__color_backend_devicen_init:nnn and \__color_backend_devicen_init:w.)

```

\\_\_color\_backend\_iccbased\_init:nnn Lots of data to save here: we only want to do that once per file, so track it by name.

```

1008 \cs_new_protected:Npn \__color_backend_iccbased_init:nnn #1#2#3
1009 {
1010     \pdf_object_if_exist:nF { __color_icc_ #1 }
1011     {
1012         \pdf_object_new:nn { __color_icc_ #1 } { fstream }
1013         \pdf_object_write:nx { __color_icc_ #1 }
1014         {
1015             {
1016                 /N ~ \exp_not:n { #2 } ~
1017                 \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
1018             }
1019             {#1}
1020         }
1021     }
1022     \pdf_object_unnamed_write:nx { array }
1023     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
1024     \cs_if_exist:NT \pdfmanagement_add:nnn
1025     {
1026         \use:x
1027         {
1028             \pdfmanagement_add:nnn { Page / Resources / ColorSpace }
1029             { color \int_use:N \g__color_model_int }
1030             { ~ \pdf_object_ref_last: }
1031         }
1032     }
1033 }

```

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

\\_\_color\_backend\_iccbased\_device:nnn This is very similar to setting up a color space: the only part we skip is adding it to the page resources.

```

1034 \cs_new_protected:Npn \__color_backend_iccbased_device:nnn #1#2#3
1035 {
1036     \pdf_object_if_exist:nF { __color_icc_ #1 }
1037     {
1038         \pdf_object_new:nn { __color_icc_ #1 } { fstream }
1039         \pdf_object_write:nn { __color_icc_ #1 }
1040         {
1041             { /N ~ #3 }
1042             {#1}
1043         }
1044     }

```

```

1044     }
1045     \pdf_object_unnamed_write:nx { array }
1046     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
1047     \cs_if_exist:NT \pdfmanagement_add:nnn
1048     {
1049         \use:x
1050         {
1051             \pdfmanagement_add:nnn
1052             { Page / Resources / ColorSpace }
1053             { Default #2 }
1054             { \pdf_object_ref_last: }
1055         }
1056     }
1057 }

```

(End definition for `\__color_backend_iccbased_device:nnn`.)

```

1058 </dvipdfmx | luatex | pdftex | xetex>
1059 <*dvipdfmx | xetex>

```

`\__color_backend_select_separation:nn` 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.

```

1060 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
1061 {
1062     \cs_gset_protected:Npn \__color_backend_select_separation:nn #1#2 { }
1063     \cs_gset_eq:NN \__color_backend_select_devicen:nn
1064     \__color_backend_select_separation:nn
1065 }

```

(End definition for `\__color_backend_select_separation:nn` and `\__color_backend_select_devicen:nn`.)

```

1066 </dvipdfmx | xetex>

```

### 3.5 Fill and stroke color

Here, dvipdfmx/X<sub>Y</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.

```

1067 <*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.

```

1068 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1069 { \__color_backend_fill:n { #1 ~ k } }
1070 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1071 { \__color_backend_fill:n { #1 ~ g } }
1072 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1073 { \__color_backend_fill:n { #1 ~ rg } }
1074 \cs_new_protected:Npn \__color_backend_fill:n #1
1075 {
1076     \tl_set:Nn \l__color_backend_fill_tl {#1}

```

```

1077 \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1078 { #1 ~ \l__color_backend_stroke_tl }
1079 \group_insert_after:N \__color_backend_reset:
1080 }
1081 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1082 { \__color_backend_stroke:n { #1 ~ K } }
1083 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1084 { \__color_backend_stroke:n { #1 ~ G } }
1085 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1086 { \__color_backend_stroke:n { #1 ~ RG } }
1087 \cs_new_protected:Npn \__color_backend_stroke:n #1
1088 {
1089 \tl_set:Nn \l__color_backend_stroke_tl {#1}
1090 \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1091 { \l__color_backend_fill_tl \c_space_tl #1 }
1092 \group_insert_after:N \__color_backend_reset:
1093 }

```

(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
1094 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
1095 { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
1096 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1097 { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
1098 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1099 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

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

```

1100 </dvipdfmx | luatex | pdftex | xetex>
1101 <*dvipdfmx | xetex>

```

Deal with older (x)dvipdfmx.

```

\__color_backend_fill_cmyk:n
\__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
1102 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
1103 {
1104 \cs_gset_protected:Npn \__color_backend_fill_cmyk:n #1
1105 {
1106 \__kernel_backend_literal:n { pdf: bc ~ [#1] }
1107 \group_insert_after:N \__color_backend_reset:
1108 }
1109 \cs_gset_eq:NN \__color_backend_fill_gray:n \__color_backend_fill_cmyk:n
1110 \cs_gset_eq:NN \__color_backend_fill_rgb:n \__color_backend_fill_cmyk:n
1111 \cs_gset_protected:Npn \__color_backend_reset:
1112 { \__kernel_backend_literal:n { pdf: ec } }
1113 \cs_gset_protected:Npn \__color_backend_stroke:n #1
1114 { \__kernel_backend_literal:n {#1} }
1115 \cs_gset_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1116 \cs_gset_eq:NN \__color_backend_fill_devicen:nn
1117 \__color_backend_fill_separation:nn
1118 \cs_gset_eq:NN \__color_backend_stroke_separation:nn
1119 \__color_backend_fill_separation:nn
1120 \cs_gset_eq:NN \__color_backend_stroke_devicen:nn
1121 \__color_backend_stroke_separation:nn
1122 }

```

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

1123 `</dvipdfmx | xetex>`

1124 `<*dvips>`

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

```

1125 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1126 { \_color_backend_fill:n { cmyk ~ #1 } }
1127 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1128 { \_color_backend_fill:n { gray ~ #1 } }
1129 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1130 { \_color_backend_fill:n { rgb ~ #1 } }
1131 \cs_new_protected:Npn \_color_backend_fill:n #1
1132 {
1133   \_kernel_backend_literal:n { color~push~ #1 }
1134   \group_insert_after:N \_color_backend_reset:
1135 }
1136 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1137 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1138 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1139 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1140 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1141 { \_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
\_color_backend_fill_devicen:nn
\_color_backend_stroke_devicen:nn
1142 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1143 { \_color_backend_fill:n { separation ~ #1 ~ #2 } }
1144 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1145 { \_kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1146 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1147 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

1148 `</dvips>`

1149 `<*dvisvgm>`

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

```

1150 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1151 { \_color_backend_fill:n { cmyk ~ #1 } }
1152 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1153 { \_color_backend_fill:n { gray ~ #1 } }
1154 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1155 { \_color_backend_fill:n { rgb ~ #1 } }
1156 \cs_new_protected:Npn \_color_backend_fill:n #1
1157 {
1158   \_kernel_backend_literal:n { color~push~ #1 }
1159   \group_insert_after:N \_color_backend_reset:
1160 }

```

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

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:n
\__color_backend_stroke_cmyk:w
\__color_backend_stroke_gray:n
\__color_backend_stroke_gray_aux:n
\__color_backend_stroke_rgb:n
\__color_backend_stroke_rgb:w
\__color_backend:nnn
1161 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1162 { \__color_backend_cmyk:w #1 \s__color_stop }
1163 \cs_new_protected:Npn \__color_backend_stroke_cmyk:w
1164 #1 ~ #2 ~ #3 ~ #4 \s__color_stop
1165 {
1166   \use:x
1167   {
1168     \__color_backend:nnn
1169     { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1170     { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1171     { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }
1172   }
1173 }
1174 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1175 {
1176   \use:x
1177   {
1178     \__color_backend_stroke_gray_aux:n
1179     { \fp_eval:n { 100 * (#1) } }
1180   }
1181 }
1182 \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1183 { \__color_backend:nnn {#1} {#1} {#1} }
1184 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1185 { \__color_backend_rgb:w #1 \s__color_stop }
1186 \cs_new_protected:Npn \__color_backend_stroke_rgb:w
1187 #1 ~ #2 ~ #3 \s__color_stop
1188 {
1189   \use:x
1190   {
1191     \__color_backend:nnn
1192     { \fp_eval:n { 100 * (#1) } }
1193     { \fp_eval:n { 100 * (#2) } }
1194     { \fp_eval:n { 100 * (#3) } }
1195   }
1196 }
1197 \cs_new_protected:Npx \__color_backend:nnn #1#2#3
1198 {
1199   \__kernel_backend_scope:n
1200   {
1201     stroke =
1202     "
1203     rgb
1204     (
1205       #1 \c_percent_str ,
1206       #2 \c_percent_str ,
1207       #3 \c_percent_str
1208     )
1209     "
1210   }
1211 }

```

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

At present, these are no-ops.

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
1212 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1213 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2 { }
1214 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1215 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn
(End definition for \_color_backend_fill_separation:nn and others.)
1216 </dvisvgm>
1217 </package>

```

## 4 l3backend-draw Implementation

```

1218 <*package>
1219 <@@=draw>

```

### 4.1 dvips backend

```

1220 <*dvips>

```

The same as literal PostScript: same arguments about positioning apply her.

```

\_draw_backend_literal:n
\_draw_backend_literal:x
1221 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
1222 \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.

```

1223 \cs_new_protected:Npn \_draw_backend_begin:
1224 {
1225   \_kernel_backend_literal:n { ps::[begin] }
1226   \_draw_backend_literal:n { @beginspecial }
1227 }
1228 \cs_new_protected:Npn \_draw_backend_end:
1229 {
1230   \_draw_backend_literal:n { @endspecial }
1231   \_kernel_backend_literal:n { ps::[end] }
1232 }

```

(End definition for `\_draw_backend_begin:` and `\_draw_backend_end:.`)

```

\_draw_backend_scope_begin:
\_draw_backend_scope_end:

```

Scope here may need to contain saved definitions, so the entire memory rather than just the graphic state has to be sent to the stack.

```

1233 \cs_new_protected:Npn \_draw_backend_scope_begin:
1234 { \_draw_backend_literal:n { save } }
1235 \cs_new_protected:Npn \_draw_backend_scope_end:
1236 { \_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 the need to convert to bp. Notice that x-type expansion is included here to ensure that any variable values are forced to literals before any possible caching. There is no native rectangular path command (without also clipping, filling or stroking), so that task is done using a small amount of PostScript.

```

1237 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1238 {
1239   \_draw_backend_literal:x
1240   {
1241     \dim_to_decimal_in_bp:n {#1} ~
1242     \dim_to_decimal_in_bp:n {#2} ~ moveto
1243   }
1244 }
1245 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1246 {
1247   \_draw_backend_literal:x
1248   {
1249     \dim_to_decimal_in_bp:n {#1} ~
1250     \dim_to_decimal_in_bp:n {#2} ~ lineto
1251   }
1252 }
1253 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1254 {
1255   \_draw_backend_literal:x
1256   {
1257     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1258     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1259     moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1260   }
1261 }
1262 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1263 {
1264   \_draw_backend_literal:x
1265   {
1266     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1267     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1268     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1269     curveto
1270   }
1271 }

```

(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`

```

1272 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1273 { \bool_gset_true:N \g__draw_draw_eor_bool }
1274 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1275 { \bool_gset_false:N \g__draw_draw_eor_bool }
1276 \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 T<sub>E</sub>X 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`

```

1277 \cs_new_protected:Npn \__draw_backend_closepath:
1278 { \__draw_backend_literal:n { closepath } }
1279 \cs_new_protected:Npn \__draw_backend_stroke:
1280 {
1281   \__draw_backend_literal:n { gsave }
1282   \__draw_backend_literal:n { color.sc }
1283   \__draw_backend_literal:n { stroke }
1284   \__draw_backend_literal:n { grestore }
1285   \bool_if:NT \g__draw_draw_clip_bool
1286   {
1287     \__draw_backend_literal:x
1288     {
1289       \bool_if:NT \g__draw_draw_eor_bool { eo }
1290       clip
1291     }
1292   }
1293   \__draw_backend_literal:n { newpath }
1294   \bool_gset_false:N \g__draw_draw_clip_bool
1295 }
1296 \cs_new_protected:Npn \__draw_backend_closestroke:
1297 {
1298   \__draw_backend_closepath:
1299   \__draw_backend_stroke:
1300 }
1301 \cs_new_protected:Npn \__draw_backend_fill:
1302 {
1303   \__draw_backend_literal:x
1304   {
1305     \bool_if:NT \g__draw_draw_eor_bool { eo }
1306     fill
1307   }
1308   \bool_if:NT \g__draw_draw_clip_bool
1309   {
1310     \__draw_backend_literal:x
1311     {
1312       \bool_if:NT \g__draw_draw_eor_bool { eo }
1313       clip
1314     }
1315   }
1316   \__draw_backend_literal:n { newpath }
1317   \bool_gset_false:N \g__draw_draw_clip_bool
1318 }
1319 \cs_new_protected:Npn \__draw_backend_fillstroke:
1320 {
1321   \__draw_backend_literal:x
1322   {
1323     \bool_if:NT \g__draw_draw_eor_bool { eo }

```

```

1324         fill
1325     }
1326     \__draw_backend_literal:n { gsave }
1327     \__draw_backend_literal:n { color.sc }
1328     \__draw_backend_literal:n { stroke }
1329     \__draw_backend_literal:n { grestore }
1330     \bool_if:NT \g__draw_draw_clip_bool
1331     {
1332         \__draw_backend_literal:x
1333         {
1334             \bool_if:NT \g__draw_draw_eor_bool { eo }
1335             clip
1336         }
1337     }
1338     \__draw_backend_literal:n { newpath }
1339     \bool_gset_false:N \g__draw_draw_clip_bool
1340 }
1341 \cs_new_protected:Npn \__draw_backend_clip:
1342 { \bool_gset_true:N \g__draw_draw_clip_bool }
1343 \bool_new:N \g__draw_draw_clip_bool
1344 \cs_new_protected:Npn \__draw_backend_discardpath:
1345 {
1346     \bool_if:NT \g__draw_draw_clip_bool
1347     {
1348         \__draw_backend_literal:x
1349         {
1350             \bool_if:NT \g__draw_draw_eor_bool { eo }
1351             clip
1352         }
1353     }
1354     \__draw_backend_literal:n { newpath }
1355     \bool_gset_false:N \g__draw_draw_clip_bool
1356 }

```

(End definition for `\__draw_backend_closepath:` and others.)

Converting paths to output is again a case of mapping directly to PostScript 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:
1357 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1358 {
1359     \__draw_backend_literal:x
1360     {
1361         [
1362             \exp_args:Nf \use:n
1363             { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1364         ] ~
1365         \dim_to_decimal_in_bp:n {#2} ~ setdash
1366     }
1367 }
1368 \cs_new:Npn \__draw_backend_dash:n #1
1369 { ~ \dim_to_decimal_in_bp:n {#1} }
1370 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1371 {
1372     \__draw_backend_literal:x
1373     { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }

```

```

1374 }
1375 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1376 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1377 \cs_new_protected:Npn \__draw_backend_cap_butt:
1378 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1379 \cs_new_protected:Npn \__draw_backend_cap_round:
1380 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1381 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1382 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1383 \cs_new_protected:Npn \__draw_backend_join_miter:
1384 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1385 \cs_new_protected:Npn \__draw_backend_join_round:
1386 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1387 \cs_new_protected:Npn \__draw_backend_join_bevel:
1388 { \__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/XYTeX`). Thus we take the shortest path available and simply dump the matrix as given.

```

1389 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1390 {
1391   \__draw_backend_literal:n
1392   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1393 }

```

(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 `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).

```

1394 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1395 {
1396   \__draw_backend_literal:n { @endspecial }
1397   \__draw_backend_literal:n { [end] }
1398   \__draw_backend_literal:n { [begin] }
1399   \__draw_backend_literal:n { save }
1400   \__draw_backend_literal:n { currentpoint }
1401   \__draw_backend_literal:n { currentpoint~translate }
1402   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1403   \__draw_backend_cm:nnnn { #2 } { #3 } { #4 } { #5 }
1404   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1405   \__draw_backend_literal:n { neg~exch~neg~exch~translate }

```

```

1406     \__draw_backend_literal:n { [end] }
1407     \hbox_overlap_right:n { \box_use:N #1 }
1408     \__draw_backend_literal:n { [begin] }
1409     \__draw_backend_literal:n { restore }
1410     \__draw_backend_literal:n { [end] }
1411     \__draw_backend_literal:n { [begin] }
1412     \__draw_backend_literal:n { @beginspecial }
1413 }

```

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

```

1414 </dvips>

```

## 4.2 LuaTeX, pdfTeX, dvipdfmx and XeTeX

LuaTeX, pdfTeX, dvipdfmx and XeTeX directly produce PDF output and understand a shared set of specials for drawing commands.

```

1415 <*dvipdfmx | luatex | pdftex | xetex>

```

### 4.2.1 Drawing

\\_\_draw\_backend\_literal:n Pass data through using a dedicated interface.

```

\__draw_backend_literal:x
1416 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1417 \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:
1418 \cs_new_protected:Npn \__draw_backend_begin:
1419 { \__draw_backend_scope_begin: }
1420 \cs_new_protected:Npn \__draw_backend_end:
1421 { \__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:
1422 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1423 \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 to convert to bp.

