

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}{2021-07-12}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2021-07-12}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2021-07-12}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2021-07-12}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2021-07-12}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2021-07-12}{ }
25   {L3 backend support: XeTeX}
26 </xetex>
```

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

```
27 \cs_if_exist:NTF \_kernel_dependency_version_check: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 `"`).

```

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:` 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:` 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 $\langle *luatex | pdftex \rangle$

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   \langle *luatex \rangle
92   \tex\_pdfextension:D literal
93   \langle /luatex \rangle
94   \langle *pdftex \rangle
95   \tex\_pdfliteral:D
96   \langle /pdftex \rangle
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   \langle *luatex \rangle
103   \tex\_pdfextension:D literal ~
104   \langle /luatex \rangle
105   \langle *pdftex \rangle
106   \tex\_pdfliteral:D
107   \langle /pdftex \rangle
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   \langle *luatex \rangle
113   \tex\_pdfextension:D save \scan\_stop:
114   \langle /luatex \rangle
115   \langle *pdftex \rangle
116   \tex\_pdfsave:D
117   \langle /pdftex \rangle
118 }
119 \cs_new_protected:Npn \_kernel\_backend\_scope\_end:
120 {
121   \langle *luatex \rangle
122   \tex\_pdfextension:D restore \scan\_stop:
123   \langle /luatex \rangle
124   \langle *pdftex \rangle
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 `xdvidfpmx (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.

```

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` 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__box_clip_path_int
372   \__kernel_backend_literal_svg:x
373   { < clipPath~id = " l3cp \int_use:N \g__box_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__box_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__box_clip_path_int
(End definition for \__box_backend_clip:N and \g__box_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/XYTeX` 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 `<*dvipdfmx|xetex>`

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

```

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

```

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

```

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

```

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


```

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

```

Simple enough but needs a version check.

```

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

```

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

```

580 </dvipdfmx | xetex>

```

3.2.3 LuaTeX and pdfTeX

```

581 <*luatex | pdftex>

```

```

\__kernel_color_backend_stack_init:Nnn

```

```

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

```

(End definition for __kernel_color_backend_stack_init:Nnn.)

```

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

```

```

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

```

```

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

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

617 </luatex | pdftex>

```

3.3 General color

3.3.1 dvips-style

```

618 <*dvips | dvisvgm>

\__color_backend_select_cmyk:n Push the data to the stack. In the case of dvips also saves the drawing color in raw
\__color_backend_select_gray:n PostScript.
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc

619 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
620 { \__color_backend_select:n { cmyk ~ #1 } }
621 \cs_new_protected:Npn \__color_backend_select_gray:n #1
622 { \__color_backend_select:n { gray ~ #1 } }
623 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
624 { \__color_backend_select:n { rgb ~ #1 } }
625 \cs_new_protected:Npn \__color_backend_select:n #1
626 {
627   \__kernel_backend_literal:n { color~push~ #1 }
628 <*dvips>
629   \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
630 </dvips>
631   \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 \langle *dvips \rangle

__color_backend_select_separation:nn
__color_backend_select_devicen:nn

670 \cs_new_protected:Npn __color_backend_select_separation:nn #1#2
671 { __color_backend_select:n { separation ~ #1 ~ #2 } }
672 \cs_new_eq:NN __color_backend_select_devicen:nn __color_backend_select_separation:nn

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

__color_backend_separation_init:nnnnn
__color_backend_separation_init:nxxnn
__color_backend_separation_init_aux:nnnnn
__color_backend_separation_init_DeviceCMYK:nnn
__color_backend_separation_init_DeviceGray:nnn
__color_backend_separation_init_DeviceRGB:nnn
__color_backend_separation_init_Device:Nn
__color_backend_separation_init:nnn
__color_backend_separation_init_count:n
__color_backend_separation_init_count:w
__color_backend_separation_init:nnnn
__color_backend_separation_init:w
__color_backend_separation_init:n
__color_backend_separation_init:nw
__color_backend_separation_init_CIELAB:nnn

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

673 \cs_new_protected:Npx __color_backend_separation_init:nnnnn #1#2#3#4#5
674 {
675 \bool_if:NT \g__kernel_backend_header_bool
676 {
677 __kernel_backend_first_shipout:n
678 {
679 \exp_not:N __color_backend_separation_init_aux:nnnnn
680 {#1} {#2} {#3} {#4} {#5}
681 }
682 }
683 }
684 \cs_generate_variant:Nn __color_backend_separation_init:nnnnn { nxx }
685 \cs_new_protected:Npn __color_backend_separation_init_aux:nnnnn #1#2#3#4#5
686 {
687 __kernel_backend_literal:e
688 {
689 !
690 TeXDict ~ begin ~
691 /color \int_use:N \g__color_model_int
692 {
693 [~
694 /Separation ~ (\str_convert_pdfname:n {#1}) ~
695 [~ #2 ~] ~
696 {
697 \cs_if_exist_use:cF { __color_backend_separation_init_ #2 :nnn }
698 { __color_backend_separation_init:nnn }
699 {#3} {#4} {#5}
700 }
701] ~ setcolorspace
702 } ~ def ~
703 end
704 }
705 }
706 \cs_new:cpn { __color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
707 { __color_backend_separation_init_Device:Nn 4 {#3} }
708 \cs_new:cpn { __color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
709 { __color_backend_separation_init_Device:Nn 1 {#3} }
710 \cs_new:cpn { __color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3

```

711 { \_color_backend_separation_init_Device:Nn 2 {#3} }
712 \cs_new:Npn \_color_backend_separation_init_Device:Nn #1#2
713 {
714   #2 ~
715   \prg_replicate:nn {#1}
716   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
717   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
718 }

```

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.

```

719 \cs_new:Npn \_color_backend_separation_init:nnn #1#2#3
720 {
721   \exp_args:Ne \_color_backend_separation_init:nnnn
722   { \_color_backend_separation_init_count:n {#2} }
723   {#1} {#2} {#3}
724 }
725 \cs_new:Npn \_color_backend_separation_init_count:n #1
726 { \int_eval:n { 0 \_color_backend_separation_init_count:w #1 ~ \s_color_stop } }
727 \cs_new:Npn \_color_backend_separation_init_count:w #1 ~ #2 \s_color_stop
728 {
729   +1
730   \tl_if_blank:nF {#2}
731   { \_color_backend_separation_init_count:w #2 \s_color_stop }
732 }

```

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.

