

File I

Implementation

1 l3backend-basics Implementation

```
1 <*package>
```

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

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

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

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

```
27 \cs_if_exist:NTF \_kernel_dependency_version_check:nn
28 {
29   \_kernel_dependency_version_check:nn {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_YTeX share drawing routines.
- X_YTeX 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.

```

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.

```

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
\__kernel_backend_scope:n
\__kernel_backend_scope: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_EX 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_YTeX backend

324 <*dvipdfmx | xetex>

__box_backend_clip:N The code here is identical to that for Lua_{TeX}/pdf_{TeX}: 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_YTeX 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 \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 \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 \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^AT_EX 2_ε, 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_YL_AT_EX in particular. Whilst it is in some ways convenient to use the same approach in multiple backends, the fact that dvipdfmx/X_YL_AT_EX is PDF-based means it (largely) sticks closer to direct PDF output.

3.1 Collecting information from L^AT_EX 2_ε

3.1.1 dvips-style

```

455 <*dvisvgm | dvipdfmx | dvips | xetex>

```

Allow for L^AT_EX 2_ε 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_T_EX and pdf_T_EX

```

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_YTeX 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_YTeX

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>

```

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

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc
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     \group_insert_after:N \__color_backend_reset:
632 }
633 \cs_new_protected:Npn \__color_backend_reset:
634 { \__kernel_backend_literal:n { color~pop } }

```

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

```

635 </dvips | dvisvgm>

```

3.3.2 LuaTeX and pdfTeX

636 $\langle *dvipdfmx \mid \text{luatex} \mid \text{pdfTeX} \mid \text{xetex} \rangle$

$\backslash l_color_backend_fill_tl$
 $\backslash l_color_backend_stroke_tl$

637 $\backslash tl_new:N \backslash l_color_backend_fill_tl$
 638 $\backslash tl_new:N \backslash l_color_backend_stroke_tl$

(End definition for $\backslash l_color_backend_fill_tl$ and $\backslash l_color_backend_stroke_tl$.)

$\backslash_color_backend_select_cmyk:n$
 $\backslash_color_backend_select_gray:n$
 $\backslash_color_backend_select_rgb:n$
 $\backslash_color_backend_select:nn$
 $\backslash_color_backend_reset:$

Store the values then pass to the stack.

639 $\backslash cs_new_protected:Npn \backslash_color_backend_select_cmyk:n \#1$
 640 $\{ \backslash_color_backend_select:nn \{ \#1 \sim k \} \{ \#1 \sim K \} \}$
 641 $\backslash cs_new_protected:Npn \backslash_color_backend_select_gray:n \#1$
 642 $\{ \backslash_color_backend_select:nn \{ \#1 \sim g \} \{ \#1 \sim G \} \}$
 643 $\backslash cs_new_protected:Npn \backslash_color_backend_select_rgb:n \#1$
 644 $\{ \backslash_color_backend_select:nn \{ \#1 \sim rg \} \{ \#1 \sim RG \} \}$
 645 $\backslash cs_new_protected:Npn \backslash_color_backend_select:nn \#1\#2$
 646 $\{$
 647 $\backslash tl_set:Nn \backslash l_color_backend_fill_tl \{ \#1 \}$
 648 $\backslash tl_set:Nn \backslash l_color_backend_stroke_tl \{ \#2 \}$
 649 $\backslash_kernel_color_backend_stack_push:nn \backslash l_color_backend_stack_int \{ \#1 \sim \#2 \}$
 650 $\backslash group_insert_after:N \backslash_color_backend_reset:$
 651 $\}$
 652 $\backslash cs_new_protected:Npn \backslash_color_backend_reset:$
 653 $\{ \backslash_kernel_color_backend_stack_pop:n \backslash l_color_backend_stack_int \}$

(End definition for $\backslash_color_backend_select_cmyk:n$ and others.)

654 $\langle /dvipdfmx \mid \text{luatex} \mid \text{pdfTeX} \mid \text{xetex} \rangle$

3.3.3 dvipdfmx/X_YTeX

655 $\langle *dvipdfmx \mid \text{xetex} \rangle$

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).

Push the data to the stack.

$\backslash_color_backend_select_cmyk:n$
 $\backslash_color_backend_select_gray:n$
 $\backslash_color_backend_select_rgb:n$
 $\backslash_color_backend_reset:$

656 $\backslash int_compare:nNnT \backslash c_kernel_sys_dvipdfmx_version_int < \{ 20201111 \}$
 657 $\{$
 658 $\backslash cs_gset_protected:Npn \backslash_color_backend_select_cmyk:n \#1$
 659 $\{$
 660 $\backslash_kernel_backend_literal:n \{ pdf: bc \sim [\#1] \}$
 661 $\backslash group_insert_after:N \backslash_color_backend_reset:$
 662 $\}$
 663 $\backslash cs_gset_eq:NN \backslash_color_backend_select_gray:n \backslash_color_backend_select_cmyk:n$
 664 $\backslash cs_gset_eq:NN \backslash_color_backend_select_rgb:n \backslash_color_backend_select_cmyk:n$
 665 $\backslash cs_gset_protected:Npn \backslash_color_backend_reset:$
 666 $\{ \backslash_kernel_backend_literal:n \{ pdf: ec \} \}$
 667 $\}$

(End definition for $\backslash_color_backend_select_cmyk:n$ and others.)

668 $\langle /dvipdfmx \mid \text{xetex} \rangle$

3.4 Separations

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

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

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

```
\g_color_backend_colorant_prop
```

```
670 \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
```

```
671 \cs_new:Npx \_color_backend_devicen_colorants:n #1
```

```
672 {
```

```
673   \exp_not:N \tl_if_blank:nF {#1}
```

```
674   {
```

```
675     \c_space_tl
```

```
676     << ~
```

```
677     /Colorants ~
```

```
678     << ~
```

```
679     \exp_not:N \_color_backend_devicen_colorants:w #1 ~
```

```
680     \exp_not:N \q_recursion_tail \c_space_tl
```

```
681     \exp_not:N \q_recursion_stop
```

```
682     >> ~
```

```
683   >>
```

```
684   }
```

```
685 }
```

```
686 \cs_new:Npn \_color_backend_devicen_colorants:w #1 ~
```

```
687 {
```

```
688   \quark_if_recursion_tail_stop:n {#1}
```

```
689   \prop_if_in:NnT \g__color_backend_colorant_prop {#1}
```

```
690   {
```

```
691     #1 ~
```

```
692     \prop_item:Nn \g__color_backend_colorant_prop {#1} ~
```

```
693   }
```

```
694   \_color_backend_devicen_colorants:w
```

```
695 }
```

(End definition for _color_backend_devicen_colorants:n and _color_backend_devicen_colorants:w.)

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

```
697 <*dvips>
```

```
\_color_backend_select_separation:nn
```

```
\_color_backend_select_devicen:nn
```

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

```
699 { \_color_backend_select:n { separation ~ #1 ~ #2 } }
```

```
700 \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.

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

(End definition for `_color_backend_select_iccbased:nn`.)

```
\_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
```

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.

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

```

750 { \_color_backend_separation_init_Device:Nn 2 {#3} }
751 \cs_new:Npn \_color_backend_separation_init_Device:Nn #1#2
752 {
753   #2 ~
754   \prg_replicate:nn {#1}
755   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
756   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
757 }

```

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.

```

758 \cs_new:Npn \_color_backend_separation_init:nnn #1#2#3
759 {
760   \exp_args:Ne \_color_backend_separation_init:nnnn
761   { \_color_backend_separation_init_count:n {#2} }
762   {#1} {#2} {#3}
763 }
764 \cs_new:Npn \_color_backend_separation_init_count:n #1
765 { \int_eval:n { 0 \_color_backend_separation_init_count:w #1 ~ \s_color_stop } }
766 \cs_new:Npn \_color_backend_separation_init_count:w #1 ~ #2 \s_color_stop
767 {
768   +1
769   \tl_if_blank:nF {#2}
770   { \_color_backend_separation_init_count:w #2 \s_color_stop }
771 }

```

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.

```

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

```

```

787     { \_color_backend_separation_init:nw {#1} #2 ~ \s_color_stop }
788   }
789   \cs_new:Npn \_color_backend_separation_init:w
790     #1 ~ #2 \s_color_stop #3 ~ #4 \s_color_stop
791   {
792     #1 ~ #3 ~ 0 ~
793     \tl_if_blank:nF {#2}
794     { \_color_backend_separation_init:w #2 \s_color_stop #4 \s_color_stop }
795   }
796   \cs_new:Npn \_color_backend_separation_init:n #1
797   { \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.

```

798   \cs_new:Npn \_color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s_color_stop
799   {
800     #2 ~ #3 ~
801     2 ~ index ~ 2 ~ index ~ lt ~
802     { ~ pop ~ excl ~ pop ~ } ~
803     { ~
804       2 ~ index ~ 1 ~ index ~ gt ~
805       { ~ excl ~ pop ~ excl ~ pop ~ } ~
806       { ~ pop ~ pop ~ } ~
807     } ~
808   }
809   \ifelse ~
810   #1 ~ 1 ~ roll ~
811   \tl_if_blank:nF {#4}
812   { \_color_backend_separation_init:nw {#1} #4 \s_color_stop }
813 }

```

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.

```

814   \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
815   {
816     \_color_backend_separation_init:nxxxnn
817     {#2}
818     {
819       /CIEBasedABC ~
820       << ~
821       /RangeABC ~ [ ~ \c_color_model_range_CIELAB_tl \c_space_tl ] ~
822       /DecodeABC ~
823       [ ~
824         { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
825         { ~ 500 ~ div ~ } ~ bind ~
826         { ~ 200 ~ div ~ } ~ bind ~
827       ] ~
828       /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
829       /DecodeLMN ~
830       [ ~
831         { ~
832           dup ~ 6 ~ 29 ~ div ~ ge ~
833           { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
834           { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~

```

```

835         ifelse ~
836         0.9505 ~ mul ~
837     } ~ bind ~
838     { ~
839         dup ~ 6 ~ 29 ~ div ~ ge ~
840         { ~ dup ~ dup ~ mul ~ mul ~ } ~
841         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
842         ifelse ~
843     } ~ bind ~
844     { ~
845         dup ~ 6 ~ 29 ~ div ~ ge ~
846         { ~ dup ~ dup ~ mul ~ mul ~ } ~
847         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
848         ifelse ~
849         1.0890 ~ mul ~
850     } ~ bind
851 ] ~
852 /WhitePoint ~
853 [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
854 >>
855 }
856 { \c__color_model_range_CIELAB_tl }
857 { 100 ~ 0 ~ 0 }
858 {#3}
859 }

```

(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.

```

860 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
861 {
862     \_kernel_backend_literal:e
863     {
864         !
865         TeXDict ~ begin ~
866         /color \int_use:N \g__color_model_int
867         {
868             [ ~
869             /DeviceN ~
870             [ ~ #1 ~ ] ~
871             #2 ~
872             { ~ #3 ~ } ~
873             \_color_backend_devicen_colorants:n {#1}
874             ] ~ setcolorspace
875         } ~ def ~
876     end
877 }
878 }

```

(End definition for `_color_backend_devicen_init:nnn`.)

`_color_backend_iccbased_init:nmn` No support at present.

```

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

```


(End definition for `_color_backend_iccbased_init:nnn`.)

880 `</dvips>`

881 `<*dvisvgm>`

`_color_backend_select_separation:nn` No support at present.

`_color_backend_select_devicen:nn`

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

883 `\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`

884 `\cs_new_protected:Npn _color_backend_separation_init:nnnnn #1#2#3#4#5 { }`

885 `\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.

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

887 `{`

888 `_kernel_backend_literal_svg:x`

889 `{`

890 `<style>`

891 `@color-profile ~`

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

893 `{ device-cmyk }`

894 `{ --color \int_use:N \g__color_model_int }`

895 `\c_space_tl`

896 `{`

897 `src:("#1")`

898 `}`

899 `</style>`

900 `}`

901 `}`

(End definition for `_color_backend_select_iccbased:nn`.)

902 `</dvisvgm>`

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

`_color_backend_select_separation:nn`

`_color_backend_select_devicen:nn`

`_color_backend_select_iccbased: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.

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

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

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

907 `\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.

```

908 \cs_new_protected:Npn __color_backend_separation_init:nnnn #1#2#3#4#5
909 {
910   \pdf_object_unnamed_write:nx { dict }
911   {
912     /FunctionType ~ 2
913     /Domain ~ [0 ~ 1]
914     \tl_if_blank:nF {#3} { /Range ~ [#3] }
915     /C0 ~ [#4] ~
916     /C1 ~ [#5] /N ~ 1
917   }
918   \exp_args:Nx __color_backend_separation_init:nn
919   { \str_convert_pdfname:n {#1} } {#2}
920   \bool_lazy_and:nnT
921   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
922   { \pdfmanagement_if_active_p: }
923   {
924     \use:x
925     {
926       \pdfmanagement_add:nnn
927       { Page / Resources / ColorSpace }
928       { color \int_use:N \g__color_model_int }
929       { \pdf_object_ref_last: }
930     }
931   }
932 }
933 \cs_new_protected:Npn __color_backend_separation_init:nn #1#2
934 {
935   \pdf_object_unnamed_write:nx { array }
936   { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
937   \prop_gput:Nnx \g__color_backend_colorant_prop { /#1 }
938   { \pdf_object_ref_last: }
939 }

```

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

```

940 \cs_new_protected:Npn __color_backend_separation_init_CIELAB:nnn #1#2#3
941 {
942   \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
943   {
944     \pdf_object_new:nn { __color_illuminant_CIELAB_ #1 } { array }
945     \pdf_object_write:nx { __color_illuminant_CIELAB_ #1 }
946     {
947       /Lab ~
948       <<
949       /WhitePoint ~
950       [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
951       /Range ~ [ \c__color_model_range_CIELAB_tl ]
952       >>
953     }
954   }
955   __color_backend_separation_init:nnnn

```

```

956     {#2}
957     { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
958     { \c__color_model_range_CIELAB_t1 }
959     { 100 ~ 0 ~ 0 }
960     {#3}
961 }

```

(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.

```

962 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
963 {
964   \pdf_object_unnamed_write:nx { stream }
965   {
966     {
967       /FunctionType ~ 4 ~
968       /Domain ~
969       [ ~
970         \prg_replicate:nn
971         { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
972         { 0 ~ 1 ~ }
973       ] ~
974       /Range ~
975       [ ~
976         \str_case:nn {#2}
977         {
978           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
979           { /DeviceGray } { 0 ~ 1 }
980           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
981         } ~
982       ]
983     }
984     { {#3} }
985   }
986   \pdf_object_unnamed_write:nx { array }
987   {
988     /DeviceN ~
989     [ ~ #1 ~ ] ~
990     #2 ~
991     \pdf_object_ref_last:
992     \__color_backend_devicen_colorants:n {#1}
993   }
994   \bool_lazy_and:nnT
995   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
996   { \pdfmanagement_if_active_p: }
997   {
998     \use:x
999     {
1000       \pdfmanagement_add:nnn
1001       { Page / Resources / ColorSpace }
1002       { color \int_use:N \g__color_model_int }
1003       { \pdf_object_ref_last: }

```

```

1004     }
1005   }
1006 }
1007 \cs_new:Npn \__color_backend_devicen_init:w #1 ~ #2 \s__color_stop
1008 {
1009   + 1
1010   \tl_if_blank:nF {#2}
1011   { \__color_backend_devicen_init:w #2 \s__color_stop }
1012 }

(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.

```

1013 \cs_new_protected:Npn \__color_backend_iccbased_init:nnn #1#2#3
1014 {
1015   \pdf_object_if_exist:nF { __color_icc_ #1 }
1016   {
1017     \pdf_object_new:nn { __color_icc_ #1 } { fstream }
1018     \pdf_object_write:nx { __color_icc_ #1 }
1019     {
1020       {
1021         /N ~ \exp_not:n { #2 } ~
1022         \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
1023       }
1024       {#1}
1025     }
1026   }
1027   \pdf_object_unnamed_write:nx { array }
1028   { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
1029   \cs_if_exist:NT \pdfmanagement_add:nnn
1030   {
1031     \use:x
1032     {
1033       \pdfmanagement_add:nnn { Page / Resources / ColorSpace }
1034       { color \int_use:N \g__color_model_int }
1035       { ~ \pdf_object_ref_last: }
1036     }
1037   }
1038 }

```

(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.