```

\__draw_backend_lineto:nn
\__draw_backend_moveto:nnnnnn
\__draw_backend_rectangle:nnnn
1424 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1425 {
1426     \__draw_backend_literal:x
1427     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1428 }
1429 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1430 {
1431     \__draw_backend_literal:x
1432     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1433 }
1434 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1435 {

```

```

1436 \__draw_backend_literal:x
1437 {
1438   \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1439   \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1440   \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1441   c
1442 }
1443 }
1444 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1445 {
1446   \__draw_backend_literal:x
1447   {
1448     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1449     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1450     re
1451   }
1452 }

```

(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
1453 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1454 { \bool_gset_true:N \g__draw_draw_eor_bool }
1455 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1456 { \bool_gset_false:N \g__draw_draw_eor_bool }
1457 \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:
1458 \cs_new_protected:Npn \__draw_backend_closepath:
1459 { \__draw_backend_literal:n { h } }
1460 \cs_new_protected:Npn \__draw_backend_stroke:
1461 { \__draw_backend_literal:n { S } }
1462 \cs_new_protected:Npn \__draw_backend_closestroke:
1463 { \__draw_backend_literal:n { s } }
1464 \cs_new_protected:Npn \__draw_backend_fill:
1465 {
1466   \__draw_backend_literal:x
1467   { f \bool_if:NT \g__draw_draw_eor_bool * }
1468 }
1469 \cs_new_protected:Npn \__draw_backend_fillstroke:
1470 {
1471   \__draw_backend_literal:x
1472   { B \bool_if:NT \g__draw_draw_eor_bool * }
1473 }
1474 \cs_new_protected:Npn \__draw_backend_clip:
1475 {
1476   \__draw_backend_literal:x
1477   { W \bool_if:NT \g__draw_draw_eor_bool * }
1478 }
1479 \cs_new_protected:Npn \__draw_backend_discardpath:
1480 { \__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_but:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1481 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1482 {
1483   \__draw_backend_literal:x
1484   {
1485     [
1486       \exp_args:Nf \use:n
1487       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1488     ] ~
1489     \dim_to_decimal_in_bp:n {#2} ~ d
1490   }
1491 }
1492 \cs_new:Npn \__draw_backend_dash:n #1
1493 { ~ \dim_to_decimal_in_bp:n {#1} }
1494 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1495 {
1496   \__draw_backend_literal:x
1497   { \dim_to_decimal_in_bp:n {#1} ~ w }
1498 }
1499 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1500 { \__draw_backend_literal:x { #1 ~ M } }
1501 \cs_new_protected:Npn \__draw_backend_cap_but:
1502 { \__draw_backend_literal:n { 0 ~ J } }
1503 \cs_new_protected:Npn \__draw_backend_cap_round:
1504 { \__draw_backend_literal:n { 1 ~ J } }
1505 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1506 { \__draw_backend_literal:n { 2 ~ J } }
1507 \cs_new_protected:Npn \__draw_backend_join_miter:
1508 { \__draw_backend_literal:n { 0 ~ j } }
1509 \cs_new_protected:Npn \__draw_backend_join_round:
1510 { \__draw_backend_literal:n { 1 ~ j } }
1511 \cs_new_protected:Npn \__draw_backend_join_bevel:
1512 { \__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<sub>+</sub> versions!

```

1513 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1514 {
1515   <*luatex | pdftex>
1516   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1517   </luatex | pdftex>
1518   <*dvipdfmx | xetex>
1519   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1520   \__draw_backend_cm_aux:nnnn
1521   </dvipdfmx | xetex>

```

```

1522 }
1523 <*/dvipdfmx | xetex>
1524 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1525 {
1526   \__kernel_backend_literal:x
1527   {
1528     x:rotate~
1529     \fp_compare:nNnTF {#1} = \c_zero_fp
1530     { 0 }
1531     { \fp_eval:n { round ( -#1 , 5 ) } }
1532   }
1533   \__kernel_backend_literal:x
1534   {
1535     x:scale~
1536     \fp_eval:n { round ( #2 , 5 ) } ~
1537     \fp_eval:n { round ( #3 , 5 ) }
1538   }
1539   \__kernel_backend_literal:x
1540   {
1541     x:rotate~
1542     \fp_compare:nNnTF {#4} = \c_zero_fp
1543     { 0 }
1544     { \fp_eval:n { round ( -#4 , 5 ) } }
1545   }
1546 }
1547 </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.

```

1548 <*dvipdfmx|xetex>
1549 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1550 {
1551   \use:x
1552   {
1553     \__draw_backend_cm_decompose_auxi:nnnnN
1554     { \fp_eval:n { (#1 + #4) / 2 } }
1555     { \fp_eval:n { (#1 - #4) / 2 } }
1556     { \fp_eval:n { (#3 + #2) / 2 } }
1557     { \fp_eval:n { (#3 - #2) / 2 } }
1558   }
1559   #5
1560 }
1561 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1562 {
1563   \use:x
1564   {
1565     \__draw_backend_cm_decompose_auxii:nnnnN
1566     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1567     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1568     { \fp_eval:n { atand ( #3 , #2 ) } }
1569     { \fp_eval:n { atand ( #4 , #1 ) } }
1570   }
1571   #5
1572 }
1573 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1574 {
1575   \use:x
1576   {
1577     \__draw_backend_cm_decompose_auxiii:nnnnN
1578     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1579     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1580     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1581     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1582   }
1583   #5
1584 }
1585 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1586 {
1587   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1588   { #5 {#1} {#2} {#3} {#4} }
1589   { #5 {#1} {#3} {#2} {#4} }
1590 }
1591 </dvipdfmx|xetex>

```

(End definition for `\__draw_backend_cm_decompose:nnnnN` and others.)

`\__draw_backend_box_use:Nnnnn`

Inserting a  $\text{\TeX}$  box transformed to the requested position and using the current matrix is done using a mixture of  $\text{\TeX}$  and low-level manipulation. The offset can be handled by  $\text{\TeX}$ , 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.

```

1592 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1593 {
1594   \__kernel_backend_scope_begin:
1595   <*luatex | pdftex>
1596   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1597   </luatex | pdftex>
1598   <*dvipdfmx | xetex>
1599   \__kernel_backend_literal:n
1600   { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1601   </dvipdfmx | xetex>
1602   \hbox_overlap_right:n { \box_use:N #1 }
1603   <*dvipdfmx | xetex>
1604   \__kernel_backend_literal:n { pdf:etrans }
1605   </dvipdfmx | xetex>
1606   \__kernel_backend_scope_end:
1607   }

```

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

```

1608 </dvipdfmx | luatex | pdftex | xetex>

```

### 4.3 dvisvgm backend

```

1609 <*dvisvgm>

```

The same as the more general literal call.

```

\__draw_backend_literal:n
\__draw_backend_literal:x
1610 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1611 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

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

Use the backend-level scope mechanisms.

```

\__draw_backend_scope_begin:
\__draw_backend_scope_end:
1612 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1613 \cs_new_eq:NN \__draw_backend_scope_end: \__kernel_backend_scope_end:

```

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

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

```

\__draw_backend_begin:
\__draw_backend_end:
1614 \cs_new_protected:Npn \__draw_backend_begin:
1615 {
1616   \__kernel_backend_scope_begin:
1617   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1618   }
1619 \cs_new_eq:NN \__draw_backend_end: \__kernel_backend_scope_end:

```

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

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

```

\__draw_backend_moveto:nn
\__draw_backend_lineto:nn
\__draw_backend_rectangle:nnnn
\__draw_backend_curveto:nnnnnn
\__draw_backend_add_to_path:n
\g__draw_backend_path_tl
1620 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1621 {

```

```

1622     \__draw_backend_add_to_path:n
1623     { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1624 }
1625 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1626 {
1627     \__draw_backend_add_to_path:n
1628     { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1629 }
1630 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1631 {
1632     \__draw_backend_add_to_path:n
1633     {
1634         M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1635         h ~ \dim_to_decimal:n {#3} ~
1636         v ~ \dim_to_decimal:n {#4} ~
1637         h ~ \dim_to_decimal:n { -#3 } ~
1638         Z
1639     }
1640 }
1641 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1642 {
1643     \__draw_backend_add_to_path:n
1644     {
1645         C ~
1646         \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1647         \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1648         \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1649     }
1650 }
1651 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1652 {
1653     \tl_gset:Nx \g__draw_backend_path_tl
1654     {
1655         \g__draw_backend_path_tl
1656         \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1657         #1
1658     }
1659 }
1660 \tl_new:N \g__draw_backend_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: 1661 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1662     { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1663 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1664     { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

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

```

\__draw_backend_path:n
\__draw_backend_closepath:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int

```

Setting fill and stroke effects and doing clipping all has to be done using scopes. This means setting up the various requirements in a shared auxiliary which deals with the bits and pieces. Clipping paths are reused for path drawing: not essential but avoids constructing them twice. Discarding a path needs a separate function as it's not quite the same.

```

1665 \cs_new_protected:Npn \__draw_backend_closepath:
1666 { \__draw_backend_add_to_path:n { Z } }
1667 \cs_new_protected:Npn \__draw_backend_path:n #1
1668 {
1669   \bool_if:NTF \g__draw_draw_clip_bool
1670   {
1671     \int_gincr:N \g__kernel_clip_path_int
1672     \__draw_backend_literal:x
1673     {
1674       < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1675       { ?nl }
1676       <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1677       < /clipPath > { ? nl }
1678       <
1679       use~xlink:href =
1680       "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1681       #1
1682       />
1683     }
1684     \__kernel_backend_scope:x
1685     {
1686       clip-path =
1687       "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int )"
1688     }
1689   }
1690   {
1691     \__draw_backend_literal:x
1692     { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1693   }
1694   \tl_gclear:N \g__draw_backend_path_tl
1695   \bool_gset_false:N \g__draw_draw_clip_bool
1696 }
1697 \int_new:N \g__draw_backend_path_int
1698 \cs_new_protected:Npn \__draw_backend_stroke:
1699 { \__draw_backend_path:n { style="fill:none" } }
1700 \cs_new_protected:Npn \__draw_backend_closestroke:
1701 {
1702   \__draw_backend_closepath:
1703   \__draw_backend_stroke:
1704 }
1705 \cs_new_protected:Npn \__draw_backend_fill:
1706 { \__draw_backend_path:n { style="stroke:none" } }
1707 \cs_new_protected:Npn \__draw_backend_fillstroke:
1708 { \__draw_backend_path:n { } }
1709 \cs_new_protected:Npn \__draw_backend_clip:
1710 { \bool_gset_true:N \g__draw_draw_clip_bool }
1711 \bool_new:N \g__draw_draw_clip_bool
1712 \cs_new_protected:Npn \__draw_backend_discardpath:
1713 {
1714   \bool_if:NT \g__draw_draw_clip_bool
1715   {
1716     \int_gincr:N \g__kernel_clip_path_int
1717     \__draw_backend_literal:x
1718     {

```

```

1719         < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1720             { ?nl }
1721         <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1722         < /clipPath >
1723     }
1724     \__kernel_backend_scope:x
1725     {
1726         clip-path =
1727         "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1728     }
1729 }
1730 \tl_gclear:N \g__draw_path_tl
1731 \bool_gset_false:N \g__draw_draw_clip_bool
1732 }

```

(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:
1733 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1734 {
1735     \use:x
1736     {
1737         \__draw_backend_dash_aux:nn
1738         { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1739         { \dim_to_decimal:n {#2} }
1740     }
1741 }
1742 \cs_new:Npn \__draw_backend_dash:n #1
1743 { , \dim_to_decimal_in_bp:n {#1} }
1744 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1745 {
1746     \__kernel_backend_scope:x
1747     {
1748         stroke-dasharray =
1749         "
1750         \tl_if_empty:nTF {#1}
1751         { none }
1752         { \use_none:n #1 }
1753         " ~
1754         stroke-offset=" #2 "
1755     }
1756 }
1757 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1758 { \__kernel_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1759 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1760 { \__kernel_backend_scope:x { stroke-miterlimit=" #1 " } }
1761 \cs_new_protected:Npn \__draw_backend_cap_butt:
1762 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1763 \cs_new_protected:Npn \__draw_backend_cap_round:
1764 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1765 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1766 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1767 \cs_new_protected:Npn \__draw_backend_join_miter:

```

```

1768 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1769 \cs_new_protected:Npn \__draw_backend_join_round:
1770 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1771 \cs_new_protected:Npn \__draw_backend_join_bevel:
1772 { \__kernel_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.

```

1773 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1774 {
1775   \__kernel_backend_scope:n
1776   {
1777     transform =
1778     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1779   }
1780 }

```

(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.

```

1781 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1782 {
1783   \__kernel_backend_scope_begin:
1784   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1785   \__kernel_backend_literal_svg:n
1786   {
1787     < g~
1788     stroke="none"~
1789     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1790     >
1791   }
1792   \box_set_wd:Nn #1 { Opt }
1793   \box_set_ht:Nn #1 { Opt }
1794   \box_set_dp:Nn #1 { Opt }
1795   \box_use:N #1
1796   \__kernel_backend_literal_svg:n { </g> }
1797   \__kernel_backend_scope_end:
1798 }

```

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

```

1799 </dvisvgm>
1800 </package>

```

## 5 l3backend-graphics Implementation

```

1801 <*package>
1802 <@@=graphics>

```

### 5.1 dvips backend

```

1803 <*dvips>

```

`\_graphics_backend_getbb_eps:n` Simply use the generic function.

```

1804 \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.
1805 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1806 {
1807   \_kernel_backend_literal:x
1808   {
1809     PSfile = #1 \c_space_tl
1810     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1811     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1812     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1813     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1814   }
1815 }
(End definition for \_graphics_backend_include_eps:n.)
1816 </dvips>

```

## 5.2 LuaTeX and pdfTeX backends

1817 <\*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.

```

1818 \tl_new:N \l__graphics_graphics_attr_tl
(End definition for \l__graphics_graphics_attr_tl.)

```

`\_graphics_backend_getbb_jpg: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.

```

1819 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1820 {
1821   \int_zero:N \l_graphics_page_int
1822   \tl_clear:N \l_graphics_pagebox_tl
1823   \tl_set:Nx \l__graphics_graphics_attr_tl
1824   {
1825     \tl_if_empty:NF \l_graphics_decodearray_tl
1826     { :D \l_graphics_decodearray_tl }
1827     \bool_if:NT \l_graphics_interpolate_bool
1828     { :I }
1829   }
1830   \tl_clear:N \l__graphics_graphics_attr_tl
1831   \_graphics_backend_getbb_auxi:n {#1}
1832 }
1833 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n

```

```

1834 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1835 {
1836   \tl_clear:N \l_graphics_decodearray_tl
1837   \bool_set_false:N \l_graphics_interpolate_bool
1838   \tl_set:Nx \l__graphics_graphics_attr_tl
1839   {
1840     : \l_graphics_pagebox_tl
1841     \int_compare:nNnT \l_graphics_page_int > 1
1842     { :P \int_use:N \l_graphics_page_int }
1843   }
1844   \__graphics_backend_getbb_auxi:n {#1}
1845 }
1846 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1847 {
1848   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1849   { \__graphics_backend_getbb_auxii:n {#1} }
1850 }

```

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.

```

1851 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1852 {
1853   \tex_immediate:D \tex_pdfximage:D
1854   \bool_lazy_or:nnT
1855   { \l_graphics_interpolate_bool }
1856   { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1857   {
1858     attr ~
1859     {
1860       \tl_if_empty:NF \l_graphics_decodearray_tl
1861       { /Decode~[ \l_graphics_decodearray_tl ] }
1862       \bool_if:NT \l_graphics_interpolate_bool
1863       { /Interpolate~true }
1864     }
1865   }
1866   \int_compare:nNnT \l_graphics_page_int > 0
1867   { page ~ \int_use:N \l_graphics_page_int }
1868   \tl_if_empty:NF \l_graphics_pagebox_tl
1869   { \l_graphics_pagebox_tl }
1870   {#1}
1871   \hbox_set:Nn \l__graphics_internal_box
1872   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1873   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1874   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1875   \int_const:cn { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1876   { \tex_the:D \tex_pdflastximage:D }
1877   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1878 }

```

(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.

```

1879 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1880 {
1881   \tex_pdfrefximage:D
1882   \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1883 }
1884 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1885 \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.)

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.