```

733 \cs_new:Npn \_color_backend_separation_init:nnnn #1#2#3#4
734 {
735   \_color_backend_separation_init:w #3 ~ \s_color_stop #4 ~ \s_color_stop
736   \prg_replicate:nn {#1}
737   {
738     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
739     \int_eval:n { 3 * #1 } ~ index ~ mul ~
740     2 ~ index ~ add ~
741     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
742   }
743   \int_step_function:nnnN {#1} { -1 } { 1 }
744   \_color_backend_separation_init:n
745   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
746   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
747   \tl_if_blank:nF {#2}

```

```

748     { \_color_backend_separation_init:nw {#1} #2 ~ \s_color_stop }
749   }
750   \cs_new:Npn \_color_backend_separation_init:w
751     #1 ~ #2 \s_color_stop #3 ~ #4 \s_color_stop
752   {
753     #1 ~ #3 ~ 0 ~
754     \tl_if_blank:nF {#2}
755     { \_color_backend_separation_init:w #2 \s_color_stop #4 \s_color_stop }
756   }
757   \cs_new:Npn \_color_backend_separation_init:n #1
758   { \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.

```

759   \cs_new:Npn \_color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s_color_stop
760   {
761     #2 ~ #3 ~
762     2 ~ index ~ 2 ~ index ~ lt ~
763     { ~ pop ~ excl ~ pop ~ } ~
764     { ~
765       2 ~ index ~ 1 ~ index ~ gt ~
766       { ~ excl ~ pop ~ excl ~ pop ~ } ~
767       { ~ pop ~ pop ~ } ~
768       ifelse ~
769     }
770     ifelse ~
771     #1 ~ 1 ~ roll ~
772     \tl_if_blank:nF {#4}
773     { \_color_backend_separation_init:nw {#1} #4 \s_color_stop }
774   }

```

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.

```

775   \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
776   {
777     \_color_backend_separation_init:nxxxnn
778     {#2}
779     {
780       /CIEBasedABC ~
781       << ~
782       /RangeABC ~ [ ~ \c_color_model_range_CIELAB_tl \c_space_tl ] ~
783       /DecodeABC ~
784       [ ~
785         { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
786         { ~ 500 ~ div ~ } ~ bind ~
787         { ~ 200 ~ div ~ } ~ bind ~
788       ] ~
789       /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
790       /DecodeLMN ~
791       [ ~
792         { ~
793           dup ~ 6 ~ 29 ~ div ~ ge ~
794           { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
795           { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~

```

```

796         ifelse ~
797         0.9505 ~ mul ~
798     } ~ bind ~
799     { ~
800         dup ~ 6 ~ 29 ~ div ~ ge ~
801         { ~ dup ~ dup ~ mul ~ mul ~ } ~
802         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
803         ifelse ~
804     } ~ bind ~
805     { ~
806         dup ~ 6 ~ 29 ~ div ~ ge ~
807         { ~ dup ~ dup ~ mul ~ mul ~ } ~
808         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
809         ifelse ~
810         1.0890 ~ mul ~
811     } ~ bind
812 ] ~
813 /WhitePoint ~
814 [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
815 >>
816 }
817 { \c__color_model_range_CIELAB_tl }
818 { 100 ~ 0 ~ 0 }
819 {#3}
820 }

```

(End definition for _color_backend_separation_init:nnnnn and others.)

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

```

821 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
822 {
823     \_kernel_backend_literal:e
824     {
825         !
826         TeXDict ~ begin ~
827         /color \int_use:N \g__color_model_int
828         {
829             [ ~
830                 /DeviceN ~
831                 [ ~ #1 ~ ] ~
832                 #2 ~
833                 { ~ #3 ~ } ~
834             ] ~ setcolorspace
835         } ~ def ~
836     end
837 }
838 }

```

(End definition for _color_backend_devicen_init:nnn.)

```

839 </dvips>
840 <*dvisvgm>

```

_color_backend_select_separation:nn No support at present.

```

\_color_backend_select_devicen:nn 841 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2 { }
842 \cs_new_protected:Npn \_color_backend_select_devicen:nn #1#2 { }

```

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

`_color_backend_separation_init:nnnnn`
`_color_backend_separation_init_CIELAB:nnn`

No support at present.

```
843 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }
844 \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`.)

```
845 \</divisvgm>
```

```
846 <*dvipdfmx | luatex | pdfTeX | xetex>
```

`_color_backend_select_separation:nn`
`_color_backend_select_devicen:nn`

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

```
847 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
848 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
849 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
```

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

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

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

```
850 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5
851 {
852   \pdf_object_unnamed_write:nx { dict }
853   {
854     /FunctionType ~ 2
855     /Domain ~ [0 ~ 1]
856     \tl_if_blank:nF {#3} { /Range ~ [#3] }
857     /C0 ~ [#4] ~
858     /C1 ~ [#5] /N ~ 1
859   }
860   \_color_backend_separation_init:n
861   {
862     /Separation ~
863     / \str_convert_pdfname:n {#1} ~ #2 ~
864     \pdf_object_ref_last:
865   }
866   \cs_if_exist:NT \pdfmanagement_add:nnn
867   {
868     \use:x
869     {
870       \pdfmanagement_add:nnn
871       { Page / Resources / ColorSpace }
872       { color \int_use:N \g_color_model_int }
873       { \pdf_object_ref_last: }
874     }
875   }
876 }
877 \cs_new_protected:Npn \_color_backend_separation_init:n #1
878 {
879   \pdf_object_unnamed_write:nx { array } {#1}
880 }
```


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

```

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

```

(End definition for __color_backend_separation_init:nnnnn, __color_backend_separation_init:n, and __color_backend_separation_init_CIELAB:nnn.)

__color_backend_devicen_init:nnn
 __color_backend_devicen_init:w
 __color_backend_devicen_init:n

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

```

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

```

```

927   \_color_backend_separation_init:n
928   {
929     /DeviceN ~
930     [ ~ #1 ~ ] ~
931     #2 ~
932     \pdf_object_ref_last:
933   }
934   \cs_if_exist:NT \pdfmanagement_add:nnn
935   {
936     \use:x
937     {
938       \pdfmanagement_add:nnn
939       { Page / Resources / ColorSpace }
940       { color \int_use:N \g__color_model_int }
941       { \pdf_object_ref_last: }
942     }
943   }
944 }
945 \cs_new:Npn \_color_backend_devicen_init:w #1 ~ #2 \s__color_stop
946 {
947   + 1
948   \tl_if_blank:nF {#2}
949   { \_color_backend_devicen_init:w #2 \s__color_stop }
950 }
951 \cs_new_eq:NN \_color_backend_devicen_init:n \_color_backend_separation_init:n

```

(End definition for _color_backend_devicen_init:nnn, _color_backend_devicen_init:w, and _color_backend_devicen_init:n.)