```

1039 \cs_new_protected:Npn \__color_backend_iccbased_device:nnn #1#2#3
1040 {
1041   \pdf_object_if_exist:nF { __color_icc_ #1 }
1042   {
1043     \pdf_object_new:nn { __color_icc_ #1 } { fstream }
1044     \pdf_object_write:nn { __color_icc_ #1 }
1045     {
1046       { /N ~ #3 }
1047       {#1}
1048     }
1049   }

```

```

1049     }
1050     \pdf_object_unnamed_write:nx { array }
1051     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
1052     \cs_if_exist:NT \pdfmanagement_add:nnn
1053     {
1054         \use:x
1055         {
1056             \pdfmanagement_add:nnn
1057             { Page / Resources / ColorSpace }
1058             { Default #2 }
1059             { \pdf_object_ref_last: }
1060         }
1061     }
1062 }

```

(End definition for `__color_backend_iccbased_device:nnn`.)

```

1063 </dvipdfmx | luatex | pdftex | xetex>
1064 <*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.

```

1065 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
1066 {
1067     \cs_gset_protected:Npn \__color_backend_select_separation:nn #1#2 { }
1068     \cs_gset_eq:NN \__color_backend_select_devicen:nn
1069     \__color_backend_select_separation:nn
1070 }

```

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

```

1071 </dvipdfmx | xetex>

```

3.5 Fill and stroke color

Here, dvipdfmx/X_YTeX 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.

```

1072 <*dvipdfmx | luatex | pdftex | xetex>

```

Drawing (fill/stroke) color is handled in dvipdfmx/X_YTeX 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.

```

1073 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1074 { \__color_backend_fill:n { #1 ~ k } }
1075 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1076 { \__color_backend_fill:n { #1 ~ g } }
1077 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1078 { \__color_backend_fill:n { #1 ~ rg } }
1079 \cs_new_protected:Npn \__color_backend_fill:n #1
1080 {
1081     \tl_set:Nn \l__color_backend_fill_tl {#1}

```

```

1082 \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1083 { #1 ~ \l__color_backend_stroke_tl }
1084 \group_insert_after:N \__color_backend_reset:
1085 }
1086 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1087 { \__color_backend_stroke:n { #1 ~ K } }
1088 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1089 { \__color_backend_stroke:n { #1 ~ G } }
1090 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1091 { \__color_backend_stroke:n { #1 ~ RG } }
1092 \cs_new_protected:Npn \__color_backend_stroke:n #1
1093 {
1094   \tl_set:Nn \l__color_backend_stroke_tl {#1}
1095   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1096   { \l__color_backend_fill_tl \c_space_tl #1 }
1097   \group_insert_after:N \__color_backend_reset:
1098 }

```

(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
1099 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
1100 { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
1101 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1102 { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
1103 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1104 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

1105 </dvipdfmx | luatex | pdftex | xetex>
1106 <*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
1107 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
1108 {
1109   \cs_gset_protected:Npn \__color_backend_fill_cmyk:n #1
1110   {
1111     \__kernel_backend_literal:n { pdf: bc ~ [#1] }
1112     \group_insert_after:N \__color_backend_reset:
1113   }
1114   \cs_gset_eq:NN \__color_backend_fill_gray:n \__color_backend_fill_cmyk:n
1115   \cs_gset_eq:NN \__color_backend_fill_rgb:n \__color_backend_fill_cmyk:n
1116   \cs_gset_protected:Npn \__color_backend_reset:
1117   { \__kernel_backend_literal:n { pdf: ec } }
1118   \cs_gset_protected:Npn \__color_backend_stroke:n #1
1119   { \__kernel_backend_literal:n {#1} }
1120   \cs_gset_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1121   \cs_gset_eq:NN \__color_backend_fill_devicen:nn
1122   \__color_backend_fill_separation:nn
1123   \cs_gset_eq:NN \__color_backend_stroke_separation:nn
1124   \__color_backend_fill_separation:nn
1125   \cs_gset_eq:NN \__color_backend_stroke_devicen:nn
1126   \__color_backend_stroke_separation:nn
1127 }

```

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

1128 `</dviptdpmx | xetex>`

1129 `<*dvips>`

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

```

1130 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1131 { \_color_backend_fill:n { cmyk ~ #1 } }
1132 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1133 { \_color_backend_fill:n { gray ~ #1 } }
1134 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1135 { \_color_backend_fill:n { rgb ~ #1 } }
1136 \cs_new_protected:Npn \_color_backend_fill:n #1
1137 {
1138   \_kernel_backend_literal:n { color~push~ #1 }
1139   \group_insert_after:N \_color_backend_reset:
1140 }
1141 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1142 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1143 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1144 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1145 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1146 { \_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
1147 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1148 { \_color_backend_fill:n { separation ~ #1 ~ #2 } }
1149 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1150 { \_kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1151 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1152 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

1153 `</dvips>`

1154 `<*dvisvgm>`

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

```

1155 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1156 { \_color_backend_fill:n { cmyk ~ #1 } }
1157 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1158 { \_color_backend_fill:n { gray ~ #1 } }
1159 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1160 { \_color_backend_fill:n { rgb ~ #1 } }
1161 \cs_new_protected:Npn \_color_backend_fill:n #1
1162 {
1163   \_kernel_backend_literal:n { color~push~ #1 }
1164   \group_insert_after:N \_color_backend_reset:
1165 }

```

(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).

```

1166 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1167 { \__color_backend_cmyk:w #1 \s__color_stop }
1168 \cs_new_protected:Npn \__color_backend_stroke_cmyk:w
1169 #1 ~ #2 ~ #3 ~ #4 \s__color_stop
1170 {
1171   \use:x
1172   {
1173     \__color_backend:nnn
1174     { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1175     { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1176     { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }
1177   }
1178 }
1179 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1180 {
1181   \use:x
1182   {
1183     \__color_backend_stroke_gray_aux:n
1184     { \fp_eval:n { 100 * (#1) } }
1185   }
1186 }
1187 \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1188 { \__color_backend:nnn {#1} {#1} {#1} }
1189 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1190 { \__color_backend_rgb:w #1 \s__color_stop }
1191 \cs_new_protected:Npn \__color_backend_stroke_rgb:w
1192 #1 ~ #2 ~ #3 \s__color_stop
1193 {
1194   \use:x
1195   {
1196     \__color_backend:nnn
1197     { \fp_eval:n { 100 * (#1) } }
1198     { \fp_eval:n { 100 * (#2) } }
1199     { \fp_eval:n { 100 * (#3) } }
1200   }
1201 }
1202 \cs_new_protected:Npx \__color_backend:nnn #1#2#3
1203 {
1204   \__kernel_backend_scope:n
1205   {
1206     stroke =
1207     "
1208     rgb
1209     (
1210       #1 \c_percent_str ,
1211       #2 \c_percent_str ,
1212       #3 \c_percent_str
1213     )
1214     "
1215   }
1216 }

```


(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
1217 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1218 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2 { }
1219 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1220 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn
(End definition for \__color_backend_fill_separation:nn and others.)
1221 </dvisvgm>
1222 </package>

```

4 l3backend-draw Implementation

```

1223 <*package>
1224 <@@=draw>

```

4.1 dvips backend

```

1225 <*dvips>

```

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

```

\__draw_backend_literal:n
\__draw_backend_literal:x
1226 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_postscript:n
1227 \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.

```

1228 \cs_new_protected:Npn \__draw_backend_begin:
1229 {
1230   \__kernel_backend_literal:n { ps::[begin] }
1231   \__draw_backend_literal:n { @beginspecial }
1232 }
1233 \cs_new_protected:Npn \__draw_backend_end:
1234 {
1235   \__draw_backend_literal:n { @endspecial }
1236   \__kernel_backend_literal:n { ps::[end] }
1237 }

```

(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.

```

1238 \cs_new_protected:Npn \__draw_backend_scope_begin:
1239 { \__draw_backend_literal:n { save } }
1240 \cs_new_protected:Npn \__draw_backend_scope_end:
1241 { \__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.

```

1242 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1243 {
1244   \_draw_backend_literal:x
1245   {
1246     \dim_to_decimal_in_bp:n {#1} ~
1247     \dim_to_decimal_in_bp:n {#2} ~ moveto
1248   }
1249 }
1250 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1251 {
1252   \_draw_backend_literal:x
1253   {
1254     \dim_to_decimal_in_bp:n {#1} ~
1255     \dim_to_decimal_in_bp:n {#2} ~ lineto
1256   }
1257 }
1258 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1259 {
1260   \_draw_backend_literal:x
1261   {
1262     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1263     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1264     moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1265   }
1266 }
1267 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1268 {
1269   \_draw_backend_literal:x
1270   {
1271     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1272     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1273     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1274     curveto
1275   }
1276 }

```

(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`

```

1277 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1278 { \bool_gset_true:N \g__draw_draw_eor_bool }
1279 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1280 { \bool_gset_false:N \g__draw_draw_eor_bool }
1281 \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_EX 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`

```

1282 \cs_new_protected:Npn \__draw_backend_closepath:
1283 { \__draw_backend_literal:n { closepath } }
1284 \cs_new_protected:Npn \__draw_backend_stroke:
1285 {
1286   \__draw_backend_literal:n { gsave }
1287   \__draw_backend_literal:n { color.sc }
1288   \__draw_backend_literal:n { stroke }
1289   \__draw_backend_literal:n { grestore }
1290   \bool_if:NT \g__draw_draw_clip_bool
1291   {
1292     \__draw_backend_literal:x
1293     {
1294       \bool_if:NT \g__draw_draw_eor_bool { eo }
1295       clip
1296     }
1297   }
1298   \__draw_backend_literal:n { newpath }
1299   \bool_gset_false:N \g__draw_draw_clip_bool
1300 }
1301 \cs_new_protected:Npn \__draw_backend_closestroke:
1302 {
1303   \__draw_backend_closepath:
1304   \__draw_backend_stroke:
1305 }
1306 \cs_new_protected:Npn \__draw_backend_fill:
1307 {
1308   \__draw_backend_literal:x
1309   {
1310     \bool_if:NT \g__draw_draw_eor_bool { eo }
1311     fill
1312   }
1313   \bool_if:NT \g__draw_draw_clip_bool
1314   {
1315     \__draw_backend_literal:x
1316     {
1317       \bool_if:NT \g__draw_draw_eor_bool { eo }
1318       clip
1319     }
1320   }
1321   \__draw_backend_literal:n { newpath }
1322   \bool_gset_false:N \g__draw_draw_clip_bool
1323 }
1324 \cs_new_protected:Npn \__draw_backend_fillstroke:
1325 {
1326   \__draw_backend_literal:x
1327   {
1328     \bool_if:NT \g__draw_draw_eor_bool { eo }

```

```

1329         fill
1330     }
1331     \__draw_backend_literal:n { gsave }
1332     \__draw_backend_literal:n { color.sc }
1333     \__draw_backend_literal:n { stroke }
1334     \__draw_backend_literal:n { grestore }
1335     \bool_if:NT \g__draw_draw_clip_bool
1336     {
1337         \__draw_backend_literal:x
1338         {
1339             \bool_if:NT \g__draw_draw_eor_bool { eo }
1340             clip
1341         }
1342     }
1343     \__draw_backend_literal:n { newpath }
1344     \bool_gset_false:N \g__draw_draw_clip_bool
1345 }
1346 \cs_new_protected:Npn \__draw_backend_clip:
1347 { \bool_gset_true:N \g__draw_draw_clip_bool }
1348 \bool_new:N \g__draw_draw_clip_bool
1349 \cs_new_protected:Npn \__draw_backend_discardpath:
1350 {
1351     \bool_if:NT \g__draw_draw_clip_bool
1352     {
1353         \__draw_backend_literal:x
1354         {
1355             \bool_if:NT \g__draw_draw_eor_bool { eo }
1356             clip
1357         }
1358     }
1359     \__draw_backend_literal:n { newpath }
1360     \bool_gset_false:N \g__draw_draw_clip_bool
1361 }

```

(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:
1362 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1363 {
1364     \__draw_backend_literal:x
1365     {
1366         [
1367             \exp_args:Nf \use:n
1368             { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1369         ] ~
1370         \dim_to_decimal_in_bp:n {#2} ~ setdash
1371     }
1372 }
1373 \cs_new:Npn \__draw_backend_dash:n #1
1374 { ~ \dim_to_decimal_in_bp:n {#1} }
1375 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1376 {
1377     \__draw_backend_literal:x
1378     { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }

```

```

1379 }
1380 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1381 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1382 \cs_new_protected:Npn \__draw_backend_cap_butt:
1383 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1384 \cs_new_protected:Npn \__draw_backend_cap_round:
1385 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1386 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1387 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1388 \cs_new_protected:Npn \__draw_backend_join_miter:
1389 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1390 \cs_new_protected:Npn \__draw_backend_join_round:
1391 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1392 \cs_new_protected:Npn \__draw_backend_join_bevel:
1393 { \__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.

```

1394 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1395 {
1396   \__draw_backend_literal:n
1397   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1398 }

```

(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).

```

1399 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1400 {
1401   \__draw_backend_literal:n { @endspecial }
1402   \__draw_backend_literal:n { [end] }
1403   \__draw_backend_literal:n { [begin] }
1404   \__draw_backend_literal:n { save }
1405   \__draw_backend_literal:n { currentpoint }
1406   \__draw_backend_literal:n { currentpoint~translate }
1407   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1408   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1409   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1410   \__draw_backend_literal:n { neg~exch~neg~exch~translate }

```

```

1411     \__draw_backend_literal:n { [end] }
1412     \hbox_overlap_right:n { \box_use:N #1 }
1413     \__draw_backend_literal:n { [begin] }
1414     \__draw_backend_literal:n { restore }
1415     \__draw_backend_literal:n { [end] }
1416     \__draw_backend_literal:n { [begin] }
1417     \__draw_backend_literal:n { @beginspecial }
1418 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1419 </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.

```

1420 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

__draw_backend_literal:n Pass data through using a dedicated interface.

```

\__draw_backend_literal:x
1421 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1422 \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:
1423 \cs_new_protected:Npn \__draw_backend_begin:
1424 { \__draw_backend_scope_begin: }
1425 \cs_new_protected:Npn \__draw_backend_end:
1426 { \__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:
1427 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1428 \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
1429 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1430 {
1431     \__draw_backend_literal:x
1432     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1433 }
1434 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1435 {
1436     \__draw_backend_literal:x
1437     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1438 }
1439 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1440 {

```

```

1441 \_draw_backend_literal:x
1442 {
1443   \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1444   \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1445   \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1446   c
1447 }
1448 }
1449 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1450 {
1451   \_draw_backend_literal:x
1452   {
1453     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1454     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1455     re
1456   }
1457 }

```

(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.

```

1458 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1459 { \bool_gset_true:N \g__draw_draw_eor_bool }
1460 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1461 { \bool_gset_false:N \g__draw_draw_eor_bool }
1462 \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.