```

\__graphics_backend_getbb_eps:n
\__graphics_backend_getbb_eps:nm
\__graphics_backend_include_eps:n
\l__graphics_backend_dir_str
\l__graphics_backend_name_str
\l__graphics_backend_ext_str
1886 \sys_if_shell:T
1887 {
1888   \str_new:N \l__graphics_backend_dir_str
1889   \str_new:N \l__graphics_backend_name_str
1890   \str_new:N \l__graphics_backend_ext_str
1891   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1892   {
1893     \file_parse_full_name:nNNN {#1}
1894     \l__graphics_backend_dir_str
1895     \l__graphics_backend_name_str
1896     \l__graphics_backend_ext_str
1897     \exp_args:Nx \__graphics_backend_getbb_eps:nn
1898     {
1899       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1900       -converted-to.pdf
1901     }
1902     {#1}
1903   }
1904   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1905   {
1906     \file_compare_timestamp:nNnT {#2} > {#1}
1907     {
1908       \sys_shell_now:n
1909       { repstopdf ~ #2 ~ #1 }
1910     }
1911     \tl_set:Nn \l__graphics_name_tl {#1}
1912     \__graphics_backend_getbb_pdf:n {#1}
1913   }
1914   \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1915   {
1916     \file_parse_full_name:nNNN {#1}
1917     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1918     \exp_args:Nx \__graphics_backend_include_pdf:n
1919     {
1920       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1921       -converted-to.pdf
1922     }
1923   }
1924 }

```

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

1925 `</luatex | pdftex>`

### 5.3 dvipdfmx backend

1926 `<*dvipdfmx | xetex>`

Simply use the generic functions: only for dvipdfmx in the extraction cases.

```

\__graphics_backend_getbb_eps:n
\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
1927 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n
1928 <*dvipdfmx>
1929 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1930 {
1931   \int_zero:N \l_graphics_page_int
1932   \tl_clear:N \l_graphics_pagebox_tl
1933   \graphics_extract_bb:n {#1}
1934 }
1935 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1936 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1937 {
1938   \tl_clear:N \l_graphics_decodearray_tl
1939   \bool_set_false:N \l_graphics_interpolate_bool
1940   \graphics_extract_bb:n {#1}
1941 }
1942 </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.

1943 `\int_new:N \g__graphics_track_int`

(End definition for `\g__graphics_track_int`.)

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.

```

\__graphics_backend_include_eps:n
\__graphics_backend_include_jpg:n
\__graphics_backend_include_pdf:n
\__graphics_backend_include_png:n
1944 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1945 {
1946   \__kernel_backend_literal:x
1947   {
1948     PSfile = #1 \c_space_tl
1949     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1950     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1951     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1952     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1953   }
1954 }
1955 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1956 { \__graphics_backend_include_auxi:nn {#1} { image } }
1957 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
1958 <*dvipdfmx>
1959 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1960 { \__graphics_backend_include_auxi:nn {#1} { epdf } }
1961 </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.

```

1962 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1963 {
1964   \__graphics_backend_include_auxii:xnn
1965   {
1966     \tl_if_empty:NF \l_graphics_pagebox_tl
1967     { : \l_graphics_pagebox_tl }
1968     \int_compare:nNnT \l_graphics_page_int > 1
1969     { :P \int_use:N \l_graphics_page_int }
1970     \tl_if_empty:NF \l_graphics_decodearray_tl
1971     { :D \l_graphics_decodearray_tl }
1972     \bool_if:NT \l_graphics_interpolate_bool
1973     { :I }
1974   }
1975   {#1} {#2}
1976 }
1977 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1978 {
1979   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1980   {
1981     \__kernel_backend_literal:x
1982     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1983   }
1984   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1985 }
1986 \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!

```

1987 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1988 {
1989   \int_gincr:N \g__graphics_track_int
1990   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1991   \__kernel_backend_literal:x
1992   {
1993     pdf:#3~
1994     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1995     \int_compare:nNnT \l_graphics_page_int > 1
1996     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1997     \tl_if_empty:NF \l_graphics_pagebox_tl
1998     {
1999       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
2000       bbox ~
2001       \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
2002       \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2003       \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2004       \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
2005     }
2006     (#1)
2007     \bool_lazy_or:nnT

```

```

2008         { \l_graphics_interpolate_bool }
2009         { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
2010         {
2011             <<
2012             \tl_if_empty:NF \l_graphics_decodearray_tl
2013             { /Decode~[ \l_graphics_decodearray_tl ] }
2014             \bool_if:NT \l_graphics_interpolate_bool
2015             { /Interpolate~true> }
2016             >>
2017         }
2018     }
2019 }

```

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

```

2020 </dvipdfmx | xetex>

```

## 5.4 X<sub>Y</sub>TeX backend

```

2021 <*xetex>

```

### 5.4.1 Images

For X<sub>Y</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 X<sub>Y</sub>TeX primitive omits the text box from the page box specification, so there is also some “trimming” to do here.

```

\__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_auxii:VnN
\__graphics_backend_getbb_auxiii:nNnn
\__graphics_backend_getbb_auxiv:nnNnn
\__graphics_backend_getbb_auxiv:VnNnn
\__graphics_backend_getbb_auxv:nNnn
\__graphics_backend_getbb_auxv:nNnn
\__graphics_backend_getbb_pagebox:w
2022 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2023 {
2024     \int_zero:N \l_graphics_page_int
2025     \tl_clear:N \l_graphics_pagebox_tl
2026     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2027 }
2028 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
2029 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2030 {
2031     \tl_clear:N \l_graphics_decodearray_tl
2032     \bool_set_false:N \l_graphics_interpolate_bool
2033     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2034 }
2035 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2036 {
2037     \int_compare:nNnTF \l_graphics_page_int > 1
2038     { \__graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
2039     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2040 }
2041 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2042 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2043 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2044 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2045 {
2046     \tl_if_empty:NTF \l_graphics_pagebox_tl
2047     { \__graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
2048     { \__graphics_backend_getbb_auxv:nNnn }
2049     {#1} #2 {#3} {#4}

```

```

2050 }
2051 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2052 {
2053   \use:x
2054   {
2055     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2056     { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
2057   }
2058 }
2059 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2060 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2061 {
2062   \graphics_bb_restore:nF {#1#3}
2063   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2064 }
2065 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2066 {
2067   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
2068   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2069   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2070   \graphics_bb_save:n {#1#3}
2071 }
2072 \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  
\\_\_graphics\_backend\_include\_bitmap\_quote:w

For PDF graphics, properly supporting the pagebox concept in X<sub>Y</sub>TeX is best done using the `\tex_XeTeXpdf file: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`.

```

2073 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2074 {
2075   \tex_XeTeXpdf file:D
2076   \__graphics_backend_include_pdf_quote:w #1 "#1" \s__graphics_stop \c_space_tl
2077   \int_compare:nNnT \l_graphics_page_int > 0
2078     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
2079   \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
2080 }
2081 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \s__graphics_stop
2082 { " #2 " }

```

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

```
2083 </xetex>
```

## 5.5 dvisvgm backend

```
2084 <*dvisvgm>
```

\\_\_graphics\_backend\_getbb\_eps:n

Simply use the generic function.

```
2085 \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`

```

2086 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
2087 {
2088   \int_zero:N \l_graphics_page_int
2089   \tl_clear:N \l_graphics_pagebox_tl
2090   \graphics_extract_bb:n {#1}
2091 }
2092 \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

```

2093 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
2094 {
2095   \tl_clear:N \l_graphics_decodearray_tl
2096   \bool_set_false:N \l_graphics_interpolate_bool
2097   \graphics_extract_bb:n {#1}
2098 }

```

(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  
`\_graphics_backend_include_pdf:n` is the same as the `dvips` code.)  
`\_graphics_backend_include:nn`

```

2099 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
2100 { \_graphics_backend_include:nn { PSfile } {#1} }
2101 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
2102 { \_graphics_backend_include:nn { pdffile } {#1} }
2103 \cs_new_protected:Npn \_graphics_backend_include:nn #1#2
2104 {
2105   \__kernel_backend_literal:x
2106   {
2107     #1 = #2 \c_space_tl
2108     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
2109     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2110     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2111     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
2112   }
2113 }

```

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

`\_graphics_backend_include_png:n` The backend here has built-in support for basic graphic inclusion (see `dvisvgm.def` for a  
`\_graphics_backend_include_jpg:n` more complex approach, needed if clipping, *etc.*, is covered at the graphic backend level).  
`\_graphics_backend_include_bitmap_quote:w` 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.

```

2114 \cs_new_protected:Npn \_graphics_backend_include_png:n #1
2115 {
2116   \__kernel_backend_literal:x
2117   {
2118     dvisvgm:img~
2119     \dim_to_decimal:n { \l_graphics_ury_dim } ~
2120     \dim_to_decimal:n { \l_graphics_ury_dim } ~

```

```

2121         \_graphics_backend_include_bitmap_quote:w #1 " #1 " \s__graphics_stop
2122     }
2123 }
2124 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_png:n
2125 \cs_new:Npn \_graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \s__graphics_stop
2126 { " #2 " }

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

2127 </divisvgn>
2128 </package>

```

## 6 l3backend-pdf Implementation

```

2129 <*package>
2130 <@=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

2131 \box_new:N \l__pdf_internal_box

(End definition for \l__pdf_internal_box.)

```

### 6.2 dvips backend

```

2132 <*dvips>

\_pdf_backend_pdfmark:n Used often enough it should be a separate function.
\_pdf_backend_pdfmark:x
2133 \cs_new_protected:Npn \_pdf_backend_pdfmark:n #1
2134 { \_kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2135 \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

2136 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2137 { \_pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2138 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2139 { \_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.

```
2140 \int_new:N \g__pdf_backend_object_int
2141 \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.

```
2142 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2143 {
2144   \int_gincr:N \g__pdf_backend_object_int
2145   \int_const:cn
2146     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2147     { \g__pdf_backend_object_int }
2148   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2149 }
2150 \cs_new:Npn \__pdf_backend_object_ref:n #1
2151 { { 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:nnn

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

```
2152 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2153 {
2154   \__pdf_backend_pdfmark:x
2155   {
2156     /_objdef ~ \__pdf_backend_object_ref:n {#1}
2157     /type
2158     \str_case:e:nn
2159       { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2160       {
2161         { array } { /array }
2162         { dict } { /dict }
2163         { fstream } { /stream }
2164         { stream } { /stream }
2165       }
2166     /OBJ
2167   }
2168   \use:c
2169     { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2170     { \__pdf_backend_object_ref:n {#1} } {#2}
2171 }
2172 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2173 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2174 {
2175   \__pdf_backend_pdfmark:x
2176   { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2177 }
2178 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2179 {
2180   \__pdf_backend_pdfmark:x
2181   { #1 << \exp_not:n {#2} >> /PUT }
2182 }
2183 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
```

```

2184 {
2185   \exp_args:Nx
2186   \__pdf_backend_object_write_fstream:nnn {#1} #2
2187 }
2188 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2189 {
2190   \__kernel_backend_postscript:n
2191   {
2192     SDict ~ begin ~
2193     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2194     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2195     end
2196   }
2197 }
2198 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2199 {
2200   \exp_args:Nx
2201   \__pdf_backend_object_write_stream:nnn {#1} #2
2202 }
2203 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2204 {
2205   \__kernel_backend_postscript:n
2206   {
2207     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2208     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2209   }
2210 }

```

(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.

```

2211 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2212 {
2213   \int_gincr:N \g__pdf_backend_object_int
2214   \__pdf_backend_pdfmark:x
2215   {
2216     /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2217     /type
2218     \str_case:nn
2219       {#1}
2220       {
2221         { array } { /array }
2222         { dict } { /dict }
2223         { fstream } { /stream }
2224         { stream } { /stream }
2225       }
2226     /OBJ
2227   }
2228   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2229   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2230 }
2231 \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.

```
2232 \cs_new:Npn \__pdf_backend_object_last:
2233   { { 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.

```
2234 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2235   { { 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.

```
2236 \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.

```
2237 \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.

```
2238 \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.

```
2239 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2240   {
2241     \exp_args:Nf \__pdf_backend_annotation_aux:nnnn
2242       { \dim_eval:n {#1} } {#2} {#3} {#4}
2243   }
2244 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2245   {
2246     \box_move_down:nn {#3}
2247     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2248     \box_move_up:nn {#2}
2249     {
2250       \hbox:n
2251       {
2252         \__kernel_kern:n {#1}
2253         \__kernel_backend_postscript:n { pdf.save.ur }
2254         \__kernel_kern:n { -#1 }
2255       }

```

```

2256     }
2257     \int_gincr:N \g__pdf_backend_object_int
2258     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2259     \__pdf_backend_pdfmark:x
2260     {
2261         /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2262         pdf.rect
2263         #4 ~
2264         /ANN
2265     }
2266 }

```

(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.

```

2267 \cs_new:Npn \__pdf_backend_annotation_last:
2268 { { 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.

```

2269 \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.

```

2270 \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.

```

2271 \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.

```

2272 \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.

```

2273 \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.

```

2274 \tl_new:N \l__pdf_breaklink_pdfmark_tl
2275 \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.

```

2276 \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.

```
2277 \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`.

Notice that the link setup here uses `/Action` not `/A`. That is because Distiller *requires* this trigger word, rather than a “raw” PDF dictionary key (Ghostscript can handle either form).

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 generic mode are still to re-examine.

```
2278 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2279 {
2280   \_pdf_backend_link_begin:nw
2281   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2282 }
2283 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2284 { \_pdf_backend_link_begin:nw {#1#2} }
2285 \cs_new_protected:Npn \_pdf_backend_link_begin:nw #1
2286 {
2287   \bool_if:NF \g__pdf_backend_link_bool
2288   { \_pdf_backend_link_begin_aux:nw {#1} }
2289 }
```

The definition of `pdf.link.dict` here is needed as there is code in the PostScript headers for breaking links, and that can only work with this available.

```
2290 \cs_new_protected:Npn \_pdf_backend_link_begin_aux:nw #1
2291 {
2292   \bool_gset_true:N \g__pdf_backend_link_bool
2293   \_kernel_backend_postscript:n
2294   { /pdf.link.dict ( #1 ) def }
2295   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2296   \_pdf_backend_link_sf_save:
2297   \mode_if_math:TF
2298   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2299   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2300   \hbox_set:Nw \l__pdf_backend_content_box
2301   \_pdf_backend_link_sf_restore:
2302   \bool_if:NT \g__pdf_backend_link_math_bool
2303   { \c_math_toggle_token }
2304 }
2305 \cs_new_protected:Npn \_pdf_backend_link_end:
```

```

2306 {
2307   \bool_if:NT \g__pdf_backend_link_bool
2308   { \__pdf_backend_link_end_aux: }
2309 }
2310 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2311 {
2312   \bool_if:NT \g__pdf_backend_link_math_bool
2313   { \c_math_toggle_token }
2314   \__pdf_backend_link_sf_save:
2315   \hbox_set_end:
2316   \__pdf_backend_link_minima:
2317   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2318   \exp_args:Nx \__pdf_backend_link_outerbox:n
2319   {
2320     \int_if_odd:nTF { \value { page } }
2321     { \oddsidemargin }
2322     { \evensidemargin }
2323   }
2324   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2325   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2326   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2327   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2328   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2329   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2330   {
2331     \hbox:n
2332     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2333   }
2334   \int_gincr:N \g__pdf_backend_object_int
2335   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2336   \__kernel_backend_postscript:x
2337   {
2338     mark
2339     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2340     \g__pdf_backend_link_dict_tl \c_space_tl
2341     pdf.rect
2342     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2343   }
2344   \__pdf_backend_link_sf_restore:
2345   \bool_gset_false:N \g__pdf_backend_link_bool
2346 }
2347 \cs_new_protected:Npn \__pdf_backend_link_minima:
2348 {
2349   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2350   \__kernel_backend_postscript:x
2351   {
2352     /pdf.linkdp.pad ~
2353     \dim_to_decimal:n
2354     {
2355       \dim_max:nn
2356       {
2357         \box_dp:N \l__pdf_backend_model_box
2358         - \box_dp:N \l__pdf_backend_content_box
2359       }

```

```

2360         { Opt }
2361     } ~
2362     pdf.pt.dvi ~ def
2363 /pdf.linkht.pad ~
2364 \dim_to_decimal:n
2365 {
2366     \dim_max:nn
2367     {
2368         \box_ht:N \l__pdf_backend_model_box
2369         - \box_ht:N \l__pdf_backend_content_box
2370     }
2371     { Opt }
2372 } ~
2373 pdf.pt.dvi ~ def
2374 }
2375 }
2376 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2377 {
2378     \__kernel_backend_postscript:x
2379     {
2380         /pdf.outerbox
2381         [
2382             \dim_to_decimal:n {#1} ~
2383             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2384             \dim_to_decimal:n { #1 + \textwidth } ~
2385             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2386         ]
2387         [ exch { pdf.pt.dvi } forall ] def
2388     /pdf.baselineskip ~
2389     \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2390     { pdf.pt.dvi ~ def }
2391     { pop ~ pop }
2392     ifelse
2393 }
2394 }
2395 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2396 {
2397     \int_gset:Nn \g__pdf_backend_link_sf_int
2398     {
2399         \mode_if_horizontal:TF
2400         { \tex_spacefactor:D }
2401         { 0 }
2402     }
2403 }
2404 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2405 {
2406     \mode_if_horizontal:T
2407     {
2408         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2409         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2410     }
2411 }

```

(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.