```

952 </dvipdfmx | luatex | pdftex | xetex>
953 <*dvipdfmx | xetex>

```

```

\_color_backend_select_separation:nn
\_color_backend_select_devicen: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.

```

954 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
955 {
956   \cs_gset_protected:Npn \_color_backend_select_separation:nn #1#2 { }
957   \cs_gset_eq:NN \_color_backend_select_devicen:nn
958   \_color_backend_select_separation:nn
959 }

```

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

```

960 </dvipdfmx | xetex>

```

3.5 Fill and stroke color

Here, dvipdfmx/X_gTeX 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.

```

961 <*dvipdfmx | luatex | pdftex | xetex>

```

`_color_backend_fill_cmyk:n` Drawing (fill/stroke) color is handled in `dvipdfmx`/`XYTeX` in the same way as `LuaTeX`/`pdfTeX`.
`_color_backend_fill_gray:n` We use the same approach as earlier, except the color stack is not involved so the generic
`_color_backend_fill_rgb:n` direct PDF operation is used. There is no worry about the nature of strokes: everything
`_color_backend_fill:n` is handled automatically.

```

962 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
963 { \_color_backend_fill:n { #1 ~ k } }
964 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
965 { \_color_backend_fill:n { #1 ~ g } }
966 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
967 { \_color_backend_fill:n { #1 ~ rg } }
968 \cs_new_protected:Npn \_color_backend_fill:n #1
969 {
970   \tl_set:Nn \l__color_backend_fill_tl {#1}
971   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
972   { #1 ~ \l__color_backend_stroke_tl }
973   \group_insert_after:N \_color_backend_reset:
974 }
975 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
976 { \_color_backend_stroke:n { #1 ~ K } }
977 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
978 { \_color_backend_stroke:n { #1 ~ G } }
979 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
980 { \_color_backend_stroke:n { #1 ~ RG } }
981 \cs_new_protected:Npn \_color_backend_stroke:n #1
982 {
983   \tl_set:Nn \l__color_backend_stroke_tl {#1}
984   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
985   { \l__color_backend_fill_tl \c_space_tl #1 }
986   \group_insert_after:N \_color_backend_reset:
987 }

```

(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
988 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
989 { \_color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
990 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
991 { \_color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
992 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
993 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

```

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

```

`_color_backend_fill_cmyk:n` Deal with older (x)dvipdfmx.

```

996 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
997 {
998   \cs_gset_protected:Npn \_color_backend_fill_cmyk:n #1
999   {
1000     \__kernel_backend_literal:n { pdf: bc ~ [#1] }
1001     \group_insert_after:N \_color_backend_reset:
1002   }

```

```

1003 \cs_gset_eq:NN \_color_backend_fill_gray:n \_color_backend_fill_cmyk:n
1004 \cs_gset_eq:NN \_color_backend_fill_rgb:n \_color_backend_fill_cmyk:n
1005 \cs_gset_protected:Npn \_color_backend_reset:
1006 { \_kernel_backend_literal:n { pdf: ec } }
1007 \cs_gset_protected:Npn \_color_backend_stroke:n #1
1008 { \_kernel_backend_literal:n { #1 } }
1009 \cs_gset_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1010 \cs_gset_eq:NN \_color_backend_fill_devicen:nn
1011 \_color_backend_fill_separation:nn
1012 \cs_gset_eq:NN \_color_backend_stroke_separation:nn
1013 \_color_backend_fill_separation:nn
1014 \cs_gset_eq:NN \_color_backend_stroke_devicen:nn
1015 \_color_backend_stroke_separation:nn
1016 }

```

(End definition for _color_backend_fill_cmyk:n and others.)

```

1017 </dvipdfmx | xetex>

```

```

1018 <*dvips>

```

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

```

\_color_backend_fill_gray:n
\_color_backend_fill_rgb:n
\_color_backend_fill:n
\_color_backend_stroke_cmyk:n
\_color_backend_stroke_gray:n
\_color_backend_stroke_rgb:n
1019 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1020 { \_color_backend_fill:n { cmyk ~ #1 } }
1021 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1022 { \_color_backend_fill:n { gray ~ #1 } }
1023 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1024 { \_color_backend_fill:n { rgb ~ #1 } }
1025 \cs_new_protected:Npn \_color_backend_fill:n #1
1026 {
1027   \_kernel_backend_literal:n { color~push~ #1 }
1028   \group_insert_after:N \_color_backend_reset:
1029 }
1030 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1031 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1032 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1033 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1034 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1035 { \_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
1036 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1037 { \_color_backend_fill:n { separation ~ #1 ~ #2 } }
1038 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1039 { \_kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1040 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1041 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

(End definition for _color_backend_fill_separation:nn and others.)

```

1042 </dvips>

```

```

1043 <*dvisvgm>

```

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

```

\_color_backend_fill_gray:n 1044 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
\_color_backend_fill_rgb:n 1045 { \_color_backend_fill:n { cmyk ~ #1 } }
\_color_backend_fill:n 1046 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1047 { \_color_backend_fill:n { gray ~ #1 } }
1048 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1049 { \_color_backend_fill:n { rgb ~ #1 } }
1050 \cs_new_protected:Npn \_color_backend_fill:n #1
1051 {
1052 \_kernel_backend_literal:n { color~push~ #1 }
1053 \group_insert_after:N \_color_backend_reset:
1054 }

```

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

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