```

1463 \cs_new_protected:Npn \_draw_backend_closepath:
1464 { \_draw_backend_literal:n { h } }
1465 \cs_new_protected:Npn \_draw_backend_stroke:
1466 { \_draw_backend_literal:n { S } }
1467 \cs_new_protected:Npn \_draw_backend_closestroke:
1468 { \_draw_backend_literal:n { s } }
1469 \cs_new_protected:Npn \_draw_backend_fill:
1470 {
1471   \_draw_backend_literal:x
1472   { f \bool_if:NT \g__draw_draw_eor_bool * }
1473 }
1474 \cs_new_protected:Npn \_draw_backend_fillstroke:
1475 {
1476   \_draw_backend_literal:x
1477   { B \bool_if:NT \g__draw_draw_eor_bool * }
1478 }
1479 \cs_new_protected:Npn \_draw_backend_clip:
1480 {
1481   \_draw_backend_literal:x
1482   { W \bool_if:NT \g__draw_draw_eor_bool * }
1483 }
1484 \cs_new_protected:Npn \_draw_backend_discardpath:
1485 { \_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:
1486 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1487 {
1488   \__draw_backend_literal:x
1489   {
1490     [
1491       \exp_args:Nf \use:n
1492       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1493     ] ~
1494     \dim_to_decimal_in_bp:n {#2} ~ d
1495   }
1496 }
1497 \cs_new:Npn \__draw_backend_dash:n #1
1498 { ~ \dim_to_decimal_in_bp:n {#1} }
1499 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1500 {
1501   \__draw_backend_literal:x
1502   { \dim_to_decimal_in_bp:n {#1} ~ w }
1503 }
1504 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1505 { \__draw_backend_literal:x { #1 ~ M } }
1506 \cs_new_protected:Npn \__draw_backend_cap_but:
1507 { \__draw_backend_literal:n { 0 ~ J } }
1508 \cs_new_protected:Npn \__draw_backend_cap_round:
1509 { \__draw_backend_literal:n { 1 ~ J } }
1510 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1511 { \__draw_backend_literal:n { 2 ~ J } }
1512 \cs_new_protected:Npn \__draw_backend_join_miter:
1513 { \__draw_backend_literal:n { 0 ~ j } }
1514 \cs_new_protected:Npn \__draw_backend_join_round:
1515 { \__draw_backend_literal:n { 1 ~ j } }
1516 \cs_new_protected:Npn \__draw_backend_join_bevel:
1517 { \__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_YTeX. In the former, we have a direct method to maintain alignment: the backend can use a matrix itself. For dvipdfmx/X_YTeX, 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_YTeX, but as a matched pair so not suitable for the “stand alone” transformation set up here.) The specials used here are from xdvipdfmx originally: they are well-tested, but probably equivalent to the pdf₊ versions!

```

1518 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1519 {
1520   <*luatex | pdftex>
1521   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1522   </luatex | pdftex>
1523   <*dvipdfmx | xetex>
1524   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1525   \__draw_backend_cm_aux:nnnn
1526   </dvipdfmx | xetex>

```



```

1527 }
1528 <*dvipdfmx | xetex>
1529 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1530 {
1531   \__kernel_backend_literal:x
1532   {
1533     x:rotate~
1534     \fp_compare:nNnTF {#1} = \c_zero_fp
1535       { 0 }
1536       { \fp_eval:n { round ( -#1 , 5 ) } }
1537   }
1538   \__kernel_backend_literal:x
1539   {
1540     x:scale~
1541     \fp_eval:n { round ( #2 , 5 ) } ~
1542     \fp_eval:n { round ( #3 , 5 ) }
1543   }
1544   \__kernel_backend_literal:x
1545   {
1546     x:rotate~
1547     \fp_compare:nNnTF {#4} = \c_zero_fp
1548       { 0 }
1549       { \fp_eval:n { round ( -#4 , 5 ) } }
1550   }
1551 }
1552 </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.

```

1553 <*dvipdfmx|xetex>
1554 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1555 {
1556   \use:x
1557   {
1558     \__draw_backend_cm_decompose_auxi:nnnnN
1559     { \fp_eval:n { (#1 + #4) / 2 } }
1560     { \fp_eval:n { (#1 - #4) / 2 } }
1561     { \fp_eval:n { (#3 + #2) / 2 } }
1562     { \fp_eval:n { (#3 - #2) / 2 } }
1563   }
1564   #5
1565 }
1566 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1567 {
1568   \use:x
1569   {
1570     \__draw_backend_cm_decompose_auxii:nnnnN
1571     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1572     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1573     { \fp_eval:n { atand ( #3 , #2 ) } }
1574     { \fp_eval:n { atand ( #4 , #1 ) } }
1575   }
1576   #5
1577 }
1578 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1579 {
1580   \use:x
1581   {
1582     \__draw_backend_cm_decompose_auxiii:nnnnN
1583     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1584     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1585     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1586     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1587   }
1588   #5
1589 }
1590 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1591 {
1592   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1593   { #5 {#1} {#2} {#3} {#4} }
1594   { #5 {#1} {#3} {#2} {#4} }
1595 }
1596 </dvipdfmx|xetex>

```

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

`__draw_backend_box_use:Nnnnn` Inserting a \TeX box transformed to the requested position and using the current matrix is done using a mixture of \TeX and low-level manipulation. The offset can be handled by \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.

```

1597 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1598 {
1599   \__kernel_backend_scope_begin:
1600   <*luatex | pdftex>
1601   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1602   </luatex | pdftex>
1603   <*dvipdfmx | xetex>
1604   \__kernel_backend_literal:n
1605   { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1606   </dvipdfmx | xetex>
1607   \hbox_overlap_right:n { \box_use:N #1 }
1608   <*dvipdfmx | xetex>
1609   \__kernel_backend_literal:n { pdf:etrans }
1610   </dvipdfmx | xetex>
1611   \__kernel_backend_scope_end:
1612 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1613 </dvipdfmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1614 <*dvisvgm>

```

__draw_backend_literal:n The same as the more general literal call.

```

\__draw_backend_literal:x
1615 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1616 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for __draw_backend_literal:n.)

__draw_backend_scope_begin: Use the backend-level scope mechanisms.

```

\__draw_backend_scope_end:
1617 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1618 \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_begin: 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

```

1619 \cs_new_protected:Npn \__draw_backend_begin:
1620 {
1621   \__kernel_backend_scope_begin:
1622   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1623 }
1624 \cs_new_eq:NN \__draw_backend_end: \__kernel_backend_scope_end:

```

(End definition for __draw_backend_begin: and __draw_backend_end:.)

__draw_backend_moveto:nn Once again, some work is needed to get path constructs correct. Rather than write the 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_lineto:nn
\__draw_backend_rectangle:nnnn
\__draw_backend_curveto:nnnnnn
\__draw_backend_add_to_path:n
\g__draw_backend_path_tl
1625 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1626 {

```

```

1627   \__draw_backend_add_to_path:n
1628   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1629 }
1630 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1631 {
1632   \__draw_backend_add_to_path:n
1633   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1634 }
1635 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1636 {
1637   \__draw_backend_add_to_path:n
1638   {
1639     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1640     h ~ \dim_to_decimal:n {#3} ~
1641     v ~ \dim_to_decimal:n {#4} ~
1642     h ~ \dim_to_decimal:n { -#3 } ~
1643     Z
1644   }
1645 }
1646 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1647 {
1648   \__draw_backend_add_to_path:n
1649   {
1650     C ~
1651     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1652     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1653     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1654   }
1655 }
1656 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1657 {
1658   \tl_gset:Nx \g__draw_backend_path_tl
1659   {
1660     \g__draw_backend_path_tl
1661     \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1662     #1
1663   }
1664 }
1665 \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: 1666 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1667   { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1668 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1669   { \__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.

```

1670 \cs_new_protected:Npn \__draw_backend_closepath:
1671 { \__draw_backend_add_to_path:n { Z } }
1672 \cs_new_protected:Npn \__draw_backend_path:n #1
1673 {
1674   \bool_if:NTF \g__draw_draw_clip_bool
1675   {
1676     \int_gincr:N \g__kernel_clip_path_int
1677     \__draw_backend_literal:x
1678     {
1679       < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1680       { ?nl }
1681       <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1682       < /clipPath > { ? nl }
1683       <
1684       use~xlink:href =
1685       "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1686       #1
1687       />
1688     }
1689     \__kernel_backend_scope:x
1690     {
1691       clip-path =
1692       "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1693     }
1694   }
1695   {
1696     \__draw_backend_literal:x
1697     { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1698   }
1699   \tl_gclear:N \g__draw_backend_path_tl
1700   \bool_gset_false:N \g__draw_draw_clip_bool
1701 }
1702 \int_new:N \g__draw_backend_path_int
1703 \cs_new_protected:Npn \__draw_backend_stroke:
1704 { \__draw_backend_path:n { style="fill:none" } }
1705 \cs_new_protected:Npn \__draw_backend_closestroke:
1706 {
1707   \__draw_backend_closepath:
1708   \__draw_backend_stroke:
1709 }
1710 \cs_new_protected:Npn \__draw_backend_fill:
1711 { \__draw_backend_path:n { style="stroke:none" } }
1712 \cs_new_protected:Npn \__draw_backend_fillstroke:
1713 { \__draw_backend_path:n { } }
1714 \cs_new_protected:Npn \__draw_backend_clip:
1715 { \bool_gset_true:N \g__draw_draw_clip_bool }
1716 \bool_new:N \g__draw_draw_clip_bool
1717 \cs_new_protected:Npn \__draw_backend_discardpath:
1718 {
1719   \bool_if:NT \g__draw_draw_clip_bool
1720   {
1721     \int_gincr:N \g__kernel_clip_path_int
1722     \__draw_backend_literal:x
1723     {

```

```

1724         < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1725         { ?nl }
1726         <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1727         < /clipPath >
1728     }
1729     \__kernel_backend_scope:x
1730     {
1731         clip-path =
1732         "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1733     }
1734 }
1735 \tl_gclear:N \g__draw_path_tl
1736 \bool_gset_false:N \g__draw_draw_clip_bool
1737 }

```

(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:
1738 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1739 {
1740     \use:x
1741     {
1742         \__draw_backend_dash_aux:nn
1743         { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1744         { \dim_to_decimal:n {#2} }
1745     }
1746 }
1747 \cs_new:Npn \__draw_backend_dash:n #1
1748 { , \dim_to_decimal_in_bp:n {#1} }
1749 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1750 {
1751     \__kernel_backend_scope:x
1752     {
1753         stroke-dasharray =
1754         "
1755         \tl_if_empty:nTF {#1}
1756         { none }
1757         { \use_none:n #1 }
1758         " ~
1759         stroke-offset=" #2 "
1760     }
1761 }
1762 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1763 { \__kernel_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1764 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1765 { \__kernel_backend_scope:x { stroke-miterlimit=" #1 " } }
1766 \cs_new_protected:Npn \__draw_backend_cap_butt:
1767 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1768 \cs_new_protected:Npn \__draw_backend_cap_round:
1769 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1770 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1771 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1772 \cs_new_protected:Npn \__draw_backend_join_miter:

```

```

1773 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1774 \cs_new_protected:Npn \__draw_backend_join_round:
1775 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1776 \cs_new_protected:Npn \__draw_backend_join_bevel:
1777 { \__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.

```

1778 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1779 {
1780   \__kernel_backend_scope:n
1781   {
1782     transform =
1783     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1784   }
1785 }

```

(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.

```

1786 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1787 {
1788   \__kernel_backend_scope_begin:
1789   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1790   \__kernel_backend_literal_svg:n
1791   {
1792     < g~
1793     stroke="none"~
1794     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1795     >
1796   }
1797   \box_set_wd:Nn #1 { Opt }
1798   \box_set_ht:Nn #1 { Opt }
1799   \box_set_dp:Nn #1 { Opt }
1800   \box_use:N #1
1801   \__kernel_backend_literal_svg:n { </g> }
1802   \__kernel_backend_scope_end:
1803 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1804 </dvisvgm>
1805 </package>

```

5 l3backend-graphics Implementation

```

1806 <*package>
1807 <@@=graphics>

```

5.1 dvips backend

```

1808 <*dvips>

```

`_graphics_backend_getbb_eps:n` Simply use the generic function.

```

1809 \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.
1810 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1811 {
1812   \_kernel_backend_literal:x
1813   {
1814     PSfile = #1 \c_space_tl
1815     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1816     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1817     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1818     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1819   }
1820 }
(End definition for \_graphics_backend_include_eps:n.)
1821 </dvips>

```

5.2 LuaTeX and pdfTeX backends

1822 <*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.

```

1823 \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.

```

1824 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1825 {
1826   \int_zero:N \l_graphics_page_int
1827   \tl_clear:N \l_graphics_pagebox_tl
1828   \tl_set:Nx \l__graphics_graphics_attr_tl
1829   {
1830     \tl_if_empty:NF \l_graphics_decodearray_tl
1831     { :D \l_graphics_decodearray_tl }
1832     \bool_if:NT \l_graphics_interpolate_bool
1833     { :I }
1834   }
1835   \tl_clear:N \l__graphics_graphics_attr_tl
1836   \_graphics_backend_getbb_auxi:n {#1}
1837 }
1838 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n

```



```

1839 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1840 {
1841   \tl_clear:N \l_graphics_decodearray_tl
1842   \bool_set_false:N \l_graphics_interpolate_bool
1843   \tl_set:Nx \l__graphics_graphics_attr_tl
1844   {
1845     : \l_graphics_pagebox_tl
1846     \int_compare:nNnT \l_graphics_page_int > 1
1847     { :P \int_use:N \l_graphics_page_int }
1848   }
1849   \__graphics_backend_getbb_auxi:n {#1}
1850 }
1851 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1852 {
1853   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1854   { \__graphics_backend_getbb_auxii:n {#1} }
1855 }

```

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.

```

1856 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1857 {
1858   \tex_immediate:D \tex_pdfximage:D
1859   \bool_lazy_or:nnT
1860   { \l_graphics_interpolate_bool }
1861   { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1862   {
1863     attr ~
1864     {
1865       \tl_if_empty:NF \l_graphics_decodearray_tl
1866       { /Decode~[ \l_graphics_decodearray_tl ] }
1867       \bool_if:NT \l_graphics_interpolate_bool
1868       { /Interpolate~true }
1869     }
1870   }
1871   \int_compare:nNnT \l_graphics_page_int > 0
1872   { page ~ \int_use:N \l_graphics_page_int }
1873   \tl_if_empty:NF \l_graphics_pagebox_tl
1874   { \l_graphics_pagebox_tl }
1875   {#1}
1876   \hbox_set:Nn \l__graphics_internal_box
1877   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1878   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1879   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1880   \int_const:cn { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1881   { \tex_the:D \tex_pdflastximage:D }
1882   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1883 }

```

(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.

```

1884 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1885 {
1886   \tex_pdfrefximage:D
1887   \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1888 }
1889 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1890 \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^AT_EX 2_ε 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
1891 \sys_if_shell:T
1892 {
1893   \str_new:N \l__graphics_backend_dir_str
1894   \str_new:N \l__graphics_backend_name_str
1895   \str_new:N \l__graphics_backend_ext_str
1896   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1897   {
1898     \file_parse_full_name:nNNN {#1}
1899     \l__graphics_backend_dir_str
1900     \l__graphics_backend_name_str
1901     \l__graphics_backend_ext_str
1902     \exp_args:Nx \__graphics_backend_getbb_eps:nn
1903     {
1904       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1905       -converted-to.pdf
1906     }
1907     {#1}
1908   }
1909   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1910   {
1911     \file_compare_timestamp:nNnT {#2} > {#1}
1912     {
1913       \sys_shell_now:n
1914       { repstopdf ~ #2 ~ #1 }
1915     }
1916     \tl_set:Nn \l__graphics_name_tl {#1}
1917     \__graphics_backend_getbb_pdf:n {#1}
1918   }
1919   \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1920   {
1921     \file_parse_full_name:nNNN {#1}
1922     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1923     \exp_args:Nx \__graphics_backend_include_pdf:n
1924     {
1925       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1926       -converted-to.pdf
1927     }
1928   }
1929 }

```

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

1930 \langle /luatex | pdftex \rangle

5.3 dvipdfmx backend

1931 \langle *dvipdfmx | xetex \rangle

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
1932 \cs_new_eq:NN \_graphics_backend_getbb_eps:n \graphics_read_bb:n
1933  $\langle$ *dvipdfmx $\rangle$ 
1934 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1935 {
1936   \int_zero:N \l_graphics_page_int
1937   \tl_clear:N \l_graphics_pagebox_tl
1938   \graphics_extract_bb:n {#1}
1939 }
1940 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1941 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1942 {
1943   \tl_clear:N \l_graphics_decodearray_tl
1944   \bool_set_false:N \l_graphics_interpolate_bool
1945   \graphics_extract_bb:n {#1}
1946 }
1947  $\langle$ /dvipdfmx $\rangle$ 

```

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

`\g__graphics_track_int` Used to track the object number associated with each graphic.

1948 \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_{La}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
\_graphics_backend_include_auxi:nn
\_graphics_backend_include_auxii:nnn
\_graphics_backend_include_auxiii:nnnn
1949 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1950 {
1951   \__kernel_backend_literal:x
1952   {
1953     PSfile = #1 \c_space_tl
1954     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1955     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1956     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1957     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1958   }
1959 }
1960 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1961 { \_graphics_backend_include_auxi:nn {#1} { image } }
1962 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1963  $\langle$ *dvipdfmx $\rangle$ 
1964 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1965 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1966  $\langle$ /dvipdfmx $\rangle$ 

```

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.

```

1967 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1968 {
1969   \__graphics_backend_include_auxii:xnn
1970   {
1971     \tl_if_empty:NF \l_graphics_pagebox_tl
1972     { : \l_graphics_pagebox_tl }
1973     \int_compare:nNnT \l_graphics_page_int > 1
1974     { :P \int_use:N \l_graphics_page_int }
1975     \tl_if_empty:NF \l_graphics_decodearray_tl
1976     { :D \l_graphics_decodearray_tl }
1977     \bool_if:NT \l_graphics_interpolate_bool
1978     { :I }
1979   }
1980   {#1} {#2}
1981 }
1982 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1983 {
1984   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1985   {
1986     \__kernel_backend_literal:x
1987     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1988   }
1989   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1990 }
1991 \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!