```

2412 \use_none:n
2413 {
2414   \cs_if_exist:NT \@makecol@hook
2415   {
2416     \tl_put_right:Nn \@makecol@hook
2417     {
2418       \box_if_empty:NF \@cclv
2419       {
2420         \vbox_set:Nn \@cclv
2421         {
2422           \__kernel_backend_postscript:n
2423           {
2424             pdf.globaldict /pdf.brokenlink.rect ~ known
2425             { pdf.bordertracking.continue }
2426             if
2427             }
2428             \vbox_unpack_drop:N \@cclv
2429             \__kernel_backend_postscript:n
2430             { pdf.bordertracking.endpage }
2431           }
2432         }
2433       }
2434       \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2435       \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2436       \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2437     }
2438   }

```

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

`\__pdf_backend_link_last:` The same as annotations, but with a custom integer.

```

2439 \cs_new:Npn \__pdf_backend_link_last:
2440 { { 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.

```

2441 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2442 {
2443   \__kernel_backend_postscript:x
2444   {
2445     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2446   }
2447 }

```

(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:nnnn` anchor point actually is: worked out in PostScript. For the rectangle version, we have a  
`\__pdf_backend_destination_aux:nnnn` bit more PostScript: we need two points. `ftr` without rule spec doesn't work, so it falls  
back to `/Fit` here.

```

2448 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2449 {
2450   \__kernel_backend_postscript:n { pdf.dest.anchor }
2451   \__pdf_backend_pdfmark:x
2452   {
2453     /View
2454     [
2455       \str_case:nnF {#2}
2456       {
2457         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2458         { fit } { /Fit }
2459         { fitb } { /FitB }
2460         { fitbh } { /FitBH ~ pdf.dest.y }
2461         { fitbv } { /FitBV ~ pdf.dest.x }
2462         { fith } { /FitH ~ pdf.dest.y }
2463         { fitv } { /FitV ~ pdf.dest.x }
2464         { fitr } { /Fit }
2465       }
2466       {
2467         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2468       }
2469     ]
2470     /Dest ( \exp_not:n {#1} ) cvn
2471     /DEST
2472   }
2473 }
2474 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2475 {
2476   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2477   { \dim_eval:n {#2} } {#1} {#3} {#4}
2478 }
2479 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2480 {
2481   \vbox_to_zero:n
2482   {
2483     \__kernel_kern:n {#4}
2484     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2485     \tex_vss:D
2486   }
2487   \__kernel_kern:n {#1}
2488   \vbox_to_zero:n
2489   {
2490     \__kernel_kern:n { -#3 }
2491     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2492     \tex_vss:D
2493   }
2494   \__kernel_kern:n { -#1 }
2495   \__pdf_backend_pdfmark:n
2496   {
2497     /View
2498     [
2499       /FitR ~
2500       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2501       pdf.urx ~ pdf.ury ~ pdf.dest2device

```

```

2502     ]
2503     /Dest ( #2 ) cvn
2504     /DEST
2505   }
2506 }

```

(End definition for `\_pdf_backend_destination:nn`, `\_pdf_backend_destination:nnnn`, and `\_pdf_backend_destination_aux:nnnn`.)

## 6.2.4 Structure

Doable for the usual ps2pdf method.

```

\_pdf_backend_compresslevel:n
\_pdf_backend_compress_objects:n
2507 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2508 {
2509   \int_compare:nNnT {#1} = 0
2510   {
2511     \__kernel_backend_literal_postscript:n
2512     {
2513       /setdistillerparams ~ where
2514       { pop << /CompressPages ~ false >> setdistillerparams }
2515       if
2516     }
2517   }
2518 }
2519 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2520 {
2521   \bool_if:nF {#1}
2522   {
2523     \__kernel_backend_literal_postscript:n
2524     {
2525       /setdistillerparams ~ where
2526       { pop << /CompressStreams ~ false >> setdistillerparams }
2527       if
2528     }
2529   }
2530 }

```

(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
2531 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2532 {
2533   \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2534 }
2535 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2536 {
2537   \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2538 }

```

(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:
2539 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2540 \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

Simple wrappers.

```

2541 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2542 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2543 \cs_new_protected:Npn \__pdf_backend_emc:
2544 { \__pdf_backend_pdfmark:n { /EMC } }

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

2545 </dvips>

```

## 6.3 LuaTeX and pdfTeX backend

```

2546 < *luatex | pdftex >

```

### 6.3.1 Annotations

Simply pass the raw data through, just dealing with evaluation of dimensions.

```

2547 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2548 {
2549 < *luatex >
2550 \tex_pdfextension:D annot ~
2551 < /luatex >
2552 < *pdftex >
2553 \tex_pdfannot:D
2554 < /pdftex >
2555 width ~ \dim_eval:n {#1} ~
2556 height ~ \dim_eval:n {#2} ~
2557 depth ~ \dim_eval:n {#3} ~
2558 {#4}
2559 }

```

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

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.

```

2560 \cs_new:Npx \__pdf_backend_annotation_last:
2561 {
2562 \exp_not:N \int_value:w
2563 < *luatex >
2564 \exp_not:N \tex_pdffeedback:D lastannot ~
2565 < /luatex >
2566 < *pdftex >
2567 \exp_not:N \tex_pdflastannot:D
2568 < /pdftex >
2569 \c_space_tl 0 ~ R
2570 }

```

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

Links are all created using the same internals.

```

2571 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2572 { \__pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2573 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2

```

```

2574 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2575 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2576 {
2577   \*luatex
2578   \tex_pdfextension:D startlink ~
2579   \*pdftex
2580   \tex_pdfstartlink:D
2581   \*pdftex
2582   attr {#1}
2583   #2 {#3}
2584 }
2585 \cs_new_protected:Npn \_pdf_backend_link_end:
2586 {
2587   \*luatex
2588   \tex_pdfextension:D endlink \scan_stop:
2589   \*pdftex
2590   \tex_pdfendlink:D
2591   \*pdftex
2592 }
2593
2594

```

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

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

```

2595 \cs_new:Npx \_pdf_backend_link_last:
2596 {
2597   \exp_not:N \int_value:w
2598   \*luatex
2599   \exp_not:N \tex_pdffeedback:D lastlink ~
2600   \*pdftex
2601   \exp_not:N \tex_pdflastlink:D
2602   \*pdftex
2603   \c_space_tl 0 ~ R
2604 }
2605

```

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

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

```

2606 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2607 {
2608   \*luatex
2609   \tex_pdfvariable:D linkmargin
2610   \*pdftex
2611   \tex_pdflinkmargin:D
2612   \*pdftex
2613   \dim_eval:n {#1} \scan_stop:
2614 }
2615

```

(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.

```

2616 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2617 {
2618   \*luatex
2619     \tex_pdfextension:D dest ~
2620   \*pdfTeX
2621     \tex_pdfdest:D
2622   \*pdfTeX
2623     name {#1}
2624     \str_case:nnF {#2}
2625     {
2626       { xyz } { xyz }
2627       { fit } { fit }
2628       { fitb } { fitb }
2629       { fitbh } { fitbh }
2630       { fitbv } { fitbv }
2631       { fith } { fith }
2632       { fitv } { fitv }
2633       { fitr } { fitr }
2634     }
2635     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2636     \scan_stop:
2637   }
2638 }
2639 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2640 {
2641   \*luatex
2642     \tex_pdfextension:D dest ~
2643   \*pdfTeX
2644     \tex_pdfdest:D
2645     name {#1}
2646     fitr ~
2647     width \dim_eval:n {#2} ~
2648     height \dim_eval:n {#3} ~
2649     depth \dim_eval:n {#4} \scan_stop:
2650   }
2651 }
2652 }

```

(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`

```

2653 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2654 {
2655   \*luatex
2656     \tex_pdfextension:D catalog
2657   \*pdfTeX
2658     \tex_pdfcatalog:D
2659   \*pdfTeX
2660   }

```

```

2661     { / #1 ~ #2 }
2662   }
2663   \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2664   {
2665     <*luatex>
2666       \tex_pdfextension:D info
2667     </luatex>
2668     <*pdftex>
2669       \tex_pdfinfo:D
2670     </pdftex>
2671     { / #1 ~ #2 }
2672   }

```

(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.

```

2673 \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

```

2674 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2675 {
2676   <*luatex>
2677     \tex_pdfextension:D obj ~
2678   </luatex>
2679   <*pdftex>
2680     \tex_pdfobj:D
2681   </pdftex>
2682     reserveobjnum ~
2683     \int_const:cn
2684     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2685   <*luatex>
2686     { \tex_pdffeedback:D lastobj }
2687   </luatex>
2688   <*pdftex>
2689     { \tex_pdflastobj:D }
2690   </pdftex>
2691     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2692   }
2693   \cs_new:Npn \__pdf_backend_object_ref:n #1
2694   { \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

(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

\\_\_pdf\_exp\_not\_i:nn

\\_\_pdf\_exp\_not\_ii:nn

```

2695 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2696 {
2697   <*luatex>
2698     \tex_immediate:D \tex_pdfextension:D obj ~
2699   </luatex>
2700   <*pdftex>
2701     \tex_immediate:D \tex_pdfobj:D

```

```

2702 </pdfTeX>
2703     useobjnum ~
2704     \int_use:c
2705     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2706     \str_case_e:nn
2707     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2708     {
2709       { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2710       { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2711       { fstream }
2712       {
2713         stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2714         file ~ { \__pdf_exp_not_ii:nn #2 }
2715       }
2716       { stream }
2717       {
2718         stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2719         { \__pdf_exp_not_ii:nn #2 }
2720       }
2721     }
2722   }
2723   \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2724   \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2725   \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
2726 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2727 {
2728   <*luatex>
2729   \tex_immediate:D \tex_pdfextension:D obj ~
2730   </luatex>
2731   <*pdfTeX>
2732   \tex_immediate:D \tex_pdfobj:D
2733   </pdfTeX>
2734   \str_case:nn
2735   {#1}
2736   {
2737     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2738     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2739     { fstream }
2740     {
2741       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2742       file ~ { \__pdf_exp_not_ii:nn #2 }
2743     }
2744     { stream }
2745     {
2746       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2747       { \__pdf_exp_not_ii:nn #2 }
2748     }
2749   }
2750 }
2751 \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.

```

2752 \cs_new:Npx \_pdf_backend_object_last:
2753 {
2754   \exp_not:N \int_value:w
2755   <*luatex>
2756   \exp_not:N \tex_pdffeedback:D lastobj ~
2757   </luatex>
2758   <*pdftex>
2759   \exp_not:N \tex_pdflastobj:D
2760   </pdftex>
2761   \c_space_tl 0 ~ R
2762 }

```

(End definition for `\_pdf_backend_object_last:.`)

`\_pdf_backend_pageobject_ref:n` The usual wrapper situation; the three spaces here are essential.

```

2763 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2764 {
2765   \exp_not:N \int_value:w
2766   <*luatex>
2767   \exp_not:N \tex_pdffeedback:D pageref
2768   </luatex>
2769   <*pdftex>
2770   \exp_not:N \tex_pdfpageref:D
2771   </pdftex>
2772   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2773 }

```

(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
\_pdf_backend_objcompresslevel:n
2774 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2775 {
2776   \tex_global:D
2777   <*luatex>
2778   \tex_pdfvariable:D compresslevel
2779   </luatex>
2780   <*pdftex>
2781   \tex_pdfcompresslevel:D
2782   </pdftex>
2783   \int_value:w \int_eval:n {#1} \scan_stop:
2784 }
2785 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2786 {
2787   \bool_if:nTF {#1}
2788   { \_pdf_backend_objcompresslevel:n { 2 } }
2789   { \_pdf_backend_objcompresslevel:n { 0 } }
2790 }
2791 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2792 {

```

```

2793     \tex_global:D
2794   \<luatex>
2795     \tex_pdfvariable:D objcompresslevel
2796   \</luatex>
2797   \<pdftex>
2798     \tex_pdfobjcompresslevel:D
2799   \</pdftex>
2800     #1 \scan_stop:
2801   }

(End definition for \__pdf_backend_compresslevel:n, \__pdf_backend_compress_objects:n, and \__-
pdf_backend_objcompresslevel:n.)

```

\\_\_pdf\_backend\_version\_major\_gset:n The availability of the primitive is not universal, so we have to test at load time.

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

```

2802 \cs_new_protected:Npx \__pdf_backend_version_major_gset:n #1
2803 {
2804   \<luatex>
2805     \int_compare:nNnT \tex luatexversion:D > { 106 }
2806     {
2807       \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2808       \exp_not:N \int_eval:n {#1} \scan_stop:
2809     }
2810   \</luatex>
2811   \<pdftex>
2812     \cs_if_exist:NT \tex_pdfmajorversion:D
2813     {
2814       \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2815       \exp_not:N \int_eval:n {#1} \scan_stop:
2816     }
2817   \</pdftex>
2818 }
2819 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2820 {
2821   \tex_global:D
2822   \<luatex>
2823     \tex_pdfvariable:D minorversion
2824   \</luatex>
2825   \<pdftex>
2826     \tex_pdfminorversion:D
2827   \</pdftex>
2828     \int_eval:n {#1} \scan_stop:
2829 }

```

(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:

```

2830 \cs_new:Npx \__pdf_backend_version_major:
2831 {
2832   \<luatex>
2833     \int_compare:nNnTF \tex luatexversion:D > { 106 }
2834     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2835     { 1 }
2836   \</luatex>
2837   \<pdftex>
2838     \cs_if_exist:NTF \tex_pdfmajorversion:D

```

```

2839     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2840     { 1 }
2841   \</pdfTeX>
2842 }
2843 \cs_new:Npn \__pdf_backend_version_minor:
2844 {
2845   \tex_the:D
2846   \*luatex
2847   \tex_pdfvariable:D minorversion
2848   \</luatex>
2849   \*pdfTeX
2850   \tex_pdfminorversion:D
2851   \</pdfTeX>
2852 }

```

(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>.

```

2853 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2854 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2855 \cs_new_protected:Npn \__pdf_backend_emc:
2856 { \__kernel_backend_literal_page:n { EMC } }

```

(End definition for \\_\_pdf\_backend\_bdc:nn and \\_\_pdf\_backend\_emc:.)

```

2857 \</luatex | pdfTeX>

```

## 6.4 dvipdfmx backend

```

2858 \*dvipdfmx | xetex>

```

\\_\_pdf\_backend:n A generic function for the backend PDF specials: used where we can.

```

\__pdf_backend:x
2859 \cs_new_protected:Npx \__pdf_backend:n #1
2860 { \__kernel_backend_literal:n { pdf: #1 } }
2861 \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
2862 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2863 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2864 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2865 { \__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.

```
2866 \int_new:N \g__pdf_backend_object_int
2867 \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.

```
2868 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2869 {
2870   \int_gincr:N \g__pdf_backend_object_int
2871   \int_const:cn
2872     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2873     { \g__pdf_backend_object_int }
2874   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2875 }
2876 \cs_new:Npn \__pdf_backend_object_ref:n #1
2877 { @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.

```
2878 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2879 {
2880   \exp_args:Nx \__pdf_backend_object_write:nnn
2881     { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2882 }
2883 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2884 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2885 {
2886   \use:c { __pdf_backend_object_write_ #1 :nn }
2887   { \__pdf_backend_object_ref:n {#2} } {#3}
2888 }
2889 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2890 {
2891   \__pdf_backend:x
2892     { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2893 }
2894 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2895 {
2896   \__pdf_backend:x
2897     { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2898 }
2899 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2900 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2901 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2902 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2903 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2904 {
2905   \__pdf_backend:x
2906     {
2907       #1 stream ~ #2 ~
2908       ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2909     }
2910 }
```

```
2910 }
```

(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
2911 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2912 {
2913   \int_gincr:N \g__pdf_backend_object_int
2914   \exp_args:Nnx \use:c { \_pdf_backend_object_write_ #1 :nn }
2915   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2916   {#2}
2917 }
2918 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }
```

(End definition for `\_pdf_backend_object_now:nn`.)

`\_pdf_backend_object_last:`

```
2919 \cs_new:Npn \_pdf_backend_object_last:
2920 { @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.

```
2921 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2922 { @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.

```
2923 \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.

```
2924 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2925 {
2926   \int_gincr:N \g__pdf_backend_object_int
2927   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2928   \_pdf_backend:x
2929   {
2930     ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
2931     width ~ \dim_eval:n {#1} ~
2932     height ~ \dim_eval:n {#2} ~
2933     depth ~ \dim_eval:n {#3} ~
2934     << /Type /Annot #4 >>
2935   }
2936 }
```

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

`\_pdf_backend_annotation_last:`

```
2937 \cs_new:Npn \_pdf_backend_annotation_last:
2938 { @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.

2939 \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

2940 \cs\_new\_protected:Npn \\_pdf\_backend\_link\_begin\_goto:nnw #1#2

2941 { \\_pdf\_backend\_link\_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }

\\_pdf\_backend\_link\_begin:n

2942 \cs\_new\_protected:Npn \\_pdf\_backend\_link\_begin\_user:nnw #1#2

\\_pdf\_backend\_link\_end:

2943 { \\_pdf\_backend\_link\_begin:n {#1#2} }

2944 \cs\_new\_protected:Npx \\_pdf\_backend\_link\_begin:n #1

2945 {

2946 \int\_compare:nNnF \c\_kernel\_sys\_dvipdfmx\_version\_int < { 20201111 }

2947 {

2948 \exp\_not:N \int\_gincr:N \exp\_not:N \g\_pdf\_backend\_link\_int

2949 }

2950 \\_pdf\_backend:x

2951 {

2952 bann ~

2953 \int\_compare:nNnF \c\_kernel\_sys\_dvipdfmx\_version\_int < { 20201111 }

2954 {

2955 @pdf.lnk

2956 \exp\_not:N \int\_use:N \exp\_not:N \g\_pdf\_backend\_link\_int

2957 \c\_space\_tl

2958 }

2959 <<

2960 /Type /Annot

2961 #1

2962 >>

2963 }

2964 }

2965 \cs\_new\_protected:Npn \\_pdf\_backend\_link\_end:

2966 { \\_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.

2967 \cs\_new:Npx \\_pdf\_backend\_link\_last:

2968 {

2969 \int\_compare:nNnF \c\_kernel\_sys\_dvipdfmx\_version\_int < { 20201111 }

2970 {

2971 @pdf.lnk

2972 \exp\_not:N \int\_use:N \exp\_not:N \g\_pdf\_backend\_link\_int

2973 }

2974 }

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

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

2975 \cs\_new\_protected:Npn \\_pdf\_backend\_link\_margin:n #1

2976 { \\_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  $\text{\TeX}$  by using the backend data for `@xpos` and `@ypos`. `/FitR` without rule spec doesn't work, so it falls back to `/Fit` here.