```

\_color_backend_stroke_cmyk:w 1055 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
\_color_backend_stroke_gray:n 1056 { \_color_backend_cmyk:w #1 \s__color_stop }
\_color_backend_stroke_gray_aux:n 1057 \cs_new_protected:Npn \_color_backend_stroke_cmyk:w
\_color_backend_stroke_rgb:n 1058 #1 ~ #2 ~ #3 ~ #4 \s__color_stop
\_color_backend_stroke_rgb:w 1059 {
\_color_backend:nnn 1060 \use:x
1061 {
1062 \_color_backend:nnn
1063 { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1064 { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1065 { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }
1066 }
1067 }
1068 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1069 {
1070 \use:x
1071 {
1072 \_color_backend_stroke_gray_aux:n
1073 { \fp_eval:n { 100 * (#1) } }
1074 }
1075 }
1076 \cs_new_protected:Npn \_color_backend_stroke_gray_aux:n #1
1077 { \_color_backend:nnn {#1} {#1} {#1} }
1078 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1079 { \_color_backend_rgb:w #1 \s__color_stop }
1080 \cs_new_protected:Npn \_color_backend_stroke_rgb:w
1081 #1 ~ #2 ~ #3 \s__color_stop
1082 {
1083 \use:x
1084 {
1085 \_color_backend:nnn
1086 { \fp_eval:n { 100 * (#1) } }
1087 { \fp_eval:n { 100 * (#2) } }
1088 { \fp_eval:n { 100 * (#3) } }
1089 }
1090 }
1091 \cs_new_protected:Npx \_color_backend:nnn #1#2#3

```

```

1092 {
1093   \__kernel_backend_scope:n
1094   {
1095     stroke =
1096     "
1097       rgb
1098       (
1099         #1 \c_percent_str ,
1100         #2 \c_percent_str ,
1101         #3 \c_percent_str
1102       )
1103     "
1104   }
1105 }

```

(End definition for __color_backend_stroke_cmyk:n and others.)

```

\__color_backend_fill_separation:nn
\__color_backend_stroke_separation:nn
\__color_backend_fill_devicen:nn
\__color_backend_stroke_devicen:nn
1106 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1107 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2 { }
1108 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1109 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

1110 </dvisvgm>
1111 </package>

```

4 l3backend-draw Implementation

```

1112 <*package>
1113 <@@=draw>

```

4.1 dvips backend

```

1114 <*dvips>

```

```

\__draw_backend_literal:n
\__draw_backend_literal:x
1115 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_postscript:n
1116 \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.

```

1117 \cs_new_protected:Npn \__draw_backend_begin:
1118 {

```

```

1119     \__kernel_backend_literal:n { ps::[begin] }
1120     \__draw_backend_literal:n { @beginspecial }
1121   }
1122   \cs_new_protected:Npn \__draw_backend_end:
1123   {
1124     \__draw_backend_literal:n { @endspecial }
1125     \__kernel_backend_literal:n { ps::[end] }
1126   }

```

(End definition for __draw_backend_begin: and __draw_backend_end:.)

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

```

1127   \cs_new_protected:Npn \__draw_backend_scope_begin:
1128   { \__draw_backend_literal:n { save } }
1129   \cs_new_protected:Npn \__draw_backend_scope_end:
1130   { \__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.

```

1131   \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1132   {
1133     \__draw_backend_literal:x
1134     {
1135       \dim_to_decimal_in_bp:n {#1} ~
1136       \dim_to_decimal_in_bp:n {#2} ~ moveto
1137     }
1138   }
1139   \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1140   {
1141     \__draw_backend_literal:x
1142     {
1143       \dim_to_decimal_in_bp:n {#1} ~
1144       \dim_to_decimal_in_bp:n {#2} ~ lineto
1145     }
1146   }
1147   \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1148   {
1149     \__draw_backend_literal:x
1150     {
1151       \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1152       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1153       moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1154     }
1155   }
1156   \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1157   {
1158     \__draw_backend_literal:x
1159     {

```

```

1160         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1161         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1162         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1163         curveto
1164     }
1165 }

```

(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
1166 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1167 { \bool_gset_true:N \g__draw_draw_eor_bool }
1168 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1169 { \bool_gset_false:N \g__draw_draw_eor_bool }
1170 \bool_new:N \g__draw_draw_eor_bool

```

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

```

\__draw_backend_closepath: Unlike PDF, PostScript doesn't track separate colors for strokes and other elements. It is
\__draw_backend_stroke: also desirable to have the clip keyword after a stroke or fill. To achieve those outcomes,
\__draw_backend_closestroke: there is some work to do. For color, the stroke color is simple but the fill one has to be
\__draw_backend_fill: inserted by hand. For clipping, the required ordering is achieved using a TEX switch.
\__draw_backend_fillstroke: All of the operations end with a new path instruction as they do not terminate (again in
\__draw_backend_clip: contrast to PDF).
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
1171 \cs_new_protected:Npn \__draw_backend_closepath:
1172 { \__draw_backend_literal:n { closepath } }
1173 \cs_new_protected:Npn \__draw_backend_stroke:
1174 {
1175     \__draw_backend_literal:n { gsave }
1176     \__draw_backend_literal:n { color.sc }
1177     \__draw_backend_literal:n { stroke }
1178     \__draw_backend_literal:n { grestore }
1179     \bool_if:NT \g__draw_draw_clip_bool
1180     {
1181         \__draw_backend_literal:x
1182         {
1183             \bool_if:NT \g__draw_draw_eor_bool { eo }
1184             clip
1185         }
1186     }
1187     \__draw_backend_literal:n { newpath }
1188     \bool_gset_false:N \g__draw_draw_clip_bool
1189 }
1190 \cs_new_protected:Npn \__draw_backend_closestroke:
1191 {
1192     \__draw_backend_closepath:
1193     \__draw_backend_stroke:
1194 }
1195 \cs_new_protected:Npn \__draw_backend_fill:
1196 {
1197     \__draw_backend_literal:x
1198     {
1199         \bool_if:NT \g__draw_draw_eor_bool { eo }

```



```

1200         fill
1201     }
1202     \bool_if:NT \g__draw_draw_clip_bool
1203     {
1204         \__draw_backend_literal:x
1205         {
1206             \bool_if:NT \g__draw_draw_eor_bool { eo }
1207             clip
1208         }
1209     }
1210     \__draw_backend_literal:n { newpath }
1211     \bool_gset_false:N \g__draw_draw_clip_bool
1212 }
1213 \cs_new_protected:Npn \__draw_backend_fillstroke:
1214 {
1215     \__draw_backend_literal:x
1216     {
1217         \bool_if:NT \g__draw_draw_eor_bool { eo }
1218         fill
1219     }
1220     \__draw_backend_literal:n { gsave }
1221     \__draw_backend_literal:n { color.sc }
1222     \__draw_backend_literal:n { stroke }
1223     \__draw_backend_literal:n { grestore }
1224     \bool_if:NT \g__draw_draw_clip_bool
1225     {
1226         \__draw_backend_literal:x
1227         {
1228             \bool_if:NT \g__draw_draw_eor_bool { eo }
1229             clip
1230         }
1231     }
1232     \__draw_backend_literal:n { newpath }
1233     \bool_gset_false:N \g__draw_draw_clip_bool
1234 }
1235 \cs_new_protected:Npn \__draw_backend_clip:
1236 { \bool_gset_true:N \g__draw_draw_clip_bool }
1237 \bool_new:N \g__draw_draw_clip_bool
1238 \cs_new_protected:Npn \__draw_backend_discardpath:
1239 {
1240     \bool_if:NT \g__draw_draw_clip_bool
1241     {
1242         \__draw_backend_literal:x
1243         {
1244             \bool_if:NT \g__draw_draw_eor_bool { eo }
1245             clip
1246         }
1247     }
1248     \__draw_backend_literal:n { newpath }
1249     \bool_gset_false:N \g__draw_draw_clip_bool
1250 }