```

1992 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1993 {
1994   \int_gincr:N \g__graphics_track_int
1995   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1996   \__kernel_backend_literal:x
1997   {
1998     pdf:#3~
1999     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
2000     \int_compare:nNnT \l_graphics_page_int > 1
2001     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
2002     \tl_if_empty:NF \l_graphics_pagebox_tl
2003     {
2004       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
2005       bbox ~
2006       \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
2007       \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2008       \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2009       \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
2010     }
2011     (#1)
2012     \bool_lazy_or:nnT

```

```

2013         { \l_graphics_interpolate_bool }
2014         { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
2015         {
2016             <<
2017             \tl_if_empty:NF \l_graphics_decodearray_tl
2018             { /Decode~[ \l_graphics_decodearray_tl ] }
2019             \bool_if:NT \l_graphics_interpolate_bool
2020             { /Interpolate~true> }
2021             >>
2022         }
2023     }
2024 }

```

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

```

2025 </dvipdfmx | xetex>

```

5.4 X_YTeX backend

```

2026 <*xetex>

```

5.4.1 Images

For X_YTeX, 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_YTeX 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_auxiii:nNnn
\__graphics_backend_getbb_auxiv:nNnn
\__graphics_backend_getbb_auxiv:NnNnn
\__graphics_backend_getbb_auxv:nNnn
\__graphics_backend_getbb_auxv:NnNnn
\__graphics_backend_getbb_pagebox:w
2027 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2028 {
2029     \int_zero:N \l_graphics_page_int
2030     \tl_clear:N \l_graphics_pagebox_tl
2031     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2032 }
2033 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
2034 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2035 {
2036     \tl_clear:N \l_graphics_decodearray_tl
2037     \bool_set_false:N \l_graphics_interpolate_bool
2038     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2039 }
2040 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2041 {
2042     \int_compare:nNnTF \l_graphics_page_int > 1
2043     { \__graphics_backend_getbb_auxii:Nn \l_graphics_page_int {#1} #2 }
2044     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2045 }
2046 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nNn #1#2#3
2047 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2048 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nNn { V }
2049 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2050 {
2051     \tl_if_empty:NTF \l_graphics_pagebox_tl
2052     { \__graphics_backend_getbb_auxiv:NnNnn \l_graphics_pagebox_tl }
2053     { \__graphics_backend_getbb_auxv:nNnn }
2054     {#1} #2 {#3} {#4}

```

```

2055 }
2056 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2057 {
2058   \use:x
2059   {
2060     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2061     { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
2062   }
2063 }
2064 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2065 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2066 {
2067   \graphics_bb_restore:nF {#1#3}
2068   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2069 }
2070 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2071 {
2072   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
2073   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2074   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2075   \graphics_bb_save:n {#1#3}
2076 }
2077 \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

__graphics_backend_include_pdf:n For PDF graphics, properly supporting the pagebox concept in X_YTeX is best done using the \tex_XeTeXpdffile:D primitive. The syntax here is the same as for the graphic measurement part, although we know at this stage that there must be some valid setting for \l_graphics_pagebox_tl.

```

2078 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2079 {
2080   \tex_XeTeXpdffile:D
2081   \__graphics_backend_include_pdf_quote:w #1 "#1" \s__graphics_stop \c_space_tl
2082   \int_compare:nNnT \l_graphics_page_int > 0
2083     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
2084   \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
2085 }
2086 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \s__graphics_stop
2087 { " #2 " }

```

(End definition for __graphics_backend_include_pdf:n and __graphics_backend_include_bitmap_quote:w.)

```
2088 </xetex>
```

5.5 dvisvgm backend

```
2089 <*dvisvgm>
```

__graphics_backend_getbb_eps:n Simply use the generic function.

```
2090 \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`

```

2091 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
2092 {
2093   \int_zero:N \l_graphics_page_int
2094   \tl_clear:N \l_graphics_pagebox_tl
2095   \graphics_extract_bb:n {#1}
2096 }
2097 \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

```

2098 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
2099 {
2100   \tl_clear:N \l_graphics_decodearray_tl
2101   \bool_set_false:N \l_graphics_interpolate_bool
2102   \graphics_extract_bb:n {#1}
2103 }

```

(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`

```

2104 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
2105 { \_graphics_backend_include:nn { PSfile } {#1} }
2106 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
2107 { \_graphics_backend_include:nn { pdffile } {#1} }
2108 \cs_new_protected:Npn \_graphics_backend_include:nn #1#2
2109 {
2110   \__kernel_backend_literal:x
2111   {
2112     #1 = #2 \c_space_tl
2113     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
2114     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2115     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2116     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
2117   }
2118 }

```

(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.

```

2119 \cs_new_protected:Npn \_graphics_backend_include_png:n #1
2120 {
2121   \__kernel_backend_literal:x
2122   {
2123     dvisvgm:img~
2124     \dim_to_decimal:n { \l_graphics_ury_dim } ~
2125     \dim_to_decimal:n { \l_graphics_ury_dim } ~

```

```

2126         \_graphics_backend_include_bitmap_quote:w #1 " #1 " \s__graphics_stop
2127     }
2128 }
2129 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_png:n
2130 \cs_new:Npn \_graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \s__graphics_stop
2131 { " #2 " }

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

2132 </divsvgm>
2133 </package>

```

6 l3backend-pdf Implementation

```

2134 <*package>
2135 <@=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

2136 \box_new:N \l__pdf_internal_box

(End definition for \l__pdf_internal_box.)

```

6.2 dvips backend

```

2137 <*dvips>

\_pdf_backend_pdfmark:n Used often enough it should be a separate function.
\_pdf_backend_pdfmark:x
2138 \cs_new_protected:Npn \_pdf_backend_pdfmark:n #1
2139 { \_kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2140 \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
2141 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2142 { \_pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2143 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2144 { \_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.

```
2145 \int_new:N \g__pdf_backend_object_int
2146 \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.

```
2147 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2148 {
2149   \int_gincr:N \g__pdf_backend_object_int
2150   \int_const:cn
2151     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2152     { \g__pdf_backend_object_int }
2153   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2154 }
2155 \cs_new:Npn \__pdf_backend_object_ref:n #1
2156 { { 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.

```
2157 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2158 {
2159   \__pdf_backend_pdfmark:x
2160   {
2161     /_objdef ~ \__pdf_backend_object_ref:n {#1}
2162     /type
2163     \str_case_e:nn
2164       { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2165       {
2166         { array } { /array }
2167         { dict } { /dict }
2168         { fstream } { /stream }
2169         { stream } { /stream }
2170       }
2171     /OBJ
2172   }
2173   \use:c
2174     { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2175     { \__pdf_backend_object_ref:n {#1} } {#2}
2176 }
2177 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2178 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2179 {
2180   \__pdf_backend_pdfmark:x
2181     { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2182 }
2183 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2184 {
2185   \__pdf_backend_pdfmark:x
2186     { #1 << \exp_not:n {#2} >> /PUT }
2187 }
2188 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
```

```

2189 {
2190   \exp_args:Nx
2191   \__pdf_backend_object_write_fstream:nnn {#1} #2
2192 }
2193 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2194 {
2195   \__kernel_backend_postscript:n
2196   {
2197     SDict ~ begin ~
2198     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2199     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2200     end
2201   }
2202 }
2203 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2204 {
2205   \exp_args:Nx
2206   \__pdf_backend_object_write_stream:nnn {#1} #2
2207 }
2208 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2209 {
2210   \__kernel_backend_postscript:n
2211   {
2212     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2213     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2214   }
2215 }

```

(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.

```

2216 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2217 {
2218   \int_gincr:N \g__pdf_backend_object_int
2219   \__pdf_backend_pdfmark:x
2220   {
2221     /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2222     /type
2223     \str_case:nn
2224       {#1}
2225       {
2226         { array } { /array }
2227         { dict } { /dict }
2228         { fstream } { /stream }
2229         { stream } { /stream }
2230       }
2231     /OBJ
2232   }
2233   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2234   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2235 }
2236 \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.

```
2237 \cs_new:Npn \__pdf_backend_object_last:
2238 { { 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.

```
2239 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2240 { { 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.

```
2241 \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.

```
2242 \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.

```
2243 \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^AT_EX 2_ε picture of zero size). Once the data is collected, use it to set up the annotation border.

```
2244 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2245 {
2246   \exp_args:Nf \__pdf_backend_annotation_aux:nnnn
2247     { \dim_eval:n {#1} } {#2} {#3} {#4}
2248 }
2249 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2250 {
2251   \box_move_down:nn {#3}
2252   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2253   \box_move_up:nn {#2}
2254   {
2255     \hbox:n
2256     {
2257       \__kernel_kern:n {#1}
2258       \__kernel_backend_postscript:n { pdf.save.ur }
2259       \__kernel_kern:n { -#1 }
2260     }
2261   }
2262 }
```

```

2261     }
2262     \int_gincr:N \g__pdf_backend_object_int
2263     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2264     \__pdf_backend_pdfmark:x
2265     {
2266         /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2267         pdf.rect
2268         #4 ~
2269         /ANN
2270     }
2271 }

```

(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.

```

2272 \cs_new:Npn \__pdf_backend_annotation_last:
2273 { { 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.

```

2274 \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.

```

2275 \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.

```

2276 \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.

```

2277 \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.

```

2278 \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.

```

2279 \tl_new:N \l__pdf_breaklink_pdfmark_tl
2280 \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.

```

2281 \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.

2282 `\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.

2283 `\cs_new_protected:Npn _pdf_backend_link_begin_goto:nnw #1#2`

2284 `{`

2285 `_pdf_backend_link_begin:nw`

2286 `{ #1 /Subtype /Link /Action << /S /GoTo /D (#2) >> }`

2287 `}`

2288 `\cs_new_protected:Npn _pdf_backend_link_begin_user:nnw #1#2`

2289 `{ _pdf_backend_link_begin:nw {#1#2} }`

2290 `\cs_new_protected:Npn _pdf_backend_link_begin:nw #1`

2291 `{`

2292 `\bool_if:NF \g__pdf_backend_link_bool`

2293 `{ _pdf_backend_link_begin_aux:nw {#1} }`

2294 `}`

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.

2295 `\cs_new_protected:Npn _pdf_backend_link_begin_aux:nw #1`

2296 `{`

2297 `\bool_gset_true:N \g__pdf_backend_link_bool`

2298 `_kernel_backend_postscript:n`

2299 `{ /pdf.link.dict (#1) def }`

2300 `\tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}`

2301 `_pdf_backend_link_sf_save:`

2302 `\mode_if_math:TF`

2303 `{ \bool_gset_true:N \g__pdf_backend_link_math_bool }`

2304 `{ \bool_gset_false:N \g__pdf_backend_link_math_bool }`

2305 `\hbox_set:Nw \l__pdf_backend_content_box`

2306 `_pdf_backend_link_sf_restore:`

2307 `\bool_if:NT \g__pdf_backend_link_math_bool`

2308 `{ \c_math_toggle_token }`

2309 `}`

2310 `\cs_new_protected:Npn _pdf_backend_link_end:`

```

2311 {
2312   \bool_if:NT \g__pdf_backend_link_bool
2313   { \__pdf_backend_link_end_aux: }
2314 }
2315 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2316 {
2317   \bool_if:NT \g__pdf_backend_link_math_bool
2318   { \c_math_toggle_token }
2319   \__pdf_backend_link_sf_save:
2320   \hbox_set_end:
2321   \__pdf_backend_link_minima:
2322   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2323   \exp_args:Nx \__pdf_backend_link_outerbox:n
2324   {
2325     \int_if_odd:nTF { \value { page } }
2326     { \oddsidemargin }
2327     { \evensidemargin }
2328   }
2329   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2330   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2331   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2332   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2333   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2334   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2335   {
2336     \hbox:n
2337     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2338   }
2339   \int_gincr:N \g__pdf_backend_object_int
2340   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2341   \__kernel_backend_postscript:x
2342   {
2343     mark
2344     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2345     \g__pdf_backend_link_dict_tl \c_space_tl
2346     pdf.rect
2347     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2348   }
2349   \__pdf_backend_link_sf_restore:
2350   \bool_gset_false:N \g__pdf_backend_link_bool
2351 }
2352 \cs_new_protected:Npn \__pdf_backend_link_minima:
2353 {
2354   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2355   \__kernel_backend_postscript:x
2356   {
2357     /pdf.linkdp.pad ~
2358     \dim_to_decimal:n
2359     {
2360       \dim_max:nn
2361       {
2362         \box_dp:N \l__pdf_backend_model_box
2363         - \box_dp:N \l__pdf_backend_content_box
2364       }

```

```

2365         { Opt }
2366     } ~
2367     pdf.pt.dvi ~ def
2368 /pdf.linkht.pad ~
2369 \dim_to_decimal:n
2370 {
2371     \dim_max:nn
2372     {
2373         \box_ht:N \l__pdf_backend_model_box
2374         - \box_ht:N \l__pdf_backend_content_box
2375     }
2376     { Opt }
2377 } ~
2378 pdf.pt.dvi ~ def
2379 }
2380 }
2381 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2382 {
2383     \__kernel_backend_postscript:x
2384     {
2385         /pdf.outerbox
2386         [
2387             \dim_to_decimal:n {#1} ~
2388             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2389             \dim_to_decimal:n { #1 + \textwidth } ~
2390             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2391         ]
2392         [ exch { pdf.pt.dvi } forall ] def
2393     /pdf.baselineskip ~
2394     \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2395     { pdf.pt.dvi ~ def }
2396     { pop ~ pop }
2397     ifelse
2398 }
2399 }
2400 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2401 {
2402     \int_gset:Nn \g__pdf_backend_link_sf_int
2403     {
2404         \mode_if_horizontal:TF
2405         { \tex_spacefactor:D }
2406         { 0 }
2407     }
2408 }
2409 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2410 {
2411     \mode_if_horizontal:T
2412     {
2413         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2414         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2415     }
2416 }

```

(End definition for __pdf_backend_link_begin_goto:nnw 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^AT_EX 2_ε end.

```

2417 \use_none:n
2418 {
2419   \cs_if_exist:NT \@makecol@hook
2420   {
2421     \tl_put_right:Nn \@makecol@hook
2422     {
2423       \box_if_empty:NF \@cclv
2424       {
2425         \vbox_set:Nn \@cclv
2426         {
2427           \__kernel_backend_postscript:n
2428           {
2429             pdf.globaldict /pdf.brokenlink.rect ~ known
2430             { pdf.bordertracking.continue }
2431             if
2432           }
2433           \vbox_unpack_drop:N \@cclv
2434           \__kernel_backend_postscript:n
2435           { pdf.bordertracking.endpage }
2436         }
2437       }
2438     }
2439     \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2440     \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2441     \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2442   }
2443 }

```

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

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

```

2444 \cs_new:Npn \__pdf_backend_link_last:
2445 { { 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.

```

2446 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2447 {
2448   \__kernel_backend_postscript:x
2449   {
2450     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2451   }
2452 }

```

(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.


```

2453 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2454 {
2455   \__kernel_backend_postscript:n { pdf.dest.anchor }
2456   \__pdf_backend_pdfmark:x
2457   {
2458     /View
2459     [
2460       \str_case:nnF {#2}
2461       {
2462         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2463         { fit } { /Fit }
2464         { fitb } { /FitB }
2465         { fitbh } { /FitBH ~ pdf.dest.y }
2466         { fitbv } { /FitBV ~ pdf.dest.x }
2467         { fith } { /FitH ~ pdf.dest.y }
2468         { fitv } { /FitV ~ pdf.dest.x }
2469         { fitr } { /Fit }
2470       }
2471       {
2472         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2473       }
2474     ]
2475     /Dest ( \exp_not:n {#1} ) cvn
2476     /DEST
2477   }
2478 }
2479 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2480 {
2481   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2482   { \dim_eval:n {#2} } {#1} {#3} {#4}
2483 }
2484 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2485 {
2486   \vbox_to_zero:n
2487   {
2488     \__kernel_kern:n {#4}
2489     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2490     \tex_vss:D
2491   }
2492   \__kernel_kern:n {#1}
2493   \vbox_to_zero:n
2494   {
2495     \__kernel_kern:n { -#3 }
2496     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2497     \tex_vss:D
2498   }
2499   \__kernel_kern:n { -#1 }
2500   \__pdf_backend_pdfmark:n
2501   {
2502     /View
2503     [
2504       /FitR ~
2505       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2506       pdf.urx ~ pdf.ury ~ pdf.dest2device

```

```

2507     ]
2508     /Dest ( #2 ) cvn
2509     /DEST
2510   }
2511 }

```

(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
2512 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2513 {
2514   \int_compare:nNnT {#1} = 0
2515   {
2516     \__kernel_backend_literal_postscript:n
2517     {
2518       /setdistillerparams ~ where
2519       { pop << /CompressPages ~ false >> setdistillerparams }
2520       if
2521     }
2522   }
2523 }
2524 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2525 {
2526   \bool_if:nF {#1}
2527   {
2528     \__kernel_backend_literal_postscript:n
2529     {
2530       /setdistillerparams ~ where
2531       { pop << /CompressStreams ~ false >> setdistillerparams }
2532       if
2533     }
2534   }
2535 }

```

(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
2536 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2537 {
2538   \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2539 }
2540 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2541 {
2542   \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2543 }

```

(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:
2544 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2545 \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.