```

2977 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2978 {
2979   \_pdf_backend:x
2980   {
2981     dest ~ ( \exp_not:n {#1} )
2982     [
2983       @thispage
2984       \str_case:nnF {#2}
2985       {
2986         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2987         { fit } { /Fit }
2988         { fitb } { /FitB }
2989         { fitbh } { /FitBH }
2990         { fitbv } { /FitBV ~ @xpos }
2991         { fith } { /FitH ~ @ypos }
2992         { fitv } { /FitV ~ @xpos }
2993         { fitr } { /Fit }
2994       }
2995       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2996     ]
2997   }
2998 }
2999 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
3000 {
3001   \exp_args:Ne \_pdf_backend_destination_aux:nnnn
3002   { \dim_eval:n {#2} } {#1} {#3} {#4}
3003 }
3004 \cs_new_protected:Npn \_pdf_backend_destination_aux:nnnn #1#2#3#4
3005 {
3006   \vbox_to_zero:n
3007   {
3008     \_kernel_kern:n {#4}
3009     \hbox:n
3010     {
3011       \_pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
3012       \_pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
3013     }
3014     \tex_vss:D
3015   }
3016   \_kernel_kern:n {#1}
3017   \vbox_to_zero:n
3018   {
3019     \_kernel_kern:n { -#3 }
3020     \hbox:n
3021     {
3022       \_pdf_backend:n
3023       {
3024         dest ~ (#2)
3025         [
3026           @thispage

```

```

3027             /FitR ~
3028             @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
3029             @xpos ~ @ypos
3030         ]
3031     }
3032 }
3033 \tex_vss:D
3034 }
3035 \__kernel_kern:n { -#1 }
3036 }

```

(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.

```

3037 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
3038 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
3039 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
3040 {
3041     \bool_if:nF {#1}
3042     { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
3043 }

```

(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.

```

3044 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
3045 {
3046     \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
3047     \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
3048 }
3049 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
3050 {
3051     \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
3052     \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
3053 }

```

(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.

```

3054 \cs_new:Npn \__pdf_backend_version_major: { 1 }
3055 \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  
\\_\_pdf\_backend\_emc:

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

```

3056 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
3057 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
3058 \cs_new_protected:Npn \__pdf_backend_emc:
3059 { \__kernel_backend_literal_page:n { EMC } }

```

(End definition for \\_pdf\_backend\_bdc:nn and \\_pdf\_backend\_emc:.)

3060 </dvipdfmx | xetex>

## 6.5 dvisvgm backend

3061 <\*dvisvgm>

### 6.5.1 Catalogue entries

No-op.

3062 \cs\_new\_protected:Npn \\_pdf\_backend\_catalog\_gput:nn #1#2 { }

3063 \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

All no-ops here.

3064 \cs\_new\_protected:Npn \\_pdf\_backend\_object\_new:nn #1#2 { }

3065 \cs\_new:Npn \\_pdf\_backend\_object\_ref:n #1 { }

3066 \cs\_new\_protected:Npn \\_pdf\_backend\_object\_write:nn #1#2 { }

3067 \cs\_new\_protected:Npn \\_pdf\_backend\_object\_write:nx #1#2 { }

3068 \cs\_new\_protected:Npn \\_pdf\_backend\_object\_now:nn #1#2 { }

3069 \cs\_new\_protected:Npn \\_pdf\_backend\_object\_now:nx #1#2 { }

3070 \cs\_new:Npn \\_pdf\_backend\_object\_last: { }

3071 \cs\_new:Npn \\_pdf\_backend\_pageobject\_ref:n #1 { }

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

### 6.5.3 Structure

These are all no-ops.

3072 \cs\_new\_protected:Npn \\_pdf\_backend\_compresslevel:n #1 { }

3073 \cs\_new\_protected:Npn \\_pdf\_backend\_compress\_objects:n #1 { }

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

Data not available!

3074 \cs\_new\_protected:Npn \\_pdf\_backend\_version\_major\_gset:n #1 { }

3075 \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.)

Data not available!

3076 \cs\_new:Npn \\_pdf\_backend\_version\_major: { -1 }

3077 \cs\_new:Npn \\_pdf\_backend\_version\_minor: { -1 }

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

More no-ops.

3078 \cs\_new\_protected:Npn \\_pdf\_backend\_bdc:nn #1#2 { }

3079 \cs\_new\_protected:Npn \\_pdf\_backend\_emc: { }

(End definition for \\_pdf\_backend\_bdc:nn and \\_pdf\_backend\_emc:.)

3080 </dvisvgm>

3081 </package>

## 7 l3backend-opacity Implementation

```

3082 <*package>
3083 <@@=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.

```

3084 <*dvips>

```

No stack so set values directly. The need to deal with Distiller and Ghostscript separately means we use a common auxiliary: the two systems require different PostScript for transparency. This is of course not quite as efficient as doing one test for setting all transparency, but it keeps things clearer here. Thanks to Alex Grahn for the detail on testing for GhostScript.

```

3085 \cs_new_protected:Npn \__opacity_backend_select:n #1
3086 {
3087   \exp_args:Nx \__opacity_backend_select_aux:n
3088   { \fp_eval:n { min(max(0,#1),1) } }
3089 }
3090 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3091 {
3092   \__opacity_backend:nnn {#1} { fill } { ca }
3093   \__opacity_backend:nnn {#1} { stroke } { CA }
3094 }
3095 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3096 {
3097   \__opacity_backend:xnn
3098   { \fp_eval:n { min(max(0,#1),1) } }
3099   { fill }
3100   { ca }
3101 }
3102 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3103 {
3104   \__opacity_backend:xnn
3105   { \fp_eval:n { min(max(0,#1),1) } }
3106   { stroke }
3107   { CA }
3108 }
3109 \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3110 {
3111   \__kernel_backend_postscript:n
3112   {
3113     product ~ (Ghostscript) ~ search
3114     {
3115       pop ~ pop ~ pop ~
3116       #1 ~ .set #2 constantalpha
3117     }
3118     {
3119       pop ~
3120       mark ~
3121       /#3 ~ #1

```

```

3122         /SetTransparency ~
3123         pdfmark
3124     }
3125     ifelse
3126 }
3127 }
3128 \cs_generate_variant:Nn \_opacity_backend:nnn { x }

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

3129 </dvips>
3130 <*dvipdfmx | luatex | pdftex | xetex>

```

\c\_\_opacity\_backend\_stack\_int Set up a stack.

```

3131 \bool_lazy_and:nnT
3132 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3133 { \pdfmanagement_if_active_p:}
3134 {
3135     \_kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3136     { page ~ direct } { /opacity 1 ~ gs }
3137     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3138     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3139 }

(End definition for \c__opacity_backend_stack_int.)

```

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

```

\l__opacity_backend_stroke_tl
3140 \tl_new:N \l__opacity_backend_fill_tl
3141 \tl_new:N \l__opacity_backend_stroke_tl

```

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

\\_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:
3142 \cs_new_protected:Npn \_opacity_backend_select:n #1
3143 {
3144     \exp_args:Nx \_opacity_backend_select_aux:n
3145     { \fp_eval:n { min(max(0,#1),1) } }
3146 }
3147 \cs_new_protected:Npn \_opacity_backend_select_aux:n #1
3148 {
3149     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3150     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3151     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3152     { opacity #1 }
3153     { << /ca ~ #1 /CA ~ #1 >> }
3154     \_kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3155     { /opacity #1 ~ gs }
3156     \group_insert_after:N \_opacity_backend_reset:
3157 }
3158 \bool_lazy_and:nnF
3159 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3160 { \pdfmanagement_if_active_p:}
3161 {
3162     \cs_gset_protected:Npn \_opacity_backend_select_aux:n #1 { }
3163 }

```

```

3164 \cs_new_protected:Npn \__opacity_backend_reset:
3165 { \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int }

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

```

```

\__opacity_backend_fill:n
\__opacity_backend_stroke:n
  \__opacity_backend_fillstroke:nn
  \__opacity_backend_fillstroke:xx

```

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.

```

3166 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3167 {
3168   \__opacity_backend_fill_stroke:xx
3169   { \fp_eval:n { min(max(0,#1),1) } }
3170   \l__opacity_backend_stroke_tl
3171 }
3172 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3173 {
3174   \__opacity_backend_fill_stroke:xx
3175   \l__opacity_backend_fill_tl
3176   { \fp_eval:n { min(max(0,#1),1) } }
3177 }
3178 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3179 {
3180   \str_if_eq:nnTF {#1} {#2}
3181   { \__opacity_backend_select_aux:n {#1} }
3182   {
3183     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3184     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3185     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3186     { opacity.fill #1 }
3187     { << /ca ~ #1 >> }
3188     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3189     { opacity.stroke #1 }
3190     { << /CA ~ #2 >> }
3191     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3192     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3193     \group_insert_after:N \__opacity_backend_reset:
3194   }
3195 }
3196 \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.)

```

3197 </dvipdfmx | luatex | pdftex | xetex>
3198 <*dvipdfmx | xdvipdfmx>

```

```
\__opacity_backend_select:n
```

Older backends have no stack support, so everything is done directly.

```

3199 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
3200 {
3201   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1
3202   {
3203     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3204     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3205     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3206     { opacity #1 }

```

```

3207         { << /ca ~ #1 /CA ~ #1 >> }
3208         \__kernel_backend_literal_pdf:n { /opacity #1 ~ gs }
3209     }
3210     \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3211     {
3212         \str_if_eq:nnTF {#1} {#2}
3213         { \__opacity_backend_select_aux:n {#1} }
3214         {
3215             \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3216             \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3217             \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3218             { opacity.fill #1 }
3219             { << /ca ~ #1 >> }
3220             \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3221             { opacity.stroke #1 }
3222             { << /CA ~ #2 >> }
3223             \__kernel_backend_literal_pdf:n
3224             { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3225         }
3226     }
3227 }

```

(End definition for \\_\_opacity\_backend\_select:n.)

```

3228 </dvipdfmx | xdvipdfmx>

```

```

3229 <*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
3230 \cs_new_protected:Npn \__opacity_backend_select:n #1
3231 { \__opacity_backend:nn {#1} { } }
3232 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3233 { \__opacity_backend:nn {#1} { fill- } }
3234 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3235 { \__opacity_backend:nn { {#1} } { stroke- } }
3236 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3237 { \__kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

```

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

```

3238 </dvisvgm>

```

```

3239 </package>

```

## 8 l3backend-header Implementation

```

3240 <*dvips & header>

```

color.sc Empty definition for color at the top level.

```

3241 /color.sc { } def

```

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

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

```

3242 TeXDict begin
3243 /TeXcolorseparation { setcolor } def
3244 end

```

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

**pdf.globaldict** A small global dictionary for backend use.

```

3245 true setglobal
3246 /pdf.globaldict 4 dict def
3247 false setglobal

```

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

**pdf.cvs pdf.dvi.pt pdf.pt.dvi pdf.rect.ht** Small utilities for PostScript manipulations. Conversion to DVI dimensions is done here to allow for Resolution. The total height of a rectangle (an array) needs a little maths, in contrast to simply extracting a value.

```

3248 /pdf.cvs { 65534 string cvs } def
3249 /pdf.dvi.pt { 72.27 mul Resolution div } def
3250 /pdf.pt.dvi { 72.27 div Resolution mul } def
3251 /pdf.rect.ht { dup 1 get neg exch 3 get add } def

```

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

**pdf.linkmargin** Settings which are defined up-front in SDict.

```

3252 /pdf.linkmargin { 1 pdf.pt.dvi } def
3253 /pdf.linkdp.pad { 0 } def
3254 /pdf.linkht.pad { 0 } def

```

*(End definition for pdf.linkmargin, pdf.linkdp.pad, and pdf.linkht.pad. These functions are documented on page ??.)*

**pdf.rect pdf.save.ll pdf.save.ur** Functions for marking the limits of an annotation/link, plus drawing the border. We separate links for generic annotations to support adding a margin and setting a minimal size.

```

3255 /pdf.rect
3256 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
3257 /pdf.save.ll
3258 {
3259     currentpoint
3260     /pdf.lly exch def
3261     /pdf.llx exch def
3262 }
3263 def
3264 /pdf.save.ur
3265 {
3266     currentpoint
3267     /pdf.ury exch def
3268     /pdf.urx exch def
3269 }
3270 def
3271 /pdf.save.linkll
3272 {

```

```

3273     currentpoint
3274     pdf.linkmargin add
3275     pdf.linkdp.pad add
3276     /pdf.lly exch def
3277     pdf.linkmargin sub
3278     /pdf.llx exch def
3279   }
3280   def
3281 /pdf.save.linkur
3282 {
3283     currentpoint
3284     pdf.linkmargin sub
3285     pdf.linkht.pad sub
3286     /pdf.ury exch def
3287     pdf.linkmargin add
3288     /pdf.urx exch def
3289 }
3290 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 3291 /pdf.dest.anchor
pdf.dev.y 3292 {
pdf.tmpa 3293     currentpoint exch
pdf.tmpb 3294     pdf.dvi.pt 72 add
pdf.tmpc 3295     /pdf.dest.x exch def
pdf.tmpd 3296     pdf.dvi.pt
3297     vsize 72 sub exch sub
3298     /pdf.dest.y exch def
3299 }
3300 def
3301 /pdf.dest.point
3302 { pdf.dest.x pdf.dest.y } def
3303 /pdf.dest2device
3304 {
3305     /pdf.dest.y exch def
3306     /pdf.dest.x exch def
3307     matrix currentmatrix
3308     matrix defaultmatrix
3309     matrix invertmatrix
3310     matrix concatmatrix
3311     cvx exec
3312     /pdf.dev.y exch def
3313     /pdf.dev.x exch def
3314     /pdf.tmpd exch def
3315     /pdf.tmpc exch def
3316     /pdf.tmpb exch def
3317     /pdf.tmpa exch def
3318     pdf.dest.x pdf.tmpa mul

```

```

3319     pdf.dest.y pdf.tmpc mul add
3320     pdf.dev.x add
3321 pdf.dest.x pdf.tmpb mul
3322     pdf.dest.y pdf.tmpd mul add
3323     pdf.dev.y add
3324 }
3325 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
pdf.bordertracking.begin can be done by hooking into a and x operations: those names have to be retained. The
pdf.bordertracking.end boundary is stored at the end of the operation. Special effort is needed at the start and
pdf.leftboundary end of pages (or rather galleys), such that everything works properly.
pdf.rightboundary
pdf.brokenlink.rect 3326 /pdf.bordertracking false def
pdf.brokenlink.skip 3327 /pdf.bordertracking.begin
pdf.brokenlink.dict 3328 {
3329     SDict /pdf.bordertracking true put
3330     SDict /pdf.leftboundary undef
3331     SDict /pdf.rightboundary undef
3332     /a where
3333     {
3334         /a
3335         {
3336             currentpoint pop
3337             SDict /pdf.rightboundary known dup
3338             {
3339                 SDict /pdf.rightboundary get 2 index lt
3340                 { not }
3341                 if
3342             }
3343             if
3344             { pop }
3345             { SDict exch /pdf.rightboundary exch put }
3346             ifelse
3347             moveto
3348             currentpoint pop
3349             SDict /pdf.leftboundary known dup
3350             {
3351                 SDict /pdf.leftboundary get 2 index gt
3352                 { not }
3353                 if
3354             }
3355             if
3356             { pop }
3357             { SDict exch /pdf.leftboundary exch put }
3358             ifelse
3359         }
3360         put
3361     }
3362     if
3363 }
3364 def
3365 /pdf.bordertracking.end

```