```

(End definition for __draw_backend_closepath: and others.)

```

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

```

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

```

1251 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1252 {
1253   \__draw_backend_literal:x
1254   {
1255     [
1256       \exp_args:Nf \use:n
1257       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1258     ] ~
1259     \dim_to_decimal_in_bp:n {#2} ~ setdash
1260   }
1261 }
1262 \cs_new:Npn \__draw_backend_dash:n #1
1263 { ~ \dim_to_decimal_in_bp:n {#1} }
1264 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1265 {
1266   \__draw_backend_literal:x
1267   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1268 }
1269 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1270 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1271 \cs_new_protected:Npn \__draw_backend_cap_but:
1272 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1273 \cs_new_protected:Npn \__draw_backend_cap_round:
1274 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1275 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1276 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1277 \cs_new_protected:Npn \__draw_backend_join_miter:
1278 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1279 \cs_new_protected:Npn \__draw_backend_join_round:
1280 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1281 \cs_new_protected:Npn \__draw_backend_join_bevel:
1282 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

(End definition for __draw_backend_dash_pattern:nn and others.)

__draw_backend_cm:nnnn In dvips, keeping the transformations in line with the engine is unfortunately not possible for scaling and rotations: even if we decompose the matrix into those operations, there is still no backend tracking (*cf.* dvipdfmx/X_YTEX). Thus we take the shortest path available and simply dump the matrix as given.

```

1283 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1284 {
1285   \__draw_backend_literal:n
1286   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1287 }

```

(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).

```

1288 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1289 {
1290   \__draw_backend_literal:n { @endspecial }
1291   \__draw_backend_literal:n { [end] }
1292   \__draw_backend_literal:n { [begin] }
1293   \__draw_backend_literal:n { save }
1294   \__draw_backend_literal:n { currentpoint }
1295   \__draw_backend_literal:n { currentpoint~translate }
1296   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1297   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1298   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1299   \__draw_backend_literal:n { neg~exch~neg~exch~translate }
1300   \__draw_backend_literal:n { [end] }
1301   \hbox_overlap_right:n { \box_use:N #1 }
1302   \__draw_backend_literal:n { [begin] }
1303   \__draw_backend_literal:n { restore }
1304   \__draw_backend_literal:n { [end] }
1305   \__draw_backend_literal:n { [begin] }
1306   \__draw_backend_literal:n { @beginspecial }
1307 }

```

(End definition for `__draw_backend_box_use:Nnnnn`.)

```

1308 </dvips>

```

4.2 Lua \TeX , pdf \TeX , dvipdfmx and X \TeX

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

```

1309 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

```

\__draw_backend_literal:n Pass data through using a dedicated interface.
\__draw_backend_literal:x
1310 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1311 \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:
1312 \cs_new_protected:Npn \__draw_backend_begin:
1313 { \__draw_backend_scope_begin: }
1314 \cs_new_protected:Npn \__draw_backend_end:
1315 { \__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:`

```

1316 \cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:
1317 \cs_new_eq:NN \_draw_backend_scope_end: \_kernel_backend_scope_end:

```

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

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

`_draw_backend_lineto:nn` to convert to bp.

`_draw_backend_curveto:nnnnnn`

`_draw_backend_rectangle:nnnn`

```

1318 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1319 {
1320   \_draw_backend_literal:x
1321   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1322 }
1323 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1324 {
1325   \_draw_backend_literal:x
1326   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1327 }
1328 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1329 {
1330   \_draw_backend_literal:x
1331   {
1332     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1333     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1334     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1335     c
1336   }
1337 }
1338 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1339 {
1340   \_draw_backend_literal:x
1341   {
1342     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1343     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1344     re
1345   }
1346 }

```

(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`

```

1347 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1348 { \bool_gset_true:N \g__draw_draw_eor_bool }
1349 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1350 { \bool_gset_false:N \g__draw_draw_eor_bool }
1351 \bool_new:N \g__draw_draw_eor_bool

```

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

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

`_draw_backend_stroke:`

`_draw_backend_closestroke:`

`_draw_backend_fill:`

`_draw_backend_fillstroke:`

`_draw_backend_clip:`

`_draw_backend_discardpath:`

```

1355 { \_draw_backend_literal:n { S } }
1356 \cs_new_protected:Npn \_draw_backend_closestroke:
1357 { \_draw_backend_literal:n { s } }
1358 \cs_new_protected:Npn \_draw_backend_fill:
1359 {
1360   \_draw_backend_literal:x
1361   { f \bool_if:NT \g__draw_draw_eor_bool * }
1362 }
1363 \cs_new_protected:Npn \_draw_backend_fillstroke:
1364 {
1365   \_draw_backend_literal:x
1366   { B \bool_if:NT \g__draw_draw_eor_bool * }
1367 }
1368 \cs_new_protected:Npn \_draw_backend_clip:
1369 {
1370   \_draw_backend_literal:x
1371   { W \bool_if:NT \g__draw_draw_eor_bool * }
1372 }
1373 \cs_new_protected:Npn \_draw_backend_discardpath:
1374 { \_draw_backend_literal:n { n } }

```

(End definition for _draw_backend_closepath: and others.)

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