```

\__pdf_backend_bdc:nn
\__pdf_backend_emc:
2546 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2547 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2548 \cs_new_protected:Npn \__pdf_backend_emc:
2549 { \__pdf_backend_pdfmark:n { /EMC } }

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

2550 </dvips>

```

6.3 LuaTeX and pdfTeX backend

2551 <*luatex | pdftex>

6.3.1 Annotations

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

```

2552 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2553 {
2554   <*luatex>
2555     \tex_pdfextension:D annot ~
2556   </luatex>
2557   <*pdftex>
2558     \tex_pdfannot:D
2559   </pdftex>
2560     width ~ \dim_eval:n {#1} ~
2561     height ~ \dim_eval:n {#2} ~
2562     depth ~ \dim_eval:n {#3} ~
2563     {#4}
2564 }

```

(End definition for __pdf_backend_annotation:nnnn.)

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

```

2565 \cs_new:Npx \__pdf_backend_annotation_last:
2566 {
2567   \exp_not:N \int_value:w
2568   <*luatex>
2569     \exp_not:N \tex_pdffeedback:D lastannot ~
2570   </luatex>
2571   <*pdftex>
2572     \exp_not:N \tex_pdflastannot:D
2573   </pdftex>
2574   \c_space_tl 0 ~ R
2575 }

```

(End definition for __pdf_backend_annotation_last:.)

__pdf_backend_link_begin_goto:nnw Links are all created using the same internals.

```

\__pdf_backend_link_begin_user:nnw
\__pdf_backend_link_begin:nnnw
2576 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2577 { \__pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
\__pdf_backend_link_end:
2578 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2

```

```

2579 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2580 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2581 {
2582   <*luatex>
2583   \tex_pdfextension:D startlink ~
2584   </luatex>
2585   <*pdftex>
2586   \tex_pdfstartlink:D
2587   </pdftex>
2588   attr {#1}
2589   #2 {#3}
2590 }
2591 \cs_new_protected:Npn \_pdf_backend_link_end:
2592 {
2593   <*luatex>
2594   \tex_pdfextension:D endlink \scan_stop:
2595   </luatex>
2596   <*pdftex>
2597   \tex_pdfendlink:D
2598   </pdftex>
2599 }

```

(End definition for _pdf_backend_link_begin_goto:nnw and others.)

_pdf_backend_link_last: Formatted for direct use.

```

2600 \cs_new:Npx \_pdf_backend_link_last:
2601 {
2602   \exp_not:N \int_value:w
2603   <*luatex>
2604   \exp_not:N \tex_pdffeedback:D lastlink ~
2605   </luatex>
2606   <*pdftex>
2607   \exp_not:N \tex_pdflastlink:D
2608   </pdftex>
2609   \c_space_tl 0 ~ R
2610 }

```

(End definition for _pdf_backend_link_last:.)

_pdf_backend_link_margin:n A simple task: pass the data to the primitive.

```

2611 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2612 {
2613   <*luatex>
2614   \tex_pdfvariable:D linkmargin
2615   </luatex>
2616   <*pdftex>
2617   \tex_pdflinkmargin:D
2618   </pdftex>
2619   \dim_eval:n {#1} \scan_stop:
2620 }

```

(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.

```

2621 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2622 {
2623   \*luatex
2624     \tex_pdfextension:D dest ~
2625   \*pdfTeX
2626     \tex_pdfdest:D
2627   \*pdfTeX
2628     name {#1}
2629     \str_case:nnF {#2}
2630     {
2631       { xyz } { xyz }
2632       { fit } { fit }
2633       { fitb } { fitb }
2634       { fitbh } { fitbh }
2635       { fitbv } { fitbv }
2636       { fith } { fith }
2637       { fitv } { fitv }
2638       { fitr } { fitr }
2639     }
2640     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2641     \scan_stop:
2642   }
2643 }
2644 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2645 {
2646   \*luatex
2647     \tex_pdfextension:D dest ~
2648   \*pdfTeX
2649     \tex_pdfdest:D
2650     name {#1}
2651     fitr ~
2652     width \dim_eval:n {#2} ~
2653     height \dim_eval:n {#3} ~
2654     depth \dim_eval:n {#4} \scan_stop:
2655   }
2656 }
2657

```

(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`

```

2658 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2659 {
2660   \*luatex
2661     \tex_pdfextension:D catalog
2662   \*pdfTeX
2663     \tex_pdfcatalog:D
2664   \*pdfTeX
2665

```

```

2666     { / #1 ~ #2 }
2667   }
2668   \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2669   {
2670     <*luatex>
2671       \tex_pdfextension:D info
2672     </luatex>
2673     <*pdftex>
2674       \tex_pdfinfo:D
2675     </pdftex>
2676     { / #1 ~ #2 }
2677   }

```

(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.

```

2678 \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

```

2679 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2680 {
2681   <*luatex>
2682     \tex_pdfextension:D obj ~
2683   </luatex>
2684   <*pdftex>
2685     \tex_pdfobj:D
2686   </pdftex>
2687     reserveobjnum ~
2688     \int_const:cn
2689     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2690   <*luatex>
2691     { \tex_pdffeedback:D lastobj }
2692   </luatex>
2693   <*pdftex>
2694     { \tex_pdflastobj:D }
2695   </pdftex>
2696     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2697   }
2698   \cs_new:Npn \__pdf_backend_object_ref:n #1
2699   { \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

```

2700 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2701 {
2702   <*luatex>
2703     \tex_immediate:D \tex_pdfextension:D obj ~
2704   </luatex>
2705   <*pdftex>
2706     \tex_immediate:D \tex_pdfobj:D

```

```

2707 </pdfTeX>
2708     useobjnum ~
2709     \int_use:c
2710     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2711     \str_case_e:nn
2712     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2713     {
2714       { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2715       { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2716       { fstream }
2717       {
2718         stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2719         file ~ { \__pdf_exp_not_ii:nn #2 }
2720       }
2721     { stream }
2722     {
2723       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2724       { \__pdf_exp_not_ii:nn #2 }
2725     }
2726   }
2727 }
2728 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2729 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2730 \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
2731 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2732 {
2733   <*luatex>
2734   \tex_immediate:D \tex_pdfextension:D obj ~
2735   </luatex>
2736   <*pdfTeX>
2737   \tex_immediate:D \tex_pdfobj:D
2738   </pdfTeX>
2739   \str_case:nn
2740   {#1}
2741   {
2742     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2743     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2744     { fstream }
2745     {
2746       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2747       file ~ { \__pdf_exp_not_ii:nn #2 }
2748     }
2749     { stream }
2750     {
2751       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2752       { \__pdf_exp_not_ii:nn #2 }
2753     }
2754   }
2755 }
2756 \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.

```

2757 \cs_new:Npx \_pdf_backend_object_last:
2758 {
2759   \exp_not:N \int_value:w
2760   <*luatex>
2761   \exp_not:N \tex_pdffeedback:D lastobj ~
2762   </luatex>
2763   <*pdftex>
2764   \exp_not:N \tex_pdflastobj:D
2765   </pdftex>
2766   \c_space_tl 0 ~ R
2767 }

```

(End definition for _pdf_backend_object_last:.)

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

```

2768 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2769 {
2770   \exp_not:N \int_value:w
2771   <*luatex>
2772   \exp_not:N \tex_pdffeedback:D pageref
2773   </luatex>
2774   <*pdftex>
2775   \exp_not:N \tex_pdfpageref:D
2776   </pdftex>
2777   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2778 }

```

(End definition for _pdf_backend_pageobject_ref:n.)

6.3.4 Structure

_pdf_backend_compresslevel:n Simply pass data to the engine.

```

2779 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2780 {
2781   \tex_global:D
2782   <*luatex>
2783   \tex_pdfvariable:D compresslevel
2784   </luatex>
2785   <*pdftex>
2786   \tex_pdfcompresslevel:D
2787   </pdftex>
2788   \int_value:w \int_eval:n {#1} \scan_stop:
2789 }
2790 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2791 {
2792   \bool_if:nTF {#1}
2793   { \_pdf_backend_objcompresslevel:n { 2 } }
2794   { \_pdf_backend_objcompresslevel:n { 0 } }
2795 }
2796 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2797 {

```



```

2798     \tex_global:D
2799   \*luatex
2800     \tex_pdfvariable:D objcompresslevel
2801   \*luatex
2802   \*pdftex
2803     \tex_pdfobjcompresslevel:D
2804   \*pdftex
2805     #1 \scan_stop:
2806   }

(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

```

2807 \cs_new_protected:Npx \_pdf_backend_version_major_gset:n #1
2808 {
2809   \*luatex
2810     \int_compare:nNnT \tex luatexversion:D > { 106 }
2811     {
2812       \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2813       \exp_not:N \int_eval:n {#1} \scan_stop:
2814     }
2815   \*luatex
2816   \*pdftex
2817     \cs_if_exist:NT \tex_pdfmajorversion:D
2818     {
2819       \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2820       \exp_not:N \int_eval:n {#1} \scan_stop:
2821     }
2822   \*pdftex
2823 }
2824 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2825 {
2826   \tex_global:D
2827   \*luatex
2828     \tex_pdfvariable:D minorversion
2829   \*luatex
2830   \*pdftex
2831     \tex_pdfminorversion:D
2832   \*pdftex
2833     \int_eval:n {#1} \scan_stop:
2834 }

```

(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:

```

2835 \cs_new:Npx \_pdf_backend_version_major:
2836 {
2837   \*luatex
2838     \int_compare:nNnTF \tex luatexversion:D > { 106 }
2839     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2840     { 1 }
2841   \*luatex
2842   \*pdftex
2843     \cs_if_exist:NTF \tex_pdfmajorversion:D

```

```

2844         { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2845         { 1 }
2846     </pdfTeX>
2847 }
2848 \cs_new:Npn \__pdf_backend_version_minor:
2849 {
2850     \tex_the:D
2851     <*luatex>
2852     \tex_pdfvariable:D minorversion
2853     </luatex>
2854     <*pdfTeX>
2855     \tex_pdfminorversion:D
2856     </pdfTeX>
2857 }

```

(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>.

```

2858 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2859 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2860 \cs_new_protected:Npn \__pdf_backend_emc:
2861 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2862 </luatex | pdfTeX>

```

6.4 dvipdfmx backend

```

2863 <*dvipdfmx | xetex>

```

__pdf_backend:n A generic function for the backend PDF specials: used where we can.

```

\__pdf_backend:x
2864 \cs_new_protected:Npx \__pdf_backend:n #1
2865 { \__kernel_backend_literal:n { pdf: #1 } }
2866 \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
2867 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2868 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2869 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2870 { \__pdf_backend:n { docinfo << /#1 ~ #2 >> } }

```

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

6.4.2 Objects

\g__pdf_backend_object_int
\g__pdf_backend_object_prop

For tracking objects to allow finalisation.

```
2871 \int_new:N \g__pdf_backend_object_int
2872 \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

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

```
2873 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2874 {
2875   \int_gincr:N \g__pdf_backend_object_int
2876   \int_const:cn
2877   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2878   { \g__pdf_backend_object_int }
2879   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2880 }
2881 \cs_new:Npn \__pdf_backend_object_ref:n #1
2882 { @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:nnn
__pdf_backend_object_write_array:nn
__pdf_backend_object_write_dict:nn
__pdf_backend_object_write_fstream:nn
__pdf_backend_object_write_stream:nn
__pdf_backend_object_write_stream:nnnn

This is where we choose the actual type.

```
2883 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2884 {
2885   \exp_args:Nx \__pdf_backend_object_write:nnn
2886   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2887 }
2888 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2889 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2890 {
2891   \use:c { __pdf_backend_object_write_ #1 :nn }
2892   { \__pdf_backend_object_ref:n {#2} } {#3}
2893 }
2894 \cs_new_protected:Npn \__pdf_backend_object_write_array: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_dict:nn #1#2
2900 {
2901   \__pdf_backend:x
2902   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2903 }
2904 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2905 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2906 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2907 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2908 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2909 {
2910   \__pdf_backend:x
2911   {
2912     #1 stream ~ #2 ~
2913     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2914   }
```

```
2915 }
```

(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
2916 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2917 {
2918   \int_gincr:N \g__pdf_backend_object_int
2919   \exp_args:Nnx \use:c { \_pdf_backend_object_write_ #1 :nn }
2920   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2921   {#2}
2922 }
2923 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }
```

(End definition for `_pdf_backend_object_now:nn`.)

`_pdf_backend_object_last:`

```
2924 \cs_new:Npn \_pdf_backend_object_last:
2925 { @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_YTeX.

```
2926 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2927 { @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.

```
2928 \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.

```
2929 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2930 {
2931   \int_gincr:N \g__pdf_backend_object_int
2932   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2933   \_pdf_backend:x
2934   {
2935     ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
2936     width ~ \dim_eval:n {#1} ~
2937     height ~ \dim_eval:n {#2} ~
2938     depth ~ \dim_eval:n {#3} ~
2939     << /Type /Annot #4 >>
2940   }
2941 }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

`_pdf_backend_annotation_last:`

```
2942 \cs_new:Npn \_pdf_backend_annotation_last:
2943 { @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.

2944 \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

2945 \cs_new_protected:Npn _pdf_backend_link_begin_goto:nnw #1#2

2946 { _pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D (#2) >> } }

_pdf_backend_link_begin:n

2947 \cs_new_protected:Npn _pdf_backend_link_begin_user:nnw #1#2

_pdf_backend_link_end:

2948 { _pdf_backend_link_begin:n {#1#2} }

2949 \cs_new_protected:Npx _pdf_backend_link_begin:n #1

2950 {

2951 \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }

2952 {

2953 \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int

2954 }

2955 _pdf_backend:x

2956 {

2957 bann ~

2958 \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }

2959 {

2960 @pdf.lnk

2961 \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int

2962 \c_space_tl

2963 }

2964 <<

2965 /Type /Annot

2966 #1

2967 >>

2968 }

2969 }

2970 \cs_new_protected:Npn _pdf_backend_link_end:

2971 { _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.

2972 \cs_new:Npx _pdf_backend_link_last:

2973 {

2974 \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }

2975 {

2976 @pdf.lnk

2977 \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int

2978 }

2979 }

(End definition for _pdf_backend_link_last:.)

_pdf_backend_link_margin:n Pass to dvipdfmx.

2980 \cs_new_protected:Npn _pdf_backend_link_margin:n #1

2981 { _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 \TeX by using the backend data for `@xpos` and `@ypos`. `/FitR` without rule spec doesn't work, so it falls back to `/Fit` here.

```

2982 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2983 {
2984   \_pdf_backend:x
2985   {
2986     dest ~ ( \exp_not:n {#1} )
2987     [
2988       @thispage
2989       \str_case:nnF {#2}
2990       {
2991         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2992         { fit } { /Fit }
2993         { fitb } { /FitB }
2994         { fitbh } { /FitBH }
2995         { fitbv } { /FitBV ~ @xpos }
2996         { fith } { /FitH ~ @ypos }
2997         { fitv } { /FitV ~ @xpos }
2998         { fitr } { /Fit }
2999       }
3000       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
3001     ]
3002   }
3003 }
3004 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
3005 {
3006   \exp_args:Ne \_pdf_backend_destination_aux:nnnn
3007   { \dim_eval:n {#2} } {#1} {#3} {#4}
3008 }
3009 \cs_new_protected:Npn \_pdf_backend_destination_aux:nnnn #1#2#3#4
3010 {
3011   \vbox_to_zero:n
3012   {
3013     \_kernel_kern:n {#4}
3014     \hbox:n
3015     {
3016       \_pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
3017       \_pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
3018     }
3019     \tex_vss:D
3020   }
3021   \_kernel_kern:n {#1}
3022   \vbox_to_zero:n
3023   {
3024     \_kernel_kern:n { -#3 }
3025     \hbox:n
3026     {
3027       \_pdf_backend:n
3028       {
3029         dest ~ (#2)
3030         [
3031           @thispage

```

```

3032             /FitR ~
3033             @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
3034             @xpos ~ @ypos
3035         ]
3036     }
3037 }
3038 \tex_vss:D
3039 }
3040 \__kernel_kern:n { -#1 }
3041 }

```

(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.