```

3366 {
3367   /a where { /a { moveto } put } if
3368   /x where { /x { 0 exch rmoveto } put } if
3369   SDict /pdf.leftboundary known
3370     { pdf.outerbox 0 pdf.leftboundary put }
3371   if
3372   SDict /pdf.rightboundary known
3373     { pdf.outerbox 2 pdf.rightboundary put }
3374   if
3375   SDict /pdf.bordertracking false put
3376 }
3377 def
3378 /pdf.bordertracking.endpage
3379 {
3380   pdf.bordertracking
3381   {
3382     pdf.bordertracking.end
3383     true setglobal
3384     pdf.globaldict
3385       /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3386     pdf.globaldict
3387       /pdf.brokenlink.skip pdf.baselineskip put
3388     pdf.globaldict
3389       /pdf.brokenlink.dict
3390         pdf.link.dict pdf.cvs put
3391     false setglobal
3392     mark pdf.link.dict cvx exec /Rect
3393     [
3394       pdf.llx
3395       pdf.lly
3396       pdf.outerbox 2 get pdf.linkmargin add
3397       currentpoint exch pop
3398       pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3399     ]
3400     /ANN pdf.pdfmark
3401   }
3402   if
3403 }
3404 def
3405 /pdf.bordertracking.continue
3406 {
3407   /pdf.link.dict pdf.globaldict
3408   /pdf.brokenlink.dict get def
3409   /pdf.outerbox pdf.globaldict
3410   /pdf.brokenlink.rect get def
3411   /pdf.baselineskip pdf.globaldict
3412   /pdf.brokenlink.skip get def
3413   pdf.globaldict dup dup
3414   /pdf.brokenlink.dict undef
3415   /pdf.brokenlink.skip undef
3416   /pdf.brokenlink.rect undef
3417   currentpoint
3418   /pdf.originy exch def
3419   /pdf.originx exch def

```

```

3420     /a where
3421     {
3422         /a
3423         {
3424             moveto
3425             SDict
3426             begin
3427                 currentpoint pdf.originy ne exch
3428                 pdf.originx ne or
3429                 {
3430                     pdf.save.link11
3431                     /pdf.lly
3432                     pdf.lly pdf.outerbox 1 get sub def
3433                     pdf.bordertracking.begin
3434                 }
3435                 if
3436                 end
3437             }
3438             put
3439         }
3440     if
3441     /x where
3442     {
3443         /x
3444         {
3445             0 exch rmoveto
3446             SDict
3447             begin
3448                 currentpoint
3449                 pdf.originy ne exch pdf.originx ne or
3450                 {
3451                     pdf.save.link11
3452                     /pdf.lly
3453                     pdf.lly pdf.outerbox 1 get sub def
3454                     pdf.bordertracking.begin
3455                 }
3456                 if
3457                 end
3458             }
3459             put
3460         }
3461     if
3462 }
3463 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.

```

3464 /pdf.breaklink
3465 {

```

```

3466 pop
3467 counttomark 2 mod 0 eq
3468 {
3469     counttomark /pdf.count exch def
3470     {
3471         pdf.count 0 eq { exit } if
3472         counttomark 2 roll
3473         1 index /Rect eq
3474         {
3475             dup 4 array copy
3476             dup dup
3477                 1 get
3478                 pdf.outerbox pdf.rect.ht
3479                 pdf.linkmargin 2 mul add sub
3480                 3 exch put
3481             dup
3482                 pdf.outerbox 2 get
3483                 pdf.linkmargin add
3484                 2 exch put
3485             dup dup
3486                 3 get
3487                 pdf.outerbox pdf.rect.ht
3488                 pdf.linkmargin 2 mul add add
3489                 1 exch put
3490             /pdf.currentrect exch def
3491             pdf.breaklink.write
3492             {
3493                 pdf.currentrect
3494                 dup
3495                     pdf.outerbox 0 get
3496                     pdf.linkmargin sub
3497                     0 exch put
3498                 dup
3499                     pdf.outerbox 2 get
3500                     pdf.linkmargin add
3501                     2 exch put
3502                 dup dup
3503                     1 get
3504                     pdf.baselineskip add
3505                     1 exch put
3506                 dup dup
3507                     3 get
3508                     pdf.baselineskip add
3509                     3 exch put
3510                 /pdf.currentrect exch def
3511                 pdf.breaklink.write
3512             }
3513             1 index 3 get
3514             pdf.linkmargin 2 mul add
3515             pdf.outerbox pdf.rect.ht add
3516             2 index 1 get sub
3517             pdf.baselineskip div round cvi 1 sub
3518             exch
3519             repeat

```

```

3520         pdf.currentrect
3521         dup
3522             pdf.outerbox 0 get
3523             pdf.linkmargin sub
3524             0 exch put
3525         dup dup
3526             1 get
3527             pdf.baselineskip add
3528             1 exch put
3529         dup dup
3530             3 get
3531             pdf.baselineskip add
3532             3 exch put
3533         dup 2 index 2 get 2 exch put
3534         /pdf.currentrect exch def
3535         pdf.breaklink.write
3536         SDict /pdf.pdfmark.good false put
3537         exit
3538     }
3539     { pdf.count 2 sub /pdf.count exch def }
3540     ifelse
3541 }
3542 loop
3543 }
3544 if
3545 /ANN
3546 }
3547 def
3548 /pdf.breaklink.write
3549 {
3550     counttomark 1 sub
3551     index /_objdef eq
3552     {
3553         counttomark -2 roll
3554         dup wcheck
3555         {
3556             readonly
3557             counttomark 2 roll
3558         }
3559         { pop pop }
3560     } ifelse
3561 }
3562 if
3563 counttomark 1 add copy
3564 pop pdf.currentrect
3565 /ANN pdfmark
3566 }
3567 def

```

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

```

pdf.pdfmark
pdf.pdfmark.good
pdf.outerbox
pdf.baselineskip
pdf.pdfmark.dict

```

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.

```

3568 /pdf.pdfmark
3569 {
3570   SDict /pdf.pdfmark.good true put
3571   dup /ANN eq
3572   {
3573     pdf.pdfmark.store
3574     pdf.pdfmark.dict
3575     begin
3576       Subtype /Link eq
3577       currentdict /Rect known and
3578       SDict /pdf.outerbox known and
3579       SDict /pdf.baselineskip known and
3580       {
3581         Rect 3 get
3582         pdf.linkmargin 2 mul add
3583         pdf.outerbox pdf.rect.ht add
3584         Rect 1 get sub
3585         pdf.baselineskip div round cvi 0 gt
3586         { pdf.breaklink }
3587         if
3588       }
3589       if
3590     end
3591     SDict /pdf.outerbox undef
3592     SDict /pdf.baselineskip undef
3593     currentdict /pdf.pdfmark.dict undef
3594   }
3595   if
3596   pdf.pdfmark.good
3597   { pdfmark }
3598   { cleartomark }
3599   ifelse
3600 }
3601 def
3602 /pdf.pdfmark.store
3603 {
3604   /pdf.pdfmark.dict 65534 dict def
3605   counttomark 1 add copy
3606   pop
3607   {
3608     dup mark eq
3609     {
3610       pop
3611       exit
3612     }
3613     {
3614       pdf.pdfmark.dict
3615       begin def end
3616     }
3617     ifelse
3618   }
3619   loop

```

```
3620 }  
3621 def  
  
(End definition for pdf.pdfmark and others. These functions are documented on page ??.)  
3622  $\langle$ /dvips & header $\rangle$ 
```

# 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.

- \\_ ..... 154
- A**
- \AtBeginDvi ..... 57
- B**
- bool commands:
  - \bool\_gset\_false:N ..... 1275, 1294, 1317, 1339, 1355, 1456, 1695, 1731, 2299, 2345
  - \bool\_gset\_true:N ..... .. 1273, 1342, 1454, 1710, 2292, 2298
  - \bool\_if:NTF ..... 67, 699, 1285, 1289, 1305, 1308, 1312, 1323, 1330, 1334, 1346, 1350, 1467, 1472, 1477, 1669, 1714, 1827, 1862, 1972, 2014, 2287, 2302, 2307, 2312
  - \bool\_if:nTF ..... 2521, 2787, 3041
  - \bool\_lazy\_and:nnTF ..... .. 915, 989, 3131, 3158
  - \bool\_lazy\_or:nnTF ..... 1854, 2007
  - \bool\_new:N ..... .. 1276, 1343, 1457, 1711, 2272, 2273
  - \bool\_set\_false:N ..... .. 1837, 1939, 2032, 2096
- box commands:
  - \box\_dp:N ..... . 224, 226, 274, 276, 331, 333, 380, 382, 384, 386, 2324, 2357, 2358, 2383
  - \box\_ht:N ..... 226, 276, 333, 384, 386, 1874, 2069, 2329, 2368, 2369, 2385
  - \box\_if\_empty:NTF ..... 2418
  - \box\_move\_down:nn ..... 2246, 2324
  - \box\_move\_up:nn ..... 2248, 2329
  - \box\_new:N ..... 2131, 2236, 2237
  - \box\_set\_dp:Nn ..... 1794
  - \box\_set\_ht:Nn ..... 1793
  - \box\_set\_wd:Nn ..... 288, 1792
  - \box\_use:N ..... 231, 249, 263, 279, 306, 320, 336, 352, 364, 415, 429, 448, 1407, 1602, 1795, 2277
  - \box\_wd:N ..... 225, 233, 275, 281, 332, 338, 381, 383, 1873, 2068
- box internal commands:
  - \\_\_box\_backend\_clip:N ..... .. 213, 268, 325, 369
  - \l\_\_box\_backend\_cos\_fp ..... 283
  - \\_\_box\_backend\_rotate:Nn ..... .. 235, 283, 340, 419
  - \\_\_box\_backend\_rotate\_aux:Nn ..... .. 235, 283, 340
  - \\_\_box\_backend\_scale:Nnn ..... .. 252, 311, 355, 432
  - \l\_\_box\_backend\_sin\_fp ..... 283
- C**
- char commands:
  - \char\_set\_catcode\_space:n ..... 154
- clist commands:
  - \clist\_map\_function:nN ..... .. 1363, 1487, 1738
- color internal commands:
  - \\_\_color\_backend:nnn ..... 1161
  - \\_\_color\_backend\_cmyk:w ..... 1162
  - \g\_\_color\_backend\_colorant\_prop ..... .. 665, 684, 687, 707, 932
  - \\_\_color\_backend\_devicen\_-colorants:n ..... 666, 868, 987
  - \\_\_color\_backend\_devicen\_-colorants:w ..... 666
  - \\_\_color\_backend\_devicen\_-init:nnn ..... 855, 957
  - \\_\_color\_backend\_devicen\_init:w ..... 957
  - \\_\_color\_backend\_fill:n ..... .. 1068, 1095, 1125, 1143, 1150
  - \\_\_color\_backend\_fill\_cmyk:n ..... .. 1068, 1102, 1125, 1150
  - \\_\_color\_backend\_fill\_devicen:nn ..... 1094, 1116, 1142, 1212
  - \\_\_color\_backend\_fill\_gray:n ..... .. 1068, 1102, 1125, 1150
  - \\_\_color\_backend\_fill\_rgb:n ..... .. 1068, 1102, 1125, 1150
  - \\_\_color\_backend\_fill\_separation:nn ..... 1094, 1102, 1142, 1212
  - \l\_\_color\_backend\_fill\_tl ..... .. 636, 646, 1076, 1091
  - \\_\_color\_backend\_iccbase\_-device:nnn ..... 1034
  - \\_\_color\_backend\_iccbase\_-init:nnn ..... 874, 1008
  - \c\_\_color\_backend\_main\_stack\_int ..... 516
  - \\_\_color\_backend\_pickup:N ..... 456, 479
  - \\_\_color\_backend\_pickup:w ..... 14, 456, 479

\_color_backend_reset: . . . . .	619, 638, 654, 1079, 1092, 1102, 1134, 1159
\_color_backend_rgb:w . . . . .	1185
\_color_backend_select:n . .	619, 694
\_color_backend_select:nn .	638, 900
\_color_backend_select_cmyk:n . .	619, 638, 654
\_color_backend_select_devicen:nn	693, 877, 899, 1060
\_color_backend_select_gray:n . .	619, 638, 654
\_color_backend_select_iccbased:nn	696, 881, 899
\_color_backend_select_rgb:n . . .	619, 638, 654
\_color_backend_select_separation:nn	693, 877, 899, 1060
\_color_backend_separation_-init:n . . . . .	697
\_color_backend_separation_-init:nn . . . . .	903
\_color_backend_separation_-init:nnn . . . . .	697
\_color_backend_separation_-init:nnnn . . . . .	697
\_color_backend_separation_-init:nnnnn . . . . .	697, 879, 903
\_color_backend_separation_-init:nw . . . . .	697
\_color_backend_separation_-init:w . . . . .	697
\_color_backend_separation_-init_/DeviceCMYK:nnn . . . . .	697
\_color_backend_separation_-init_/DeviceGray:nnn . . . . .	697
\_color_backend_separation_-init_/DeviceRGB:nnn . . . . .	697
\_color_backend_separation_-init_aux:nnnnn . . . . .	697
\_color_backend_separation_-init_CIELAB:nnn . . . .	697, 879, 903
\_color_backend_separation_-init_CIELAB:nnnnn . . . . .	880
\_color_backend_separation_-init_count:n . . . . .	697
\_color_backend_separation_-init_count:w . . . . .	697
\_color_backend_separation_-init_Device:Nn . . . . .	697
\g_color_backend_stack_int . . .	516
\l_color_backend_stack_int . . . .	513, 541, 547, 648, 651, 1077, 1090
\_color_backend_stroke:n . . . . .	1068, 1097, 1102
\_color_backend_stroke_cmyk:n . .	1068, 1125, 1161
\_color_backend_stroke_cmyk:w .	1161
\_color_backend_stroke_devicen:nn	1094, 1120, 1142, 1212
\_color_backend_stroke_gray:n . .	1068, 1125, 1161
\_color_backend_stroke_gray_-aux:n . . . . .	1161
\_color_backend_stroke_rgb:n . . .	1068, 1125, 1161
\_color_backend_stroke_rgb:w .	1161
\_color_backend_stroke_separation:nn	1094, 1102, 1142, 1212
\l_color_backend_stroke_tl . . . .	636, 647, 1078, 1089
\g_color_model_int . . . . .	704, 713, 861, 889, 923, 997, 1029
\c_color_model_range_CIELAB_tl .	816, 851, 946, 953
color.sc . . . . .	619, 3241
cs commands:	
\cs_generate_variant:Nn . .	49, 63, 66, 99, 138, 143, 170, 201, 207, 569, 606, 718, 1222, 1417, 1611, 1986, 2043, 2059, 2135, 2172, 2231, 2723, 2751, 2861, 2883, 2918, 3128, 3196
\cs_gset:Npx . .	2533, 2537, 3046, 3051
\cs_gset_eq:NN . . . . .	658, 659, 1063, 1109, 1110, 1116, 1118, 1120
\cs_gset_protected:Npn . . . . .	551, 656, 660, 1062, 1104, 1111, 1113, 1115, 3162, 3201, 3210
\cs_if_exist:NTF . . . . .	27, 50, 457, 480, 539, 1024, 1047, 2414, 2812, 2838
\cs_if_exist_p:N .	916, 990, 3132, 3159
\cs_if_exist_use:NTF . . . . .	38, 731
\cs_new:Npn . . . .	681, 740, 742, 744, 746, 753, 759, 761, 767, 784, 791, 793, 1002, 1368, 1492, 1742, 2072, 2081, 2125, 2150, 2232, 2234, 2267, 2439, 2539, 2540, 2693, 2724, 2725, 2843, 2876, 2919, 2921, 2937, 3054, 3055, 3065, 3070, 3071, 3076, 3077
\cs_new:Npx . . . . .	666, 2560, 2595, 2752, 2763, 2830, 2967
\cs_new_eq:NN . . . . .	46, 57, 59, 695, 878, 901, 902, 1098, 1099, 1146, 1147, 1214, 1215, 1221, 1416, 1422, 1423, 1610, 1612, 1613, 1619, 1804, 1833, 1884, 1885, 1927, 1935, 1957, 2028, 2085, 2092, 2124, 2277
\cs_new_protected:Npn . . . . .	47, 54, 61, 64, 72, 78, 83, 85, 89, 100,



```

\__draw_backend_cm_decompose_-
  auxiii:nnnnN ..... 1548
\__draw_backend_curveto:nnnnnn ..
  ..... 1237, 1424, 1620
\__draw_backend_dash:n .....
  ..... 1357, 1481, 1733
\__draw_backend_dash_aux:nn .. 1733
\__draw_backend_dash_pattern:nn .
  ..... 1357, 1481, 1733
\__draw_backend_discardpath: ...
  ..... 1277, 1458, 1665
\__draw_backend_end: 1223, 1418, 1614
\__draw_backend_evenodd_rule: ...
  ..... 1272, 1453, 1661
\__draw_backend_fill: 1277, 1458, 1665
\__draw_backend_fillstroke: ....
  ..... 1277, 1458, 1665
\__draw_backend_join_bevel: ....
  ..... 1357, 1481, 1733
\__draw_backend_join_miter: ....
  ..... 1357, 1481, 1733
\__draw_backend_join_round: ....
  ..... 1357, 1481, 1733
\__draw_backend_lineto:nn .....
  ..... 1237, 1424, 1620
\__draw_backend_linewidth:n ....
  ..... 1357, 1481, 1733
\__draw_backend_literal:n .....
  ..... 1221, 1226, 1230, 1234,
  1236, 1239, 1247, 1255, 1264, 1278,
  1281, 1282, 1283, 1284, 1287, 1293,
  1303, 1310, 1316, 1321, 1326, 1327,
  1328, 1329, 1332, 1338, 1348, 1354,
  1359, 1372, 1376, 1378, 1380, 1382,
  1384, 1386, 1388, 1391, 1396, 1397,
  1398, 1399, 1400, 1401, 1405, 1406,
  1408, 1409, 1410, 1411, 1412, 1416,
  1426, 1431, 1436, 1446, 1459, 1461,
  1463, 1466, 1471, 1476, 1480, 1483,
  1496, 1500, 1502, 1504, 1506, 1508,
  1510, 1512, 1610, 1672, 1691, 1717
\__draw_backend_miterlimit:n ...
  ..... 1357, 1481, 1733
\__draw_backend_moveto:nn .....
  ..... 1237, 1424, 1620
\__draw_backend_nonzero_rule: ...
  ..... 1272, 1453, 1661
\__draw_backend_path:n ..... 1665
\g__draw_backend_path_int 1680, 1697
\g__draw_backend_path_tl .....
  ..... 1620, 1676, 1692, 1694, 1721
\__draw_backend_rectangle:nnnn ..
  ..... 1237, 1424, 1620
\__draw_backend_scope_begin: ...
  ..... 1233, 1419, 1422, 1612
\__draw_backend_scope_end: .....
  ..... 1233, 1421, 1422, 1612
\__draw_backend_stroke: .....
  ..... 1277, 1458, 1665
\g__draw_draw_clip_bool .. 1277, 1665
\g__draw_draw_eor_bool .....
  ... 1272, 1289, 1305, 1312, 1323,
  1334, 1350, 1453, 1467, 1472, 1477
\g__draw_draw_path_int ..... 1665
\g__draw_path_tl ..... 1730

E
\errmessage ..... 38
\evensidemargin ..... 2322
exp commands:
\exp_after:wN ..... 159, 465, 2079
\exp_args:Ne ..... 755, 2476, 3001
\exp_args:Nf ..... 1362, 1486, 2241
\exp_args:NNf ..... 236, 284, 341
\exp_args:Nnx ..... 2228, 2914
\exp_args:NV ..... 461
\exp_args:Nx . 701, 913, 1897, 1918,
  2185, 2200, 2318, 2880, 3087, 3144
\exp_last_unbraced:Nx ..... 470, 484
\exp_not:N ..... 522, 523, 531,
  533, 668, 674, 675, 676, 703, 704,
  707, 708, 713, 2562, 2564, 2567,
  2597, 2599, 2602, 2754, 2756, 2759,
  2765, 2767, 2770, 2807, 2808, 2814,
  2815, 2834, 2839, 2948, 2956, 2972
\exp_not:n .....
  .... 48, 97, 108, 136, 1016, 2176,
  2181, 2470, 2709, 2710, 2724, 2725,
  2737, 2738, 2892, 2897, 2908, 2981
\ExplBackendFileDate ..... 1

F
file commands:
\file_compare_timestamp:nNnTF . 1906
\file_parse_full_name:nNNN 1893, 1916
\fmtversion ..... 52
fp commands:
\fp_compare:nNnTF .....
  . 243, 290, 296, 348, 1529, 1542, 1587
\fp_eval:n . 236, 245, 258, 259, 284,
  301, 316, 318, 341, 350, 361, 362,
  426, 441, 442, 1169, 1170, 1171,
  1179, 1192, 1193, 1194, 1531, 1536,
  1537, 1544, 1554, 1555, 1556, 1557,
  1566, 1567, 1568, 1569, 1578, 1579,
  1580, 1581, 2467, 2636, 2995, 3088,
  3098, 3105, 3145, 3169, 3176, 3237