```

\_draw_backend_dash_pattern:nn
\_draw_backend_dash:n
\_draw_backend_linewidth:n
\_draw_backend_miterlimit:n
\_draw_backend_cap_butt:
\_draw_backend_cap_round:
\_draw_backend_cap_rectangle:
\_draw_backend_join_miter:
\_draw_backend_join_round:
\_draw_backend_join_bevel:
1375 \cs_new_protected:Npn \_draw_backend_dash_pattern:nn #1#2
1376 {
1377   \_draw_backend_literal:x
1378   {
1379     [
1380       \exp_args:Nf \use:n
1381       { \clist_map_function:nN {#1} \_draw_backend_dash:n }
1382     ] ~
1383     \dim_to_decimal_in_bp:n {#2} ~ d
1384   }
1385 }
1386 \cs_new:Npn \_draw_backend_dash:n #1
1387 { ~ \dim_to_decimal_in_bp:n {#1} }
1388 \cs_new_protected:Npn \_draw_backend_linewidth:n #1
1389 {
1390   \_draw_backend_literal:x
1391   { \dim_to_decimal_in_bp:n {#1} ~ w }
1392 }
1393 \cs_new_protected:Npn \_draw_backend_miterlimit:n #1
1394 { \_draw_backend_literal:x { #1 ~ M } }
1395 \cs_new_protected:Npn \_draw_backend_cap_butt:
1396 { \_draw_backend_literal:n { 0 ~ J } }
1397 \cs_new_protected:Npn \_draw_backend_cap_round:
1398 { \_draw_backend_literal:n { 1 ~ J } }
1399 \cs_new_protected:Npn \_draw_backend_cap_rectangle:
1400 { \_draw_backend_literal:n { 2 ~ J } }
1401 \cs_new_protected:Npn \_draw_backend_join_miter:
1402 { \_draw_backend_literal:n { 0 ~ j } }
1403 \cs_new_protected:Npn \_draw_backend_join_round:
1404 { \_draw_backend_literal:n { 1 ~ j } }

```

```

1405 \cs_new_protected:Npn \__draw_backend_join_bevel:
1406 { \__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!

```

1407 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1408 {
1409   <*luatex | pdftex>
1410   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1411   </luatex | pdftex>
1412   <*dvipdfmx | xetex>
1413   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1414   \__draw_backend_cm_aux:nnnn
1415   </dvipdfmx | xetex>
1416 }
1417 <*dvipdfmx | xetex>
1418 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1419 {
1420   \__kernel_backend_literal:x
1421   {
1422     x:rotate~
1423     \fp_compare:nNnTF {#1} = \c_zero_fp
1424     { 0 }
1425     { \fp_eval:n { round ( -#1 , 5 ) } }
1426   }
1427   \__kernel_backend_literal:x
1428   {
1429     x:scale~
1430     \fp_eval:n { round ( #2 , 5 ) } ~
1431     \fp_eval:n { round ( #3 , 5 ) }
1432   }
1433   \__kernel_backend_literal:x
1434   {
1435     x:rotate~
1436     \fp_compare:nNnTF {#4} = \c_zero_fp
1437     { 0 }
1438     { \fp_eval:n { round ( -#4 , 5 ) } }
1439   }
1440 }
1441 </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.

```

1442 <*dvipdfmx | xetex>
1443 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1444 {
1445   \use:x
1446   {
1447     \__draw_backend_cm_decompose_auxi:nnnnN
1448     { \fp_eval:n { (#1 + #4) / 2 } }
1449     { \fp_eval:n { (#1 - #4) / 2 } }
1450     { \fp_eval:n { (#3 + #2) / 2 } }
1451     { \fp_eval:n { (#3 - #2) / 2 } }
1452   }
1453   #5
1454 }
1455 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1456 {
1457   \use:x
1458   {
1459     \__draw_backend_cm_decompose_auxii:nnnnN
1460     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1461     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1462     { \fp_eval:n { atand ( #3 , #2 ) } }
1463     { \fp_eval:n { atand ( #4 , #1 ) } }
1464   }
1465   #5
1466 }
1467 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1468 {
1469   \use:x
1470   {

```

```

1471 \__draw_backend_cm_decompose_auxiii:nnnnN
1472 { \fp_eval:n { ( #4 - #3 ) / 2 } }
1473 { \fp_eval:n { ( #1 + #2 ) / 2 } }
1474 { \fp_eval:n { ( #1 - #2 ) / 2 } }
1475 { \fp_eval:n { ( #4 + #3 ) / 2 } }
1476 }
1477 #5
1478 }
1479 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1480 {
1481 \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1482 { #5 {#1} {#2} {#3} {#4} }
1483 { #5 {#1} {#3} {#2} {#4} }
1484 }
1485 </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.

```

1486 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1487 {
1488 \__kernel_backend_scope_begin:
1489 <*luatex | pdftex>
1490 \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1491 </luatex | pdftex>
1492 <*dvipdfmx | xetex>
1493 \__kernel_backend_literal:n
1494 { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1495 </dvipdfmx | xetex>
1496 \hbox_overlap_right:n { \box_use:N #1 }
1497 <*dvipdfmx | xetex>
1498 \__kernel_backend_literal:n { pdf:etrans }
1499 </dvipdfmx | xetex>
1500 \__kernel_backend_scope_end:
1501 }

```

(End definition for `__draw_backend_box_use:Nnnnn`.)

```

1502 </dvipdfmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1503 <*dvisvgm>

```

`__draw_backend_literal:n`
`__draw_backend_literal:x`

The same as the more general literal call.

```

1504 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1505 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for `__draw_backend_literal:n`.)

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

```

1506 \cs_new_protected:Npn \__draw_backend_begin:
1507 {
1508   \__kernel_backend_scope_begin:
1509   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1510 }
1511 \cs_new_eq:NN \__draw_backend_end: \__kernel_backend_scope_end:

```

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

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

```

1512 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1513 {
1514   \__draw_backend_add_to_path:n
1515   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1516 }
1517 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1518 {
1519   \__draw_backend_add_to_path:n
1520   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1521 }
1522 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1523 {
1524   \__draw_backend_add_to_path:n
1525   {
1526     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1527     h ~ \dim_to_decimal:n {#3} ~
1528     v ~ \dim_to_decimal:n {#4} ~
1529     h ~ \dim_to_decimal:n { -#3 } ~
1530     Z
1531   }
1532 }
1533 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1534 {
1535   \__draw_backend_add_to_path:n
1536   {
1537     C ~
1538     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1539     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1540     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1541   }
1542 }
1543 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1544 {
1545   \tl_gset:Nx \g__draw_draw_path_tl
1546   {
1547     \g__draw_draw_path_tl
1548     \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1549     #1

```

```

1550     }
1551   }
1552   \tl_new:N \g__draw_draw_path_tl

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

```

__draw_backend_evenodd_rule: The fill rules here have to be handled as scopes.
 __draw_backend_nonzero_rule:

```

1553 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1554 { \__draw_backend_scope:n { fill-rule="evenodd" } }
1555 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1556 { \__draw_backend_scope:n { fill-rule="nonzero" } }

```

(End definition for __draw_backend_evenodd_rule: and __draw_backend_nonzero_rule:.)