```

3042 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
3043 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
3044 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
3045 {
3046   \bool_if:nF {#1}
3047   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
3048 }

```

(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.

```

3049 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
3050 {
3051   \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
3052   \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
3053 }
3054 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
3055 {
3056   \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
3057   \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
3058 }

```

(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.

```

3059 \cs_new:Npn \__pdf_backend_version_major: { 1 }
3060 \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>.

```

3061 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
3062 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
3063 \cs_new_protected:Npn \__pdf_backend_emc:
3064 { \__kernel_backend_literal_page:n { EMC } }

```

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

3065 </dvipdfmx | xetex>

6.5 dvisvgm backend

3066 <*dvisvgm>

6.5.1 Catalogue entries

No-op.

_pdf_backend_catalog_gput:nn
_pdf_backend_info_gput:nn

3067 \cs_new_protected:Npn _pdf_backend_catalog_gput:nn #1#2 { }

3068 \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.

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

3069 \cs_new_protected:Npn _pdf_backend_object_new:nn #1#2 { }

3070 \cs_new:Npn _pdf_backend_object_ref:n #1 { }

3071 \cs_new_protected:Npn _pdf_backend_object_write:nn #1#2 { }

3072 \cs_new_protected:Npn _pdf_backend_object_write:nx #1#2 { }

3073 \cs_new_protected:Npn _pdf_backend_object_now:nn #1#2 { }

3074 \cs_new_protected:Npn _pdf_backend_object_now:nx #1#2 { }

3075 \cs_new:Npn _pdf_backend_object_last: { }

3076 \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.

_pdf_backend_compresslevel:n
_pdf_backend_compress_objects:n

3077 \cs_new_protected:Npn _pdf_backend_compresslevel:n #1 { }

3078 \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!

_pdf_backend_version_major_gset:n
_pdf_backend_version_minor_gset:n

3079 \cs_new_protected:Npn _pdf_backend_version_major_gset:n #1 { }

3080 \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!

_pdf_backend_version_major:
_pdf_backend_version_minor:

3081 \cs_new:Npn _pdf_backend_version_major: { -1 }

3082 \cs_new:Npn _pdf_backend_version_minor: { -1 }

(End definition for _pdf_backend_version_major: and _pdf_backend_version_minor:.)

More no-ops.

_pdf_backend_bdc:nn
 _pdf_backend_emc:

3083 \cs_new_protected:Npn _pdf_backend_bdc:nn #1#2 { }

3084 \cs_new_protected:Npn _pdf_backend_emc: { }

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

3085 </dvisvgm>

3086 </package>

7 l3backend-opacity Implementation

```
3087 <*package>
3088 <@@=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.

```
3089 <*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.

```
3090 \cs_new_protected:Npn \__opacity_backend_select:n #1
3091 {
3092   \exp_args:Nx \__opacity_backend_select_aux:n
3093   { \fp_eval:n { min(max(0,#1),1) } }
3094 }
3095 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3096 {
3097   \__opacity_backend:nnn {#1} { fill } { ca }
3098   \__opacity_backend:nnn {#1} { stroke } { CA }
3099 }
3100 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3101 {
3102   \__opacity_backend:xnn
3103   { \fp_eval:n { min(max(0,#1),1) } }
3104   { fill }
3105   { ca }
3106 }
3107 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3108 {
3109   \__opacity_backend:xnn
3110   { \fp_eval:n { min(max(0,#1),1) } }
3111   { stroke }
3112   { CA }
3113 }
3114 \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3115 {
3116   \__kernel_backend_postscript:n
3117   {
3118     product ~ (Ghostscript) ~ search
3119     {
3120       pop ~ pop ~ pop ~
3121       #1 ~ .set #2 constantalpha
3122     }
3123     {
3124       pop ~
3125       mark ~
3126       /#3 ~ #1
```

```

3127         /SetTransparency ~
3128         pdfmark
3129     }
3130     ifelse
3131 }
3132 }
3133 \cs_generate_variant:Nn \_opacity_backend:nnn { x }

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

3134 </dvips>
3135 <*dvipdfmx | luatex | pdftex | xetex>

```

\c__opacity_backend_stack_int Set up a stack.

```

3136 \bool_lazy_and:nnT
3137 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3138 { \pdfmanagement_if_active_p:}
3139 {
3140     \__kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3141     { page ~ direct } { /opacity 1 ~ gs }
3142     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3143     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3144 }

(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
3145 \tl_new:N \l__opacity_backend_fill_tl
3146 \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:
3147 \cs_new_protected:Npn \_opacity_backend_select:n #1
3148 {
3149     \exp_args:Nx \_opacity_backend_select_aux:n
3150     { \fp_eval:n { min(max(0,#1),1) } }
3151 }
3152 \cs_new_protected:Npn \_opacity_backend_select_aux:n #1
3153 {
3154     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3155     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3156     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3157     { opacity #1 }
3158     { << /ca ~ #1 /CA ~ #1 >> }
3159     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3160     { /opacity #1 ~ gs }
3161     \group_insert_after:N \_opacity_backend_reset:
3162 }
3163 \bool_lazy_and:nnF
3164 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3165 { \pdfmanagement_if_active_p:}
3166 {
3167     \cs_gset_protected:Npn \_opacity_backend_select_aux:n #1 { }
3168 }

```

```

3169 \cs_new_protected:Npn \__opacity_backend_reset:
3170 { \__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.

```

3171 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3172 {
3173   \__opacity_backend_fill_stroke:xx
3174   { \fp_eval:n { min(max(0,#1),1) } }
3175   \l__opacity_backend_stroke_tl
3176 }
3177 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3178 {
3179   \__opacity_backend_fill_stroke:xx
3180   \l__opacity_backend_fill_tl
3181   { \fp_eval:n { min(max(0,#1),1) } }
3182 }
3183 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3184 {
3185   \str_if_eq:nnTF {#1} {#2}
3186   { \__opacity_backend_select_aux:n {#1} }
3187   {
3188     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3189     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3190     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3191     { opacity.fill #1 }
3192     { << /ca ~ #1 >> }
3193     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3194     { opacity.stroke #1 }
3195     { << /CA ~ #2 >> }
3196     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3197     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3198     \group_insert_after:N \__opacity_backend_reset:
3199   }
3200 }
3201 \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.)

```

3202 </dviPDFmx | luatex | pdftex | xetex>
3203 <*dviPDFmx | xdvipdfmx>

```

```
\__opacity_backend_select:n
```

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

```

3204 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
3205 {
3206   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1
3207   {
3208     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3209     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3210     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3211     { opacity #1 }

```

```

3212         { << /ca ~ #1 /CA ~ #1 >> }
3213         \__kernel_backend_literal_pdf:n { /opacity #1 ~ gs }
3214     }
3215     \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3216     {
3217         \str_if_eq:nnTF {#1} {#2}
3218         { \__opacity_backend_select_aux:n {#1} }
3219         {
3220             \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3221             \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3222             \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3223             { opacity.fill #1 }
3224             { << /ca ~ #1 >> }
3225             \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3226             { opacity.stroke #1 }
3227             { << /CA ~ #2 >> }
3228             \__kernel_backend_literal_pdf:n
3229             { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3230         }
3231     }
3232 }

```

(End definition for __opacity_backend_select:n.)

```

3233 </dvipdfmx | xdvipdfmx>

```

```

3234 <*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
3235 \cs_new_protected:Npn \__opacity_backend_select:n #1
3236 { \__opacity_backend:nn {#1} { } }
3237 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3238 { \__opacity_backend:nn {#1} { fill- } }
3239 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3240 { \__opacity_backend:nn { {#1} } { stroke- } }
3241 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3242 { \__kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

```

(End definition for __opacity_backend_select:n and others.)

```

3243 </dvisvgm>

```

```

3244 </package>

```

8 l3backend-header Implementation

```

3245 <*dvips & header>

```

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

```

3246 /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.

```

3247 TeXDict begin
3248 /TeXcolorseparation { setcolor } def
3249 end

```

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

pdf.globaldict A small global dictionary for backend use.

```

3250 true setglobal
3251 /pdf.globaldict 4 dict def
3252 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.

```

3253 /pdf.cvs { 65534 string cvs } def
3254 /pdf.dvi.pt { 72.27 mul Resolution div } def
3255 /pdf.pt.dvi { 72.27 div Resolution mul } def
3256 /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 pdf.linkdp.pad pdf.linkht.pad Settings which are defined up-front in SDict.

```

3257 /pdf.linkmargin { 1 pdf.pt.dvi } def
3258 /pdf.linkdp.pad { 0 } def
3259 /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 pdf.save.linkll pdf.save.linkur pdf.llx pdf.lly pdf.urx pdf.ury 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.

```

3260 /pdf.rect
3261 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
3262 /pdf.save.ll
3263 {
3264     currentpoint
3265     /pdf.lly exch def
3266     /pdf.llx exch def
3267 }
3268 def
3269 /pdf.save.ur
3270 {
3271     currentpoint
3272     /pdf.ury exch def
3273     /pdf.urx exch def
3274 }
3275 def
3276 /pdf.save.linkll
3277 {

```

```

3278     currentpoint
3279     pdf.linkmargin add
3280     pdf.linkdp.pad add
3281     /pdf.lly exch def
3282     pdf.linkmargin sub
3283     /pdf.llx exch def
3284   }
3285   def
3286 /pdf.save.linkur
3287   {
3288     currentpoint
3289     pdf.linkmargin sub
3290     pdf.linkht.pad sub
3291     /pdf.ury exch def
3292     pdf.linkmargin add
3293     /pdf.urx exch def
3294   }
3295   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 3296 /pdf.dest.anchor
pdf.dev.y 3297   {
pdf.tmpa 3298     currentpoint exch
pdf.tmpb 3299     pdf.dvi.pt 72 add
pdf.tmpc 3300     /pdf.dest.x exch def
pdf.tmpd 3301     pdf.dvi.pt
3302     vsize 72 sub exch sub
3303     /pdf.dest.y exch def
3304   }
3305   def
3306 /pdf.dest.point
3307   { pdf.dest.x pdf.dest.y } def
3308 /pdf.dest2device
3309   {
3310     /pdf.dest.y exch def
3311     /pdf.dest.x exch def
3312     matrix currentmatrix
3313     matrix defaultmatrix
3314     matrix invertmatrix
3315     matrix concatmatrix
3316     cvx exec
3317     /pdf.dev.y exch def
3318     /pdf.dev.x exch def
3319     /pdf.tmpd exch def
3320     /pdf.tmpc exch def
3321     /pdf.tmpb exch def
3322     /pdf.tmpa exch def
3323     pdf.dest.x pdf.tmpa mul

```

```

3324     pdf.dest.y pdf.tmpc mul add
3325     pdf.dev.x add
3326 pdf.dest.x pdf.tmpb mul
3327     pdf.dest.y pdf.tmpd mul add
3328     pdf.dev.y add
3329 }
3330 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 3331 /pdf.bordertracking false def
pdf.brokenlink.skip 3332 /pdf.bordertracking.begin
pdf.brokenlink.dict 3333 {
3334     SDict /pdf.bordertracking true put
3335     SDict /pdf.leftboundary undef
3336     SDict /pdf.rightboundary undef
3337     /a where
3338     {
3339         /a
3340         {
3341             currentpoint pop
3342             SDict /pdf.rightboundary known dup
3343             {
3344                 SDict /pdf.rightboundary get 2 index lt
3345                 { not }
3346                 if
3347             }
3348             if
3349             { pop }
3350             { SDict exch /pdf.rightboundary exch put }
3351             ifelse
3352             moveto
3353             currentpoint pop
3354             SDict /pdf.leftboundary known dup
3355             {
3356                 SDict /pdf.leftboundary get 2 index gt
3357                 { not }
3358                 if
3359             }
3360             if
3361             { pop }
3362             { SDict exch /pdf.leftboundary exch put }
3363             ifelse
3364         }
3365         put
3366     }
3367     if
3368 }
3369 def
3370 /pdf.bordertracking.end

```

```

3371 {
3372   /a where { /a { moveto } put } if
3373   /x where { /x { 0 exch rmoveto } put } if
3374   SDict /pdf.leftboundary known
3375     { pdf.outerbox 0 pdf.leftboundary put }
3376   if
3377   SDict /pdf.rightboundary known
3378     { pdf.outerbox 2 pdf.rightboundary put }
3379   if
3380   SDict /pdf.bordertracking false put
3381 }
3382 def
3383 /pdf.bordertracking.endpage
3384 {
3385   pdf.bordertracking
3386   {
3387     pdf.bordertracking.end
3388     true setglobal
3389     pdf.globaldict
3390       /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3391     pdf.globaldict
3392       /pdf.brokenlink.skip pdf.baselineskip put
3393     pdf.globaldict
3394       /pdf.brokenlink.dict
3395         pdf.link.dict pdf.cvs put
3396     false setglobal
3397     mark pdf.link.dict cvx exec /Rect
3398     [
3399       pdf.llx
3400       pdf.lly
3401       pdf.outerbox 2 get pdf.linkmargin add
3402       currentpoint exch pop
3403       pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3404     ]
3405     /ANN pdf.pdfmark
3406   }
3407   if
3408 }
3409 def
3410 /pdf.bordertracking.continue
3411 {
3412   /pdf.link.dict pdf.globaldict
3413   /pdf.brokenlink.dict get def
3414   /pdf.outerbox pdf.globaldict
3415   /pdf.brokenlink.rect get def
3416   /pdf.baselineskip pdf.globaldict
3417   /pdf.brokenlink.skip get def
3418   pdf.globaldict dup dup
3419   /pdf.brokenlink.dict undef
3420   /pdf.brokenlink.skip undef
3421   /pdf.brokenlink.rect undef
3422   currentpoint
3423   /pdf.originy exch def
3424   /pdf.originx exch def

```



```

3425     /a where
3426     {
3427         /a
3428         {
3429             moveto
3430             SDict
3431             begin
3432                 currentpoint pdf.originy ne exch
3433                 pdf.originx ne or
3434                 {
3435                     pdf.save.linkll
3436                     /pdf.lly
3437                     pdf.lly pdf.outerbox 1 get sub def
3438                     pdf.bordertracking.begin
3439                 }
3440                 if
3441                 end
3442             }
3443             put
3444         }
3445     if
3446     /x where
3447     {
3448         /x
3449         {
3450             0 exch rmoveto
3451             SDict
3452             begin
3453                 currentpoint
3454                 pdf.originy ne exch pdf.originx ne or
3455                 {
3456                     pdf.save.linkll
3457                     /pdf.lly
3458                     pdf.lly pdf.outerbox 1 get sub def
3459                     pdf.bordertracking.begin
3460                 }
3461                 if
3462                 end
3463             }
3464             put
3465         }
3466     if
3467 }
3468 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.