```

\fp_new:N	309, 310	\__graphics_backend_getbb_-auxvi:nNnn	2063, 2065
\fp_set:Nn	289, 292	\__graphics_backend_getbb_eps:n	1804, 1886, 1927, 2085
\fp_use:N	295, 299, 304	\__graphics_backend_getbb_eps:nm	1886
\fp_zero:N	291	\__graphics_backend_getbb_eps:nn	1897, 1904
\c_zero_fp	243, 290, 296, 348, 1529, 1542	\__graphics_backend_getbb_jpg:n	1819, 1927, 2022, 2086
<b>G</b>			
graphics commands:			
\graphics_bb_restore:nTF	1848, 2062	\__graphics_backend_getbb_-pagebox:w	2022, 2079
\graphics_bb_save:n	1877, 2070	\__graphics_backend_getbb_pdf:n	1819, 1912, 1927, 2022, 2093
\l_graphics_decodearray_tl	1825, 1826, 1836, 1856, 1860, 1861, 1938, 1970, 1971, 2009, 2012, 2013, 2031, 2095	\__graphics_backend_getbb_png:n	1819, 1927, 2022, 2086
\graphics_extract_bb:n	1933, 1940, 2090, 2097	\__graphics_backend_include:nn	2099
\l_graphics_interpolate_bool	1827, 1837, 1855, 1862, 1939, 1972, 2008, 2014, 2032, 2096	\__graphics_backend_include_-auxi:nn	1944
\l_graphics_llx_dim	1810, 1949, 2001, 2108	\__graphics_backend_include_-auxii:nnn	1944
\l_graphics_lly_dim	1811, 1950, 2002, 2109	\__graphics_backend_include_-auxiii:nnn	1944
\l_graphics_name_tl	1911	\__graphics_backend_include_-bitmap_quote:w	2073, 2114
\l_graphics_page_int	1821, 1841, 1842, 1866, 1867, 1931, 1968, 1969, 1995, 1996, 2024, 2037, 2038, 2077, 2078, 2088	\__graphics_backend_include_-eps:n	1805, 1886, 1944, 2099
\l_graphics_pagebox_tl	54, 1822, 1840, 1868, 1869, 1932, 1966, 1967, 1997, 1999, 2025, 2046, 2047, 2079, 2089	\__graphics_backend_include_-jpg:n	1879, 1944, 2114
\graphics_read_bb:n	1804, 1927, 2085	\__graphics_backend_include_-pdf:n	1879, 1918, 1944, 2073, 2099
\l_graphics_urx_dim	1812, 1873, 1951, 2003, 2068, 2110	\__graphics_backend_include_-pdf_-quote:w	2076, 2081
\l_graphics_ury_dim	1813, 1874, 1952, 2004, 2069, 2111, 2119, 2120	\__graphics_backend_include_-png:n	1879, 1944, 2114
graphics internal commands:			
\l__graphics_backend_dir_str	1886	\l_graphics_backend_name_str	1886
\l__graphics_backend_ext_str	1886	\l_graphics_graphics_attr_tl	1818, 1823, 1830, 1838, 1848, 1875, 1877, 1882
\__graphics_backend_getbb_auxi:n	1819	\l__graphics_internal_box	1871, 1873, 1874, 2067, 2068, 2069
\__graphics_backend_getbb_-auxi:nN	2022	\g_graphics_track_int	1943, 1989, 1990
\__graphics_backend_getbb_-auxii:n	1819	group commands:	
\__graphics_backend_getbb_-auxiii:nnN	2022	\group_begin:	151, 179, 198
\__graphics_backend_getbb_-auxiv:nnNnn	2022	\group_end:	164, 187
\__graphics_backend_getbb_-auxv:nNnn	2022	\group_insert_after:N	1079, 1092, 1107, 1134, 1159, 3156, 3193
<b>H</b>			
hbox commands:			
\hbox:n	2247, 2250, 2325, 2331, 2484, 2491, 3009, 3020		

`\hbox_overlap_right:n` ..... 231,  
 263, 279, 320, 336, 364, 448, 1407, 1602  
`\hbox_set:Nn` .. 1871, 2067, 2317, 2349  
`\hbox_set:Nw` ..... 2300  
`\hbox_set_end:` ..... 2315  
`\hbox_unpack:N` ..... 2436  
 hook commands:  
`\hook_gput_code:nnn` ..... 55

## I

int commands:  
`\int_compare:nNnTF` ..... 516,  
 558, 654, 1060, 1102, 1841, 1866,  
 1968, 1995, 2037, 2077, 2408, 2509,  
 2805, 2833, 2946, 2953, 2969, 3199  
`\int_const:Nn` ..... 157, 163, 523,  
 549, 584, 1875, 1990, 2145, 2683, 2871  
`\int_eval:n` .....  
 . 565, 575, 604, 615, 751, 760, 773,  
 775, 779, 792, 2533, 2537, 2783,  
 2808, 2815, 2828, 3038, 3046, 3051  
`\int_gincr:N` ..... 205, 371,  
 522, 1671, 1716, 1989, 2144, 2213,  
 2257, 2334, 2870, 2913, 2926, 2948  
`\int_gset:Nn` ..... 180, 199, 2397  
`\int_gset_eq:NN` 188, 2258, 2335, 2927  
`\int_if_exist:NTF` ..... 1979  
`\int_if_odd:nTF` ..... 2320  
`\int_new:N` ..... 171, 172,  
 418, 513, 519, 1697, 1943, 2140,  
 2238, 2269, 2271, 2866, 2923, 2939  
`\int_set:Nn` ..... 541  
`\int_set_eq:NN` ... 176, 195, 547, 2409  
`\int_step_function:nnnN` ..... 777  
`\int_use:N` ..... 373, 404, 531, 542,  
 704, 713, 861, 889, 923, 997, 1029,  
 1674, 1680, 1687, 1719, 1727, 1842,  
 1867, 1882, 1969, 1982, 1994, 1996,  
 2078, 2151, 2216, 2229, 2233, 2261,  
 2268, 2339, 2440, 2694, 2704, 2877,  
 2915, 2920, 2930, 2938, 2956, 2972  
`\int_value:w` .....  
 ..... 2562, 2597, 2754, 2765, 2783  
`\int_zero:N` ... 1821, 1931, 2024, 2088

## K

kernel internal commands:  
`\__kernel_backend_align_begin:` ..  
 ..... 72, 216, 240, 255  
`\__kernel_backend_align_end:` ...  
 ..... 72, 230, 248, 262  
`\__kernel_backend_first_shipout:n`  
 ..... 50, 69, 526, 701

`\g__kernel_backend_header_bool` ..  
 ..... 67, 699  
`\__kernel_backend_literal:n` ....  
 ..... 46, 62, 65, 70,  
 74, 81, 84, 86, 142, 145, 147, 149,  
 169, 345, 358, 528, 553, 554, 562,  
 572, 627, 633, 657, 661, 721, 857,  
 1106, 1112, 1114, 1133, 1158, 1225,  
 1231, 1526, 1533, 1539, 1599, 1604,  
 1807, 1946, 1981, 1991, 2105, 2116,  
 2860, 2976, 3038, 3042, 3047, 3052  
`\__kernel_backend_literal_page:n`  
 .... 100, 144, 2854, 2856, 3057, 3059  
`\__kernel_backend_literal_pdf:n` .  
 .. 89, 141, 271, 328, 1416, 3208, 3223  
`\__kernel_backend_literal_-  
 postscript:n` .....  
 .... 61, 75, 76, 80, 217, 218, 220,  
 221, 229, 241, 256, 1221, 2511, 2523  
`\__kernel_backend_literal_svg:n` .  
 ..... 168, 175, 186, 194, 204,  
 372, 374, 391, 883, 1610, 1785, 1796  
`\__kernel_backend_matrix:n` .....  
 ..... 128, 293, 314, 1516  
`\__kernel_backend_postscript:n` ..  
 ..... 64,  
 629, 1137, 1139, 1141, 1145, 2134,  
 2190, 2205, 2247, 2253, 2293, 2325,  
 2332, 2336, 2350, 2378, 2422, 2429,  
 2435, 2443, 2450, 2484, 2491, 3111  
`\__kernel_backend_scope:n` 173, 401,  
 406, 1199, 1617, 1662, 1664, 1684,  
 1724, 1746, 1758, 1760, 1762, 1764,  
 1766, 1768, 1770, 1772, 1775, 3237  
`\__kernel_backend_scope_begin:` ..  
 ..... 83, 110, 146, 173,  
 215, 239, 254, 270, 287, 313, 327,  
 344, 357, 1422, 1594, 1612, 1616, 1783  
`\__kernel_backend_scope_begin:n` .  
 ..... 173, 393, 421, 434  
`\__kernel_backend_scope_end:` 83,  
 110, 146, 173, 232, 250, 264, 280,  
 307, 321, 337, 353, 365, 416, 430,  
 449, 551, 1423, 1606, 1613, 1619, 1797  
`\g__kernel_backend_scope_int` ...  
 171, 178, 180, 185, 189, 197, 199, 205  
`\l__kernel_backend_scope_int` ...  
 ..... 171, 177, 190, 196  
`\g__kernel_clip_path_int` .....  
 369, 1671, 1674, 1687, 1716, 1719, 1727  
`\__kernel_color_backend_stack_-  
 init:Nnn` ..... 516, 582, 3135  
`\__kernel_color_backend_stack_-  
 pop:n` ..... 558, 596, 651, 3165