```

\__draw_backend_path:n Setting fill and stroke effects and doing clipping all has to be done using scopes. This
\__draw_backend_closepath: means setting up the various requirements in a shared auxiliary which deals with the
\__draw_backend_stroke: bits and pieces. Clipping paths are reused for path drawing: not essential but avoids
\__draw_backend_closestroke: constructing them twice. Discarding a path needs a separate function as it's not quite
\__draw_backend_fill: the same.
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int
1557 \cs_new_protected:Npn \__draw_backend_closepath:
1558 { \__draw_backend_add_to_path:n { Z } }
1559 \cs_new_protected:Npn \__draw_backend_path:n #1
1560 {
1561   \bool_if:NTF \g__draw_draw_clip_bool
1562   {
1563     \int_gincr:N \g__draw_clip_path_int
1564     \__draw_backend_literal:x
1565     {
1566       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1567       { ?nl }
1568       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1569       < /clipPath > { ? nl }
1570       <
1571       use~xlink:href =
1572       "\c_hash_str l3path \int_use:N \g__draw_path_int " ~
1573       #1
1574       />
1575     }
1576     \__draw_backend_scope:x
1577     {
1578       clip-path =
1579       "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int )"
1580     }
1581   }
1582   {
1583     \__draw_backend_literal:x
1584     { <path ~ d=" \g__draw_draw_path_tl " ~ #1 /> }
1585   }
1586   \tl_gclear:N \g__draw_draw_path_tl
1587   \bool_gset_false:N \g__draw_draw_clip_bool
1588 }
1589 \int_new:N \g__draw_path_int
1590 \cs_new_protected:Npn \__draw_backend_stroke:
1591 { \__draw_backend_path:n { style="fill:none" } }

```

```

1592 \cs_new_protected:Npn \__draw_backend_closestroke:
1593 {
1594   \__draw_backend_closepath:
1595   \__draw_backend_stroke:
1596 }
1597 \cs_new_protected:Npn \__draw_backend_fill:
1598 { \__draw_backend_path:n { style="stroke:none" } }
1599 \cs_new_protected:Npn \__draw_backend_fillstroke:
1600 { \__draw_backend_path:n { } }
1601 \cs_new_protected:Npn \__draw_backend_clip:
1602 { \bool_gset_true:N \g__draw_draw_clip_bool }
1603 \bool_new:N \g__draw_draw_clip_bool
1604 \cs_new_protected:Npn \__draw_backend_discardpath:
1605 {
1606   \bool_if:NT \g__draw_draw_clip_bool
1607   {
1608     \int_gincr:N \g__draw_clip_path_int
1609     \__draw_backend_literal:x
1610     {
1611       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1612       { ?nl }
1613       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1614       </clipPath >
1615     }
1616     \__draw_backend_scope:x
1617     {
1618       clip-path =
1619       "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1620     }
1621   }
1622   \tl_gclear:N \g__draw_draw_path_tl
1623   \bool_gset_false:N \g__draw_draw_clip_bool
1624 }

```

(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:
1625 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1626 {
1627   \use:x
1628   {
1629     \__draw_backend_dash_aux:nn
1630     { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1631     { \dim_to_decimal:n {#2} }
1632   }
1633 }
1634 \cs_new:Npn \__draw_backend_dash:n #1
1635 { , \dim_to_decimal_in_bp:n {#1} }
1636 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1637 {
1638   \__draw_backend_scope:x
1639   {
1640     stroke-dasharray =

```

```

1641         "
1642         \tl_if_empty:oTF { \use_none:n #1 }
1643         { none }
1644         { \use_none:n #1 }
1645     " ~
1646     stroke-offset=" #2 "
1647 }
1648 }
1649 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1650 { \__draw_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1651 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1652 { \__draw_backend_scope:x { stroke-miterlimit=" #1 " } }
1653 \cs_new_protected:Npn \__draw_backend_cap_but:
1654 { \__draw_backend_scope:n { stroke-linecap="butt" } }
1655 \cs_new_protected:Npn \__draw_backend_cap_round:
1656 { \__draw_backend_scope:n { stroke-linecap="round" } }
1657 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1658 { \__draw_backend_scope:n { stroke-linecap="square" } }
1659 \cs_new_protected:Npn \__draw_backend_join_miter:
1660 { \__draw_backend_scope:n { stroke-linejoin="miter" } }
1661 \cs_new_protected:Npn \__draw_backend_join_round:
1662 { \__draw_backend_scope:n { stroke-linejoin="round" } }
1663 \cs_new_protected:Npn \__draw_backend_join_bevel:
1664 { \__draw_backend_scope:n { stroke-linejoin="bevel" } }

```

(End definition for __draw_backend_dash_pattern:nn and others.)

__draw_backend_cm:nnnn The four arguments here are floats (the affine matrix), the last two are a displacement vector.

```

1665 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1666 {
1667     \__draw_backend_scope:n
1668     {
1669         transform =
1670         " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1671     }
1672 }

```

(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.

```

1673 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1674 {
1675     \__kernel_backend_scope_begin:
1676     \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1677     \__kernel_backend_literal_svg:n
1678     {
1679         < g~
1680         stroke="none"~
1681         transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1682         >
1683     }
1684     \box_set_wd:Nn #1 { Opt }

```

```

1685 \box_set_ht:Nn #1 { Opt }
1686 \box_set_dp:Nn #1 { Opt }
1687 \box_use:N #1
1688 \__kernel_backend_literal_svg:n { </g> }
1689 \__kernel_backend_scope_end:
1690 }

(End definition for \__draw_backend_box_use:Nnnnn.)

1691 </dvisvgm>
1692 </package>

```

5 l3backend-graphics Implementation

```

1693 <*package>
1694 <@@=graphics>

```

5.1 dvips backend

```

1695 <*dvips>

```

_graphics_backend_getbb_eps:n

Simply use the generic function.

```

1696 \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.