```

3469 /pdf.breaklink
3470 {

```

```

3471 pop
3472 counttomark 2 mod 0 eq
3473 {
3474   counttomark /pdf.count exch def
3475   {
3476     pdf.count 0 eq { exit } if
3477     counttomark 2 roll
3478     1 index /Rect eq
3479     {
3480       dup 4 array copy
3481       dup dup
3482       1 get
3483       pdf.outerbox pdf.rect.ht
3484       pdf.linkmargin 2 mul add sub
3485       3 exch put
3486       dup
3487       pdf.outerbox 2 get
3488       pdf.linkmargin add
3489       2 exch put
3490       dup dup
3491       3 get
3492       pdf.outerbox pdf.rect.ht
3493       pdf.linkmargin 2 mul add add
3494       1 exch put
3495       /pdf.currentrect exch def
3496       pdf.breaklink.write
3497       {
3498         pdf.currentrect
3499         dup
3500         pdf.outerbox 0 get
3501         pdf.linkmargin sub
3502         0 exch put
3503         dup
3504         pdf.outerbox 2 get
3505         pdf.linkmargin add
3506         2 exch put
3507         dup dup
3508         1 get
3509         pdf.baselineskip add
3510         1 exch put
3511         dup dup
3512         3 get
3513         pdf.baselineskip add
3514         3 exch put
3515         /pdf.currentrect exch def
3516         pdf.breaklink.write
3517       }
3518       1 index 3 get
3519       pdf.linkmargin 2 mul add
3520       pdf.outerbox pdf.rect.ht add
3521       2 index 1 get sub
3522       pdf.baselineskip div round cvi 1 sub
3523       exch
3524       repeat

```

```

3525         pdf.currentrect
3526         dup
3527             pdf.outerbox 0 get
3528             pdf.linkmargin sub
3529             0 exch put
3530         dup dup
3531             1 get
3532             pdf.baselineskip add
3533             1 exch put
3534         dup dup
3535             3 get
3536             pdf.baselineskip add
3537             3 exch put
3538         dup 2 index 2 get 2 exch put
3539         /pdf.currentrect exch def
3540         pdf.breaklink.write
3541         SDict /pdf.pdfmark.good false put
3542         exit
3543     }
3544     { pdf.count 2 sub /pdf.count exch def }
3545     ifelse
3546 }
3547 loop
3548 }
3549 if
3550 /ANN
3551 }
3552 def
3553 /pdf.breaklink.write
3554 {
3555     counttomark 1 sub
3556     index /_objdef eq
3557     {
3558         counttomark -2 roll
3559         dup wcheck
3560         {
3561             readonly
3562             counttomark 2 roll
3563         }
3564         { pop pop }
3565     } ifelse
3566 }
3567 if
3568 counttomark 1 add copy
3569 pop pdf.currentrect
3570 /ANN pdfmark
3571 }
3572 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.

```

3573 /pdf.pdfmark
3574 {
3575   SDict /pdf.pdfmark.good true put
3576   dup /ANN eq
3577   {
3578     pdf.pdfmark.store
3579     pdf.pdfmark.dict
3580     begin
3581       Subtype /Link eq
3582       currentdict /Rect known and
3583       SDict /pdf.outerbox known and
3584       SDict /pdf.baselineskip known and
3585       {
3586         Rect 3 get
3587         pdf.linkmargin 2 mul add
3588         pdf.outerbox pdf.rect.ht add
3589         Rect 1 get sub
3590         pdf.baselineskip div round cvi 0 gt
3591         { pdf.breaklink }
3592         if
3593       }
3594       if
3595     end
3596     SDict /pdf.outerbox undef
3597     SDict /pdf.baselineskip undef
3598     currentdict /pdf.pdfmark.dict undef
3599   }
3600   if
3601   pdf.pdfmark.good
3602   { pdfmark }
3603   { cleartomark }
3604   ifelse
3605 }
3606 def
3607 /pdf.pdfmark.store
3608 {
3609   /pdf.pdfmark.dict 65534 dict def
3610   counttomark 1 add copy
3611   pop
3612   {
3613     dup mark eq
3614     {
3615       pop
3616       exit
3617     }
3618     {
3619       pdf.pdfmark.dict
3620       begin def end
3621     }
3622     ifelse
3623   }
3624   loop

```

```
3625 }  
3626 def  
  
(End definition for pdf.pdfmark and others. These functions are documented on page ??.)  
3627 </dvips & header>
```

Index

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

<code>_</code>	154	<code>_box_backend_rotate:Nn</code>	235, 283, 340, <u>419</u>
A			
<code>\AtBeginDvi</code>	57	<code>_box_backend_rotate_aux:Nn</code>	235, 283, <u>340</u>
B			
bool commands:			
<code>\bool_gset_false:N</code>	1280, 1299, 1322, 1344, 1360, 1461, 1700, 1736, 2304, 2350	<code>_box_backend_scale:Nnn</code>	252, <u>311</u> , <u>355</u> , <u>432</u>
<code>\bool_gset_true:N</code>	1278, 1347, 1459, 1715, 2297, 2303	<code>\l_box_backend_sin_fp</code>	<u>283</u>
<code>\bool_if:NTF</code>	67, 704, 1290, 1294, 1310, 1313, 1317, 1328, 1335, 1339, 1351, 1355, 1472, 1477, 1482, 1674, 1719, 1832, 1867, 1977, 2019, 2292, 2307, 2312, 2317	C	
<code>\bool_if:nTF</code>	2526, 2792, 3046	char commands:	
<code>\bool_lazy_and:nnTF</code>	920, 994, 3136, 3163	<code>\char_set_catcode_space:n</code>	154
<code>\bool_lazy_or:nnTF</code>	1859, 2012	clist commands:	
<code>\bool_new:N</code>	1281, 1348, 1462, 1716, 2277, 2278	<code>\clist_map_function:nN</code>	1368, 1492, 1743
<code>\bool_set_false:N</code>	1842, 1944, 2037, 2101	color internal commands:	
box commands:		<code>_color_backend:nnn</code>	<u>1166</u>
<code>\box_dp:N</code>	224, 226, 274, 276, 331, 333, 380, 382, 384, 386, 2329, 2362, 2363, 2388	<code>_color_backend_cmyk:w</code>	<u>1167</u>
<code>\box_ht:N</code>	226, 276, 333, 384, 386, 1879, 2074, 2334, 2373, 2374, 2390	<code>\g_color_backend_colorant_prop</code>	670, 689, 692, 712, 937
<code>\box_if_empty:NTF</code>	2423	<code>_color_backend_devicen_colorants:n</code>	671, 873, 992
<code>\box_move_down:nn</code>	2251, 2329	<code>_color_backend_devicen_colorants:w</code>	671
<code>\box_move_up:nn</code>	2253, 2334	<code>_color_backend_devicen_init:nnn</code>	860, <u>962</u>
<code>\box_new:N</code>	2136, 2241, 2242	<code>_color_backend_devicen_init:w</code>	<u>962</u>
<code>\box_set_dp:Nn</code>	1799	<code>_color_backend_fill:n</code>	1073, 1100, <u>1130</u> , <u>1148</u> , <u>1155</u>
<code>\box_set_ht:Nn</code>	1798	<code>_color_backend_fill_cmyk:n</code>	1073, <u>1107</u> , <u>1130</u> , <u>1155</u>
<code>\box_set_wd:Nn</code>	288, 1797	<code>_color_backend_fill_devicen:nn</code>	1099, 1121, <u>1147</u> , <u>1217</u>
<code>\box_use:N</code>	231, 249, 263, 279, 306, 320, 336, 352, 364, 415, 429, 448, 1412, 1607, 1800, 2282	<code>_color_backend_fill_gray:n</code>	1073, <u>1107</u> , <u>1130</u> , <u>1155</u>
<code>\box_wd:N</code>	225, 233, 275, 281, 332, 338, 381, 383, 1878, 2073	<code>_color_backend_fill_rgb:n</code>	1073, <u>1107</u> , <u>1130</u> , <u>1155</u>
box internal commands:		<code>_color_backend_fill_separation:nn</code>	1099, <u>1107</u> , <u>1147</u> , <u>1217</u>
<code>_box_backend_clip:N</code>	213, 268, <u>325</u> , <u>369</u>	<code>\l_color_backend_fill_tl</code>	637, 647, 1081, 1096
<code>\l_box_backend_cos_fp</code>	<u>283</u>	<code>_color_backend_iccbased_device:nnn</code>	<u>1039</u>
		<code>_color_backend_iccbased_init:nnn</code>	879, <u>1013</u>
		<code>\c_color_backend_main_stack_int</code>	<u>516</u>
		<code>_color_backend_pickup:N</code> ..	<u>456</u> , <u>479</u>
		<code>_color_backend_pickup:w</code> ..	<u>14</u> , <u>456</u> , <u>479</u>

_color_backend_reset:	619, 639, 656, 1084, 1097, 1107, 1139, 1164
_color_backend_rgb:w	1190
_color_backend_select:n . .	619, 699
_color_backend_select:nn .	639, 905
_color_backend_select_cmyk:n . .	619, 639, 656
_color_backend_select_devicen:nn	698, 882, 904, 1065
_color_backend_select_gray:n . .	619, 639, 656
_color_backend_select_iccbased:nn	701, 886, 904
_color_backend_select_rgb:n . . .	619, 639, 656
_color_backend_select_separation:nn	698, 882, 904, 1065
_color_backend_separation_-init:n	702
_color_backend_separation_-init:nn	908
_color_backend_separation_-init:nnn	702
_color_backend_separation_-init:nnnn	702
_color_backend_separation_-init:nnnnn	702, 884, 908
_color_backend_separation_-init:nw	702
_color_backend_separation_-init:w	702
_color_backend_separation_-init_/DeviceCMYK:nnn	702
_color_backend_separation_-init_/DeviceGray:nnn	702
_color_backend_separation_-init_/DeviceRGB:nnn	702
_color_backend_separation_-init_aux:nnnnn	702
_color_backend_separation_-init_CIELAB:nnn	702, 884, 908
_color_backend_separation_-init_CIELAB:nnnnn	885
_color_backend_separation_-init_count:n	702
_color_backend_separation_-init_count:w	702
_color_backend_separation_-init_Device:Nn	702
\g_color_backend_stack_int . . .	516
\l_color_backend_stack_int	513, 541, 547, 649, 653, 1082, 1095
_color_backend_stroke:n	1073, 1102, 1107
_color_backend_stroke_cmyk:n . .	1073, 1130, 1166
_color_backend_stroke_cmyk:w	1166
_color_backend_stroke_devicen:nn	1099, 1125, 1147, 1217
_color_backend_stroke_gray:n . .	1073, 1130, 1166
_color_backend_stroke_gray_-aux:n	1166
_color_backend_stroke_rgb:n . . .	1073, 1130, 1166
_color_backend_stroke_rgb:w	1166
_color_backend_stroke_separation:nn	1099, 1107, 1147, 1217
\l_color_backend_stroke_tl	637, 648, 1083, 1094
\g_color_model_int	709, 718, 866, 894, 928, 1002, 1034
\c_color_model_range_CIELAB_tl . .	821, 856, 951, 958
color.sc	619, 3246
cs commands:	
\cs_generate_variant:Nn . .	49, 63, 66, 99, 138, 143, 170, 201, 207, 569, 606, 723, 1227, 1422, 1616, 1991, 2048, 2064, 2140, 2177, 2236, 2728, 2756, 2866, 2888, 2923, 3133, 3201
\cs_gset:Npx . .	2538, 2542, 3051, 3056
\cs_gset_eq:NN	663, 664, 1068, 1114, 1115, 1121, 1123, 1125
\cs_gset_protected:Npn	551, 658, 665, 1067, 1109, 1116, 1118, 1120, 3167, 3206, 3215
\cs_if_exist:NTF	27, 50, 457, 480, 539, 1029, 1052, 2419, 2817, 2843
\cs_if_exist_p:N . .	921, 995, 3137, 3164
\cs_if_exist_use:NTF	38, 736
\cs_new:Npn	686, 745, 747, 749, 751, 758, 764, 766, 772, 789, 796, 798, 1007, 1373, 1497, 1747, 2077, 2086, 2130, 2155, 2237, 2239, 2272, 2444, 2544, 2545, 2698, 2729, 2730, 2848, 2881, 2924, 2926, 2942, 3059, 3060, 3070, 3075, 3076, 3081, 3082
\cs_new:Npx	671, 2565, 2600, 2757, 2768, 2835, 2972
\cs_new_eq:NN	46, 57, 59, 700, 883, 906, 907, 1103, 1104, 1151, 1152, 1219, 1220, 1226, 1421, 1427, 1428, 1615, 1617, 1618, 1624, 1809, 1838, 1889, 1890, 1932, 1940, 1962, 2033, 2090, 2097, 2129, 2282
\cs_new_protected:Npn	47, 54, 61, 64, 72, 78, 83, 85, 89, 100,

<u>_draw_backend_cm_decompose_-</u> auxiii:nnnnN 1553	<u>_draw_backend_scope_begin:</u> 1238 , 1424 , 1427 , 1617
<u>_draw_backend_curveto:nnnnnn</u> 1242 , 1429 , 1625	<u>_draw_backend_scope_end:</u> 1238 , 1426 , 1427 , 1617
<u>_draw_backend_dash:n</u> 1362 , 1486 , 1738	<u>_draw_backend_stroke:</u> 1282 , 1463 , 1670
<u>_draw_backend_dash_aux:nn</u> .. 1738	<u>\g_draw_draw_clip_bool</u> .. 1282 , 1670
<u>_draw_backend_dash_pattern:nn</u> 1362 , 1486 , 1738	<u>\g_draw_draw_eor_bool</u> 1277 , 1294 , 1310 , 1317 , 1328 , 1339 , 1355 , 1458 , 1472 , 1477 , 1482
<u>_draw_backend_discardpath:</u> 1282 , 1463 , 1670	<u>\g_draw_draw_path_int</u> 1670
<u>_draw_backend_end:</u> 1228 , 1423 , 1619	<u>\g_draw_path_tl</u> 1735
<u>_draw_backend_evenodd_rule:</u> 1277 , 1458 , 1666	<div style="text-align: center;">E</div>
<u>_draw_backend_fill:</u> 1282 , 1463 , 1670	<u>\errmsgmessage</u> 38
<u>_draw_backend_fillstroke:</u> 1282 , 1463 , 1670	<u>\evensidemargin</u> 2327
<u>_draw_backend_join_bevel:</u> 1362 , 1486 , 1738	exp commands:
<u>_draw_backend_join_miter:</u> 1362 , 1486 , 1738	<u>\exp_after:wN</u> 159 , 465 , 2084
<u>_draw_backend_join_round:</u> 1362 , 1486 , 1738	<u>\exp_args:Ne</u> 760 , 2481 , 3006
<u>_draw_backend_lineto:nn</u> 1242 , 1429 , 1625	<u>\exp_args:Nf</u> 1367 , 1491 , 2246
<u>_draw_backend_linewidth:n</u> 1362 , 1486 , 1738	<u>\exp_args:NNf</u> 236 , 284 , 341
<u>_draw_backend_literal:n</u> 1226 , 1231 , 1235 , 1239 , 1241 , 1244 , 1252 , 1260 , 1269 , 1283 , 1286 , 1287 , 1288 , 1289 , 1292 , 1298 , 1308 , 1315 , 1321 , 1326 , 1331 , 1332 , 1333 , 1334 , 1337 , 1343 , 1353 , 1359 , 1364 , 1377 , 1381 , 1383 , 1385 , 1387 , 1389 , 1391 , 1393 , 1396 , 1401 , 1402 , 1403 , 1404 , 1405 , 1406 , 1410 , 1411 , 1413 , 1414 , 1415 , 1416 , 1417 , 1421 , 1431 , 1436 , 1441 , 1451 , 1464 , 1466 , 1468 , 1471 , 1476 , 1481 , 1485 , 1488 , 1501 , 1505 , 1507 , 1509 , 1511 , 1513 , 1515 , 1517 , 1615 , 1677 , 1696 , 1722	<u>\exp_args:Nnx</u> 2233 , 2919
<u>_draw_backend_miterlimit:n</u> 1362 , 1486 , 1738	<u>\exp_args:NV</u> 461
<u>_draw_backend_moveto:nn</u> 1242 , 1429 , 1625	<u>\exp_args:Nx</u> . 706 , 918 , 1902 , 1923 , 2190 , 2205 , 2323 , 2885 , 3092 , 3149
<u>_draw_backend_nonzero_rule:</u> 1277 , 1458 , 1666	<u>\exp_last_unbraced:Nx</u> 470 , 484
<u>_draw_backend_path:n</u> 1670	<u>\exp_not:N</u> 522 , 523 , 531 , 533 , 673 , 679 , 680 , 681 , 708 , 709 , 712 , 713 , 718 , 2567 , 2569 , 2572 , 2602 , 2604 , 2607 , 2759 , 2761 , 2764 , 2770 , 2772 , 2775 , 2812 , 2813 , 2819 , 2820 , 2839 , 2844 , 2953 , 2961 , 2977
<u>\g_draw_backend_path_int</u> 1685 , 1702	<u>\exp_not:n</u> 48 , 97 , 108 , 136 , 1021 , 2181 , 2186 , 2475 , 2714 , 2715 , 2729 , 2730 , 2742 , 2743 , 2897 , 2902 , 2913 , 2986
<u>\g_draw_backend_path_tl</u> 1625 , 1681 , 1697 , 1699 , 1726	<u>\ExplBackendFileDate</u> 1
<u>_draw_backend_rectangle:nnnn</u> 1242 , 1429 , 1625	<div style="text-align: center;">F</div>
	file commands:
	<u>\file_compare_timestamp:nNnTF</u> . 1911
	<u>\file_parse_full_name:nNNN</u> 1898 , 1921
	<u>\fmtversion</u> 52
	fp commands:
	<u>\fp_compare:nNnTF</u> 243 , 290 , 296 , 348 , 1534 , 1547 , 1592
	<u>\fp_eval:n</u> . 236 , 245 , 258 , 259 , 284 , 301 , 316 , 318 , 341 , 350 , 361 , 362 , 426 , 441 , 442 , 1174 , 1175 , 1176 , 1184 , 1197 , 1198 , 1199 , 1536 , 1