\_kernel_color_backend_stack_- push:nn ..... <a href="#">558</a> , <a href="#">596</a> , <a href="#">648</a> , <a href="#">1077</a> , <a href="#">1090</a> , <a href="#">3154</a> , <a href="#">3191</a>	\pdf_object_write:nn . <a href="#">940</a> , <a href="#">1013</a> , <a href="#">1039</a>
\_kernel_dependency_version_- check:Nn ..... <a href="#">1</a>	pdf internal commands:
\_kernel_dependency_version_- check:nn ..... <a href="#">27</a> , <a href="#">29</a>	\_pdf_backend:n ..... <a href="#">2859</a> , <a href="#">2863</a> , <a href="#">2865</a> , <a href="#">2891</a> , <a href="#">2896</a> , <a href="#">2905</a> , <a href="#">2928</a> , <a href="#">2950</a> , <a href="#">2966</a> , <a href="#">2979</a> , <a href="#">3011</a> , <a href="#">3012</a> , <a href="#">3022</a>
\_kernel_kern:n ..... ..... <a href="#">2252</a> , <a href="#">2254</a> , <a href="#">2483</a> , <a href="#">2487</a> , <a href="#">2490</a> , <a href="#">2494</a> , <a href="#">3008</a> , <a href="#">3016</a> , <a href="#">3019</a> , <a href="#">3035</a>	\_pdf_backend_annotation:nnnn .. ..... <a href="#">2239</a> , <a href="#">2547</a> , <a href="#">2924</a>
\c_kernel_sys_dvipdfmx_version_- int ..... <a href="#">151</a> , <a href="#">516</a> , <a href="#">558</a> , <a href="#">654</a> , <a href="#">1060</a> , <a href="#">1102</a> , <a href="#">2946</a> , <a href="#">2953</a> , <a href="#">2969</a> , <a href="#">3199</a>	\_pdf_backend_annotation_- aux:nnnn ..... <a href="#">2241</a> , <a href="#">2244</a>
	\g_pdf_backend_annotation_int .. .. <a href="#">2238</a> , <a href="#">2258</a> , <a href="#">2268</a> , <a href="#">2923</a> , <a href="#">2927</a> , <a href="#">2938</a>
	\_pdf_backend_annotation_last: . ..... <a href="#">2267</a> , <a href="#">2560</a> , <a href="#">2937</a>
	\_pdf_backend_bdc:nn ..... ..... <a href="#">2541</a> , <a href="#">2853</a> , <a href="#">3056</a> , <a href="#">3078</a>
	\_pdf_backend_catalog_gput:nn .. ..... <a href="#">2136</a> , <a href="#">2653</a> , <a href="#">2862</a> , <a href="#">3062</a>
	\_pdf_backend_compress_objects:n ..... <a href="#">2507</a> , <a href="#">2774</a> , <a href="#">3037</a> , <a href="#">3072</a>
	\_pdf_backend_compresslevel:n .. ..... <a href="#">2507</a> , <a href="#">2774</a> , <a href="#">3037</a> , <a href="#">3072</a>
	\l_pdf_backend_content_box <a href="#">2236</a> , <a href="#">2300</a> , <a href="#">2324</a> , <a href="#">2327</a> , <a href="#">2329</a> , <a href="#">2358</a> , <a href="#">2369</a>
	\_pdf_backend_destination:nn ... ..... <a href="#">2448</a> , <a href="#">2616</a> , <a href="#">2977</a>
	\_pdf_backend_destination:nnnn . ..... <a href="#">2448</a> , <a href="#">2616</a> , <a href="#">2977</a>
	\_pdf_backend_destination_- aux:nnnn ..... <a href="#">2448</a> , <a href="#">2977</a>
	\_pdf_backend_emc: ..... ..... <a href="#">2541</a> , <a href="#">2853</a> , <a href="#">3056</a> , <a href="#">3078</a>
	\_pdf_backend_info_gput:nn .... ..... <a href="#">2136</a> , <a href="#">2653</a> , <a href="#">2862</a> , <a href="#">3062</a>
	\_pdf_backend_link:nw ..... <a href="#">2278</a>
	\_pdf_backend_link_aux:nw ... <a href="#">2278</a>
	\_pdf_backend_link_begin:n .. <a href="#">2940</a>
	\_pdf_backend_link_begin:nnnw <a href="#">2571</a>
	\_pdf_backend_link_begin:nw ... ..... <a href="#">2280</a> , <a href="#">2284</a> , <a href="#">2285</a>
	\_pdf_backend_link_begin_aux:nw ..... <a href="#">2288</a> , <a href="#">2290</a>
	\_pdf_backend_link_begin_- goto:nnw ..... <a href="#">2278</a> , <a href="#">2571</a> , <a href="#">2940</a>
	\_pdf_backend_link_begin_- user:nnw ..... <a href="#">2278</a> , <a href="#">2571</a> , <a href="#">2940</a>
	\g_pdf_backend_link_bool ..... ..... <a href="#">2273</a> , <a href="#">2287</a> , <a href="#">2292</a> , <a href="#">2307</a> , <a href="#">2345</a>
	\g_pdf_backend_link_dict_tl ... ..... <a href="#">2270</a> , <a href="#">2295</a> , <a href="#">2340</a>
	\_pdf_backend_link_end: ..... ..... <a href="#">2278</a> , <a href="#">2571</a> , <a href="#">2940</a>
	\_pdf_backend_link_end_aux: . <a href="#">2278</a>
<b>M</b>	
\MessageBreak ..... <a href="#">40</a>	
mode commands:	
\mode_if_horizontal:TF ... <a href="#">2399</a> , <a href="#">2406</a>	
\mode_if_math:TF ..... <a href="#">2297</a>	
<b>O</b>	
\oddsidemargin ..... <a href="#">2321</a>	
opacity internal commands:	
\_opacity_backend:nn ..... <a href="#">3230</a>	
\_opacity_backend:nnn ..... <a href="#">3085</a>	
\_opacity_backend_fill:n ..... ..... <a href="#">3085</a> , <a href="#">3166</a> , <a href="#">3230</a>	
\_opacity_backend_fill_stroke:nn ..... <a href="#">3168</a> , <a href="#">3174</a> , <a href="#">3178</a> , <a href="#">3196</a> , <a href="#">3210</a>	
\l_opacity_backend_fill_tl .... .. <a href="#">3140</a> , <a href="#">3149</a> , <a href="#">3175</a> , <a href="#">3183</a> , <a href="#">3203</a> , <a href="#">3215</a>	
\_opacity_backend_fillstroke:nn ..... <a href="#">3166</a>	
\_opacity_backend_reset: <a href="#">3142</a> , <a href="#">3193</a>	
\_opacity_backend_select:n .... ..... <a href="#">3085</a> , <a href="#">3142</a> , <a href="#">3199</a> , <a href="#">3230</a>	
\_opacity_backend_select_aux:n . ..... <a href="#">3085</a> , <a href="#">3142</a> , <a href="#">3181</a> , <a href="#">3201</a> , <a href="#">3213</a>	
\c_opacity_backend_stack_int ... ..... <a href="#">3131</a> , <a href="#">3154</a> , <a href="#">3165</a> , <a href="#">3191</a>	
\_opacity_backend_stroke:n .... ..... <a href="#">3085</a> , <a href="#">3166</a> , <a href="#">3230</a>	
\l_opacity_backend_stroke_tl ... .. <a href="#">3140</a> , <a href="#">3150</a> , <a href="#">3170</a> , <a href="#">3184</a> , <a href="#">3204</a> , <a href="#">3216</a>	
<b>P</b>	
pdf commands:	
\pdf_object_if_exist:nTF ..... ..... <a href="#">937</a> , <a href="#">1010</a> , <a href="#">1036</a>	
\pdf_object_new:nn ... <a href="#">939</a> , <a href="#">1012</a> , <a href="#">1038</a>	
\pdf_object_ref:n .... <a href="#">952</a> , <a href="#">1023</a> , <a href="#">1046</a>	
\pdf_object_ref_last: ..... .. <a href="#">924</a> , <a href="#">931</a> , <a href="#">933</a> , <a href="#">986</a> , <a href="#">998</a> , <a href="#">1030</a> , <a href="#">1054</a>	
\pdf_object_unnamed_write:nn ... ..... <a href="#">905</a> , <a href="#">930</a> , <a href="#">959</a> , <a href="#">981</a> , <a href="#">1022</a> , <a href="#">1045</a>	

\g_pdf_backend_link_int .....	\__pdf_backend_pdfmark:n .....
..... 2269, 2335,	2133, 2137, 2139, 2154, 2175, 2180,
2339, 2440, 2939, 2948, 2956, 2972	2214, 2259, 2451, 2495, 2542, 2544
\__pdf_backend_link_last: .....	\__pdf_backend_version_major: ...
..... 2439, 2595, 2967	..... 2533,
\__pdf_backend_link_margin:n ...	2539, 2830, 3046, 3047, 3054, 3076
..... 2441, 2606, 2975	\__pdf_backend_version_major_-
\g_pdf_backend_link_math_bool ..	gset:n ..... 2531, 2802, 3044, 3074
..... 2272, 2298, 2299, 2302, 2312	\__pdf_backend_version_minor: ...
\__pdf_backend_link_minima: .. 2278	..... 2537,
\__pdf_backend_link_outerbox:n 2278	2539, 2830, 3051, 3052, 3054, 3076
\g_pdf_backend_link_sf_int ....	\__pdf_backend_version_minor_-
..... 2271, 2397, 2408, 2409	gset:n ..... 2531, 2802, 3044, 3074
\__pdf_backend_link_sf_restore: 2278	\l_pdf_breaklink_pdfmark_tl ...
\__pdf_backend_link_sf_save: . 2278	..... 2274, 2342, 2434
\l_pdf_backend_model_box . 2237,	\__pdf_breaklink_postscript:n ...
2317, 2349, 2357, 2368, 2383, 2385	..... 2276, 2326, 2328, 2435
\__pdf_backend_objcompresslevel:n	\__pdf_breaklink_usebox:N .....
..... 2774	..... 2277, 2327, 2436
\g_pdf_backend_object_int .....	\__pdf_exp_not_i:nn . 2695, 2741, 2746
..... 2140, 2144, 2147,	\__pdf_exp_not_ii:nn 2695, 2742, 2747
2213, 2216, 2229, 2233, 2257, 2258,	\l_pdf_internal_box .....
2261, 2334, 2335, 2866, 2870, 2873,	pdf.baselineskip .....
2913, 2915, 2920, 2926, 2927, 2930	2278, 3568
\__pdf_backend_object_last: ....	pdf.bordertracking .....
..... 2232, 2752, 2919, 3064	3326
\__pdf_backend_object_new:nn ...	pdf.bordertracking.begin .....
..... 2142, 2674, 2868, 3064	3326
\__pdf_backend_object_now:nn ...	pdf.bordertracking.continue .....
..... 2211, 2726, 2911, 3064	3326
\g_pdf_backend_object_prop ....	pdf.bordertracking.end .....
..... 2140, 2148, 2159, 2169,	3326
2673, 2691, 2707, 2866, 2874, 2881	pdf.bordertracking.endpage .....
\__pdf_backend_object_ref:n 2142,	3326
2156, 2170, 2674, 2868, 2887, 3064	pdf.breaklink .....
\__pdf_backend_object_write:nn ..	3464
..... 2152, 2695, 2878, 3064	pdf.breaklink.write .....
\__pdf_backend_object_write:nnn 2878	3464
\__pdf_backend_object_write_-	pdf.brokenlink.dict .....
array:nn ..... 2152, 2878	3326
\__pdf_backend_object_write_-	pdf.brokenlink.rect .....
dict:nn ..... 2152, 2878	3326
\__pdf_backend_object_write_-	pdf.brokenlink.skip .....
fstream:nn ..... 2152, 2878	3326
\__pdf_backend_object_write_-	pdf.count .....
fstream:nnn ..... 2186, 2188	3464
\__pdf_backend_object_write_-	pdf.currentrect .....
stream:nn ..... 2152, 2878	3464
\__pdf_backend_object_write_-	pdf.cvs .....
stream:nnn ..... 2152	3248
\__pdf_backend_object_write_-	pdf.dest.anchor .....
stream:nnnn ..... 2878	3291
\__pdf_backend_pageobject_ref:n .	pdf.dest.point .....
..... 2234, 2763, 2921, 3064	3291
	pdf.dest.x .....
	3291
	pdf.dest.y .....
	3291
	pdf.dest2device .....
	3291
	pdf.dev.x .....
	3291
	pdf.dev.y .....
	3291
	pdf.dvi.pt .....
	3248
	pdf.globaldict .....
	3245
	pdf.leftboundary .....
	3326
	pdf.link.dict .....
	2278
	pdf.linkdp.pad .....
	2278, 3252
	pdf.linkht.pad .....
	2278, 3252
	pdf.linkmargin .....
	3252
	pdf.llx .....
	2278, 3255
	pdf.lly .....
	2278, 3255
	pdf.originx .....
	3326
	pdf.originy .....
	3326

pdf.outerbox .....	<a href="#">2278</a> , <a href="#">3568</a>	\s_graphics_stop .....	<a href="#">2076</a> , <a href="#">2081</a> , <a href="#">2121</a> , <a href="#">2125</a>
pdf.pdfmark .....	<a href="#">3568</a>	separation .....	<a href="#">3242</a>
pdf.pdfmark.dict .....	<a href="#">3568</a>	skip commands:	
pdf.pdfmark.good .....	<a href="#">3568</a>	\skip_horizontal:n ....	<a href="#">233</a> , <a href="#">281</a> , <a href="#">338</a>
pdf.pt.dvi .....	<a href="#">3248</a>	str commands:	
pdf.rect .....	<a href="#">3255</a>	\c_hash_str ....	<a href="#">404</a> , <a href="#">1680</a> , <a href="#">1687</a> , <a href="#">1727</a>
pdf.rect.ht .....	<a href="#">3248</a>	\c_percent_str ....	<a href="#">1205</a> , <a href="#">1206</a> , <a href="#">1207</a>
pdf.rightboundary .....	<a href="#">3326</a>	\str_case:nn .....	<a href="#">971</a> , <a href="#">2218</a> , <a href="#">2734</a>
pdf.save.linkll .....	<a href="#">3255</a>	\str_case:nnTF .....	<a href="#">2455</a> , <a href="#">2625</a> , <a href="#">2984</a>
pdf.save.linkur .....	<a href="#">3255</a>	\str_case_e:nn .....	<a href="#">2158</a> , <a href="#">2706</a>
pdf.save.ll .....	<a href="#">3255</a>	\str_convert_pdfname:n .	<a href="#">708</a> , <a href="#">728</a> , <a href="#">914</a>
pdf.save.ur .....	<a href="#">3255</a>	\str_if_eq:nnTF .....	
pdf.tmpa .....	<a href="#">3291</a>	..	<a href="#">490</a> , <a href="#">493</a> , <a href="#">496</a> , <a href="#">499</a> , <a href="#">887</a> , <a href="#">3180</a> , <a href="#">3212</a>
pdf.tmpb .....	<a href="#">3291</a>	\str_new:N .....	<a href="#">1888</a> , <a href="#">1889</a> , <a href="#">1890</a>
pdf.tmpc .....	<a href="#">3291</a>	\str_tail:N .....	<a href="#">1899</a> , <a href="#">1920</a>
pdf.tmpd .....	<a href="#">3291</a>	sys commands:	
pdf.urx .....	<a href="#">3255</a>	\sys_get_shell:nnNTF .....	<a href="#">153</a>
pdf.ury .....	<a href="#">2278</a> , <a href="#">3255</a>	\sys_if_shell:TF .....	<a href="#">1886</a>
pdfmanagement commands:		\sys_shell_now:n .....	<a href="#">1908</a>
\pdfmanagement_add:nnn ....	<a href="#">921</a> , <a href="#">995</a> , <a href="#">1024</a> , <a href="#">1028</a> , <a href="#">1047</a> , <a href="#">1051</a> , <a href="#">3137</a> , <a href="#">3151</a> , <a href="#">3185</a> , <a href="#">3188</a> , <a href="#">3205</a> , <a href="#">3217</a> , <a href="#">3220</a>	sys internal commands:	
\pdfmanagement_if_active_p:	<a href="#">916</a> , <a href="#">917</a> , <a href="#">990</a> , <a href="#">991</a> , <a href="#">3132</a> , <a href="#">3133</a> , <a href="#">3159</a> , <a href="#">3160</a>	\l_sys_internal_tl .....	<a href="#">155</a> , <a href="#">159</a>
prg commands:		\__sys_tmp:w .....	<a href="#">152</a> , <a href="#">159</a>
\prg_replicate:nn .....	<a href="#">184</a> , <a href="#">749</a> , <a href="#">770</a> , <a href="#">780</a> , <a href="#">965</a>	<b>T</b>	
prop commands:		TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
\prop_gput:Nnn .....	<a href="#">707</a> , <a href="#">932</a> , <a href="#">2148</a> , <a href="#">2691</a> , <a href="#">2874</a>	\ccclv .....	<a href="#">2418</a> , <a href="#">2420</a> , <a href="#">2428</a>
\prop_if_in:NnTF .....	<a href="#">684</a>	\cifl@t@r .....	<a href="#">50</a> , <a href="#">52</a>
\prop_item:Nn .....	<a href="#">687</a> , <a href="#">2159</a> , <a href="#">2169</a> , <a href="#">2707</a> , <a href="#">2881</a>	\@makecol@hook .....	<a href="#">2412</a>
\prop_new:N ....	<a href="#">665</a> , <a href="#">2141</a> , <a href="#">2673</a> , <a href="#">2867</a>	\current@color .	<a href="#">14</a> , <a href="#">461</a> , <a href="#">465</a> , <a href="#">471</a> , <a href="#">485</a>
\ProvidesExplFile .....	<a href="#">2</a>	\special .....	<a href="#">2</a>
<b>Q</b>		tex commands:	
quark commands:		\tex_baselineskip:D .....	<a href="#">2389</a>
\quark_if_recursion_tail_stop:n	<a href="#">683</a>	\tex_endinput:D .....	<a href="#">44</a>
\q_recursion_stop .....	<a href="#">676</a>	\tex_global:D .....	<a href="#">2776</a> , <a href="#">2793</a> , <a href="#">2807</a> , <a href="#">2814</a> , <a href="#">2821</a>
\q_recursion_tail .....	<a href="#">675</a>	\tex_immediate:D .....	<a href="#">1853</a> , <a href="#">2698</a> , <a href="#">2701</a> , <a href="#">2729</a> , <a href="#">2732</a>
\q_stop .....	<a href="#">152</a> , <a href="#">160</a>	\tex_luatexversion:D ....	<a href="#">2805</a> , <a href="#">2833</a>
<b>S</b>		\tex_pdfannot:D .....	<a href="#">2553</a>
scan commands:		\tex_pdfcatalog:D .....	<a href="#">2659</a>
\scan_stop: .....	<a href="#">113</a> , <a href="#">122</a> , <a href="#">615</a> , <a href="#">2589</a> , <a href="#">2614</a> , <a href="#">2637</a> , <a href="#">2651</a> , <a href="#">2783</a> , <a href="#">2800</a> , <a href="#">2808</a> , <a href="#">2815</a> , <a href="#">2828</a>	\tex_pdfcolorstack:D .....	<a href="#">602</a> , <a href="#">613</a>
scan internal commands:		\tex_pdfcolorstackinit:D .....	<a href="#">590</a>
\s_color_stop .....	<a href="#">471</a> , <a href="#">474</a> , <a href="#">485</a> , <a href="#">488</a> , <a href="#">760</a> , <a href="#">761</a> , <a href="#">765</a> , <a href="#">769</a> , <a href="#">782</a> , <a href="#">785</a> , <a href="#">789</a> , <a href="#">793</a> , <a href="#">807</a> , <a href="#">966</a> , <a href="#">1002</a> , <a href="#">1006</a> , <a href="#">1162</a> , <a href="#">1164</a> , <a href="#">1185</a> , <a href="#">1187</a>	\tex_pdfcompresslevel:D .....	<a href="#">2781</a>
		\tex_pdfdest:D .....	<a href="#">2622</a> , <a href="#">2645</a>
		\tex_pdfendlink:D .....	<a href="#">2592</a>
		\tex_pdfextension:D .....	<a href="#">92</a> , <a href="#">103</a> , <a href="#">113</a> , <a href="#">122</a> , <a href="#">131</a> , <a href="#">599</a> , <a href="#">610</a> , <a href="#">2550</a> , <a href="#">2578</a> , <a href="#">2589</a> , <a href="#">2619</a> , <a href="#">2642</a> , <a href="#">2656</a> , <a href="#">2666</a> , <a href="#">2677</a> , <a href="#">2698</a> , <a href="#">2729</a>
		\tex_pdffeedback:D .....	<a href="#">587</a> , <a href="#">2564</a> , <a href="#">2599</a> , <a href="#">2686</a> , <a href="#">2756</a> , <a href="#">2767</a>
		\tex_pdfinfo:D .....	<a href="#">2669</a>
		\tex_pdflastannot:D .....	<a href="#">2567</a>