\fp_new:N	309, 310	__graphics_backend_getbb_-auxvi:nNnn	2068, 2070
\fp_set:Nn	289, 292	__graphics_backend_getbb_eps:n	1809, 1891, 1932, 2090
\fp_use:N	295, 299, 304	__graphics_backend_getbb_eps:nm	1891
\fp_zero:N	291	__graphics_backend_getbb_eps:nn	1902, 1909
\c_zero_fp	243, 290, 296, 348, 1534, 1547	__graphics_backend_getbb_jpg:n	1824, 1932, 2027, 2091
G			
graphics commands:			
\graphics_bb_restore:nTF	1853, 2067	__graphics_backend_getbb_-pagebox:w	2027, 2084
\graphics_bb_save:n	1882, 2075	__graphics_backend_getbb_pdf:n	1824, 1917, 1932, 2027, 2098
\l_graphics_decodearray_tl	1830, 1831, 1841, 1861, 1865, 1866, 1943, 1975, 1976, 2014, 2017, 2018, 2036, 2100	__graphics_backend_getbb_png:n	1824, 1932, 2027, 2091
\graphics_extract_bb:n	1938, 1945, 2095, 2102	__graphics_backend_include:nn	2104
\l_graphics_interpolate_bool	1832, 1842, 1860, 1867, 1944, 1977, 2013, 2019, 2037, 2101	__graphics_backend_include_-auxi:nn	1949
\l_graphics_llx_dim	1815, 1954, 2006, 2113	__graphics_backend_include_-auxii:nnn	1949
\l_graphics_lly_dim	1816, 1955, 2007, 2114	__graphics_backend_include_-auxiii:nnn	1949
\l_graphics_name_tl	1916	__graphics_backend_include_-bitmap_quote:w	2078, 2119
\l_graphics_page_int	1826, 1846, 1847, 1871, 1872, 1936, 1973, 1974, 2000, 2001, 2029, 2042, 2043, 2082, 2083, 2093	__graphics_backend_include_-eps:n	1810, 1891, 1949, 2104
\l_graphics_pagebox_tl	54, 1827, 1845, 1873, 1874, 1937, 1971, 1972, 2002, 2004, 2030, 2051, 2052, 2084, 2094	__graphics_backend_include_-jpg:n	1884, 1949, 2119
\graphics_read_bb:n	1809, 1932, 2090	__graphics_backend_include_-pdf:n	1884, 1923, 1949, 2078, 2104
\l_graphics_urx_dim	1817, 1878, 1956, 2008, 2073, 2115	__graphics_backend_include_-pdf_quote:w	2081, 2086
\l_graphics_ury_dim	1818, 1879, 1957, 2009, 2074, 2116, 2124, 2125	__graphics_backend_include_-png:n	1884, 1949, 2119
graphics internal commands:			
\l__graphics_backend_dir_str	1891	\l__graphics_backend_name_str	1891
\l__graphics_backend_ext_str	1891	\l__graphics_graphics_attr_tl	1823, 1828, 1835, 1843, 1853, 1880, 1882, 1887
__graphics_backend_getbb_auxi:n	1824	\l__graphics_internal_box	1876, 1878, 1879, 2072, 2073, 2074
__graphics_backend_getbb_-auxi:nN	2027	\g__graphics_track_int	1948, 1994, 1995
__graphics_backend_getbb_-auxii:n	1824	group commands:	
__graphics_backend_getbb_-auxii:nnN	2027	\group_begin:	151, 179, 198
__graphics_backend_getbb_-auxiii:nNnn	2027	\group_end:	164, 187
__graphics_backend_getbb_-auxiv:nnNnn	2027	\group_insert_after:N	631, 650, 661, 1084, 1097, 1112, 1139, 1164, 3161, 3198
__graphics_backend_getbb_-auxv:nNnn	2027	H	
hbox commands:			
\hbox:n	2252, 2255, 2330, 2336, 2489, 2496, 3014, 3025		

`\hbox_overlap_right:n` 231,
 263, 279, 320, 336, 364, 448, 1412, 1607
`\hbox_set:Nn` ... 1876, 2072, 2322, 2354
`\hbox_set:Nw` 2305
`\hbox_set_end:` 2320
`\hbox_unpack:N` 2441
 hook commands:
`\hook_gput_code:nnn` 55

I

int commands:
`\int_compare:nNnTF` 516,
 558, 656, 1065, 1107, 1846, 1871,
 1973, 2000, 2042, 2082, 2413, 2514,
 2810, 2838, 2951, 2958, 2974, 3204
`\int_const:Nn` 157, 163, 523,
 549, 584, 1880, 1995, 2150, 2688, 2876
`\int_eval:n`
 . 565, 575, 604, 615, 756, 765, 778,
 780, 784, 797, 2538, 2542, 2788,
 2813, 2820, 2833, 3043, 3051, 3056
`\int_gincr:N` 205, 371,
 522, 1676, 1721, 1994, 2149, 2218,
 2262, 2339, 2875, 2918, 2931, 2953
`\int_gset:Nn` 180, 199, 2402
`\int_gset_eq:NN` 188, 2263, 2340, 2932
`\int_if_exist:NTF` 1984
`\int_if_odd:nTF` 2325
`\int_new:N` 171, 172,
 418, 513, 519, 1702, 1948, 2145,
 2243, 2274, 2276, 2871, 2928, 2944
`\int_set:Nn` 541
`\int_set_eq:NN` ... 176, 195, 547, 2414
`\int_step_function:nnnN` 782
`\int_use:N` 373, 404, 531, 542,
 709, 718, 866, 894, 928, 1002, 1034,
 1679, 1685, 1692, 1724, 1732, 1847,
 1872, 1887, 1974, 1987, 1999, 2001,
 2083, 2156, 2221, 2234, 2238, 2266,
 2273, 2344, 2445, 2699, 2709, 2882,
 2920, 2925, 2935, 2943, 2961, 2977
`\int_value:w`
 2567, 2602, 2759, 2770, 2788
`\int_zero:N` ... 1826, 1936, 2029, 2093

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, 706

`\g__kernel_backend_header_bool` ..
 67, 704
`__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, 634, 660, 666, 726, 862,
 1111, 1117, 1119, 1138, 1163, 1230,
 1236, 1531, 1538, 1544, 1604, 1609,
 1812, 1951, 1986, 1996, 2110, 2121,
 2865, 2981, 3043, 3047, 3052, 3057
`__kernel_backend_literal_page:n`
 100, 144, 2859, 2861, 3062, 3064
`__kernel_backend_literal_pdf:n` .
 .. 89, 141, 271, 328, 1421, 3213, 3228
`__kernel_backend_literal_-
 postscript:n`
 61, 75, 76, 80, 217, 218, 220,
 221, 229, 241, 256, 1226, 2516, 2528
`__kernel_backend_literal_svg:n` .
 168, 175, 186, 194, 204,
 372, 374, 391, 888, 1615, 1790, 1801
`__kernel_backend_matrix:n`
 128, 293, 314, 1521
`__kernel_backend_postscript:n` ..
 64,
 629, 1142, 1144, 1146, 1150, 2139,
 2195, 2210, 2252, 2258, 2298, 2330,
 2337, 2341, 2355, 2383, 2427, 2434,
 2440, 2448, 2455, 2489, 2496, 3116
`__kernel_backend_scope:n` 173, 401,
 406, 1204, 1622, 1667, 1669, 1689,
 1729, 1751, 1763, 1765, 1767, 1769,
 1771, 1773, 1775, 1777, 1780, 3242
`__kernel_backend_scope_begin:` ..
 83, 110, 146, 173,
 215, 239, 254, 270, 287, 313, 327,
 344, 357, 1427, 1599, 1617, 1621, 1788
`__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, 1428, 1611, 1618, 1624, 1802
`\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, 1676, 1679, 1692, 1721, 1724, 1732
`__kernel_color_backend_stack_-
 init:Nnn` 516, 582, 3140
`__kernel_color_backend_stack_-
 pop:n` 558, 596, 653, 3170

_kernel_color_backend_stack_- push:nn 558 , 596 , 649 , 1082 , 1095 , 3159 , 3196	\pdf_object_write:nn . 945 , 1018 , 1044
_kernel_dependency_version_- check:Nn 1	pdf internal commands:
_kernel_dependency_version_- check:nn 27 , 29	_pdf_backend:n 2864 , 2868 , 2870 , 2896 , 2901 , 2910 , 2933 , 2955 , 2971 , 2984 , 3016 , 3017 , 3027
_kernel_kern:n 2257 , 2259 , 2488 , 2492 , 2495 , 2499 , 3013 , 3021 , 3024 , 3040	_pdf_backend_annotation:nnnn 2244 , 2552 , 2929
\c_kernel_sys_dvipdfmx_version_- int 151 , 516 , 558 , 656 , 1065 , 1107 , 2951 , 2958 , 2974 , 3204	_pdf_backend_annotation_int 2243 , 2263 , 2273 , 2928 , 2932 , 2943
	_pdf_backend_annotation_last: 2272 , 2565 , 2942
M	_pdf_backend_bdc:nn 2546 , 2858 , 3061 , 3083
\MessageBreak 40	_pdf_backend_catalog_gput:nn 2141 , 2658 , 2867 , 3067
mode commands:	_pdf_backend_compress_objects:n 2512 , 2779 , 3042 , 3077
\mode_if_horizontal:TF ... 2404 , 2411	_pdf_backend_compresslevel:n 2512 , 2779 , 3042 , 3077
\mode_if_math:TF 2302	\l_pdf_backend_content_box 2241 , 2305 , 2329 , 2332 , 2334 , 2363 , 2374
O	_pdf_backend_destination:nn 2453 , 2621 , 2982
\oddsidemargin 2326	_pdf_backend_destination:nnnn 2453 , 2621 , 2982
opacity internal commands:	_pdf_backend_destination_- aux:nnnn 2453 , 2982
_opacity_backend:nn 3235	_pdf_backend_emc: 2546 , 2858 , 3061 , 3083
_opacity_backend:nnn 3090	_pdf_backend_info_gput:nn 2141 , 2658 , 2867 , 3067
_opacity_backend_fill:n 3090 , 3171 , 3235	_pdf_backend_link:nw 2283
_opacity_backend_fill_stroke:nn 3173 , 3179 , 3183 , 3201 , 3215	_pdf_backend_link_aux:nw ... 2283
\l_opacity_backend_fill_tl 3145 , 3154 , 3180 , 3188 , 3208 , 3220	_pdf_backend_link_begin:n .. 2945
_opacity_backend_fillstroke:nn 3171	_pdf_backend_link_begin:nnnw 2576
_opacity_backend_reset: 3147 , 3198	_pdf_backend_link_begin:nw 2285 , 2289 , 2290
_opacity_backend_select:n 3090 , 3147 , 3204 , 3235	_pdf_backend_link_begin_aux:nw 2293 , 2295
_opacity_backend_select_aux:n 3090 , 3147 , 3186 , 3206 , 3218	_pdf_backend_link_begin_- goto:nnw 2283 , 2576 , 2945
\c_opacity_backend_stack_int 3136 , 3159 , 3170 , 3196	_pdf_backend_link_begin_- user:nnw 2283 , 2576 , 2945
_opacity_backend_stroke:n 3090 , 3171 , 3235	\g_pdf_backend_link_bool 2278 , 2292 , 2297 , 2312 , 2350
\l_opacity_backend_stroke_tl 3145 , 3155 , 3175 , 3189 , 3209 , 3221	\g_pdf_backend_link_dict_tl 2275 , 2300 , 2345
P	_pdf_backend_link_end: 2283 , 2576 , 2945
pdf commands:	_pdf_backend_link_end_aux: . 2283
\pdf_object_if_exist:nTF 942 , 1015 , 1041	
\pdf_object_new:nn ... 944 , 1017 , 1043	
\pdf_object_ref:n 957 , 1028 , 1051	
\pdf_object_ref_last: 929 , 936 , 938 , 991 , 1003 , 1035 , 1059	
\pdf_object_unnamed_write:nn 910 , 935 , 964 , 986 , 1027 , 1050	

\g_pdf_backend_link_int	__pdf_backend_pdfmark:n
..... 2274, 2340,	2138, 2142, 2144, 2159, 2180, 2185,
2344, 2445, 2944, 2953, 2961, 2977	2219, 2264, 2456, 2500, 2547, 2549
__pdf_backend_link_last:	__pdf_backend_version_major: ...
..... 2444, 2600, 2972 2538,
__pdf_backend_link_margin:n ...	2544, 2835, 3051, 3052, 3059, 3081
..... 2446, 2611, 2980	__pdf_backend_version_major_-
\g_pdf_backend_link_math_bool ..	gset:n 2536, 2807, 3049, 3079
..... 2277, 2303, 2304, 2307, 2317	__pdf_backend_version_minor: ...
__pdf_backend_link_minima: .. 2283 2542,
__pdf_backend_link_outerbox:n 2283	2544, 2835, 3056, 3057, 3059, 3081
\g_pdf_backend_link_sf_int	__pdf_backend_version_minor_-
..... 2276, 2402, 2413, 2414	gset:n 2536, 2807, 3049, 3079
__pdf_backend_link_sf_restore: 2283	\l_pdf_breaklink_pdfmark_tl ...
__pdf_backend_link_sf_save: . 2283 2279, 2347, 2439
\l_pdf_backend_model_box . 2242,	__pdf_breaklink_postscript:n ...
2322, 2354, 2362, 2373, 2388, 2390 2281, 2331, 2333, 2440
__pdf_backend_objcompresslevel:n	__pdf_breaklink_usebox:N
..... 2779 2282, 2332, 2441
\g_pdf_backend_object_int	__pdf_exp_not_i:nn . 2700, 2746, 2751
..... 2145, 2149, 2152,	__pdf_exp_not_ii:nn 2700, 2747, 2752
2218, 2221, 2234, 2238, 2262, 2263,	\l_pdf_internal_box
2266, 2339, 2340, 2871, 2875, 2878,	pdf.baselineskip
2918, 2920, 2925, 2931, 2932, 2935	pdf.baselineskip
__pdf_backend_object_last:	pdf.bordertracking
..... 2237, 2757, 2924, 3069	pdf.bordertracking.begin
__pdf_backend_object_new:nn ...	pdf.bordertracking.continue
..... 2147, 2679, 2873, 3069	pdf.bordertracking.end
__pdf_backend_object_now:nn ...	pdf.bordertracking.endpage
..... 2216, 2731, 2916, 3069	pdf.bordertracking
\g_pdf_backend_object_prop	pdf.breaklink
..... 2145, 2153, 2164, 2174,	pdf.breaklink.write
2678, 2696, 2712, 2871, 2879, 2886	pdf.brokenlink.dict
__pdf_backend_object_ref:n 2147,	pdf.brokenlink.rect
2161, 2175, 2679, 2873, 2892, 3069	pdf.brokenlink.skip
__pdf_backend_object_write:nn ..	pdf.count
..... 2157, 2700, 2883, 3069	pdf.currentrect
__pdf_backend_object_write:nnn 2883	pdf.cvs
__pdf_backend_object_write_-	pdf.dest.anchor
array:nn 2157, 2883	pdf.dest.point
__pdf_backend_object_write_-	pdf.dest.x
dict:nn 2157, 2883	pdf.dest.y
__pdf_backend_object_write_-	pdf.dest2device
fstream:nn 2157, 2883	pdf.dev.x
__pdf_backend_object_write_-	pdf.dev.y
fstream:nnn 2191, 2193	pdf.dvi.pt
__pdf_backend_object_write_-	pdf.globaldict
stream:nn 2157, 2883	pdf.leftboundary
__pdf_backend_object_write_-	pdf.link.dict
stream:nnn 2157	pdf.linkdp.pad
__pdf_backend_object_write_-	pdf.linkht.pad
stream:nnnn 2883	pdf.linkmargin
__pdf_backend_pageobject_ref:n .	pdf.llx
..... 2239, 2768, 2926, 3069	pdf.lly
	pdf.lly
	pdf.originx
	pdf.originy