```

1697 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1698 {
1699   \__kernel_backend_literal:x
1700   {
1701     PSfile = #1 \c_space_tl
1702     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1703     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1704     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1705     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1706   }
1707 }

```

(End definition for __graphics_backend_include_eps:n.)

```

1708 </dvips>

```

5.2 LuaTeX and pdfTeX backends

```

1709 <*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.

```

1710 \tl_new:N \l__graphics_graphics_attr_tl

```

(End definition for \l__graphics_graphics_attr_tl.)

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

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

```

1711 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1712 {
1713   \int_zero:N \l_graphics_page_int
1714   \tl_clear:N \l_graphics_pagebox_tl
1715   \tl_set:Nx \l__graphics_graphics_attr_tl
1716   {
1717     \tl_if_empty:NF \l_graphics_decodearray_tl
1718     { :D \l_graphics_decodearray_tl }
1719     \bool_if:NT \l_graphics_interpolate_bool
1720     { :I }
1721   }
1722   \tl_clear:N \l__graphics_graphics_attr_tl
1723   \_graphics_backend_getbb_auxi:n {#1}
1724 }
1725 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1726 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1727 {
1728   \tl_clear:N \l_graphics_decodearray_tl
1729   \bool_set_false:N \l_graphics_interpolate_bool
1730   \tl_set:Nx \l__graphics_graphics_attr_tl
1731   {
1732     : \l_graphics_pagebox_tl
1733     \int_compare:nNnT \l_graphics_page_int > 1
1734     { :P \int_use:N \l_graphics_page_int }
1735   }
1736   \_graphics_backend_getbb_auxi:n {#1}
1737 }
1738 \cs_new_protected:Npn \_graphics_backend_getbb_auxi:n #1
1739 {
1740   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1741   { \_graphics_backend_getbb_auxii:n {#1} }
1742 }

```

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.

```

1743 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:n #1
1744 {
1745   \tex_immediate:D \tex_pdfximage:D
1746   \bool_lazy_or:nnT
1747   { \l_graphics_interpolate_bool }
1748   { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1749   {
1750     attr ~
1751     {
1752       \tl_if_empty:NF \l_graphics_decodearray_tl
1753       { /Decode~[ \l_graphics_decodearray_tl ] }
1754       \bool_if:NT \l_graphics_interpolate_bool
1755       { /Interpolate~true }

```

```

1756     }
1757   }
1758   \int_compare:nNnT \l_graphics_page_int > 0
1759     { page ~ \int_use:N \l_graphics_page_int }
1760   \tl_if_empty:NF \l_graphics_pagebox_tl
1761     { \l_graphics_pagebox_tl }
1762   {#1}
1763   \hbox_set:Nn \l__graphics_internal_box
1764     { \tex_pdfrefximage:D \tex_pdflastximage:D }
1765   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1766   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1767   \int_const:cn { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1768     { \tex_the:D \tex_pdflastximage:D }
1769   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1770 }

```

(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.

```

1771 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1772 {
1773   \tex_pdfrefximage:D
1774   \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1775 }
1776 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1777 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

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

`__graphics_backend_getbb_eps:n`
`__graphics_backend_getbb_eps:nm`
`__graphics_backend_include_eps:n`
`\l__graphics_backend_dir_str`
`\l__graphics_backend_name_str`
`\l__graphics_backend_ext_str`

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.

```

1778 \sys_if_shell:T
1779 {
1780   \str_new:N \l__graphics_backend_dir_str
1781   \str_new:N \l__graphics_backend_name_str
1782   \str_new:N \l__graphics_backend_ext_str
1783   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1784     {
1785       \file_parse_full_name:nNNN {#1}
1786       \l__graphics_backend_dir_str
1787       \l__graphics_backend_name_str
1788       \l__graphics_backend_ext_str
1789       \exp_args:Nx \__graphics_backend_getbb_eps:nn
1790         {
1791           \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1792           -converted-to.pdf
1793         }
1794       {#1}
1795     }
1796   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2

```

```

1797 {
1798   \file_compare_timestamp:nNnT {#2} > {#1}
1799   {
1800     \sys_shell_now:n
1801     { repstopdf ~ #2 ~ #1 }
1802   }
1803   \tl_set:Nn \l_graphics_name_tl {#1}
1804   \__graphics_backend_getbb_pdf:n {#1}
1805 }
1806 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1807 {
1808   \file_parse_full_name:nNNN {#1}
1809   \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1810   \exp_args:Nx \__graphics_backend_include_pdf:n
1811   {
1812     \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1813     -converted-to.pdf
1814   }
1815 }
1816 }

```

(End definition for __graphics_backend_getbb_eps:n and others.)

1817 </luatex | pdftex>

5.3 dvipdfmx backend

1818 <*dvipdfmx | xetex>

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

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
1819 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n
1820 <*dvipdfmx>
1821 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1822 {
1823   \int_zero:N \l_graphics_page_int
1824   \tl_clear:N \l_graphics_pagebox_tl
1825   \graphics_extract_bb:n {#1}
1826 }
1827 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1828 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1829 {
1830   \tl_clear:N \l_graphics_decodearray_tl
1831   \bool_set_false:N \l_graphics_interpolate_bool
1832   \graphics_extract_bb:n {#1}
1833 }
1834 </dvipdfmx>

```

(End definition for __graphics_backend_getbb_eps:n and others.)

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

1835 \int_new:N \g__graphics_track_int

(End definition for \g__graphics_track_int.)

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

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

```

1836 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1837 {
1838   \_kernel_backend_literal:x
1839   {
1840     PSfile = #1 \c_space_tl
1841     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1842     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1843     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1844     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1845   }
1846 }
1847 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1848 { \_graphics_backend_include_auxi:nn {#1} { image } }
1849 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1850 <dvipdfmx>
1851 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1852 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1853 </dvipdfmx>

```

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

```

1854 \cs_new_protected:Npn \_graphics_backend_include_auxi:nn #1#2
1855 {
1856   \_graphics_backend_include_auxii:xnn
1857   {
1858     \tl_if_empty:NF \l_graphics_pagebox_tl
1859     { : \l_graphics_pagebox_tl }
1860     \int_compare:nNnT \l_graphics_page_int > 1
1861     { :P \int_use:N \l_graphics_page_int }
1862     \tl_if_empty:NF \l_graphics_decodearray_tl
1863     { :D \l_graphics_decodearray_tl }
1864     \bool_if:NT \l_graphics_interpolate_bool
1865     { :I }
1866   }
1867   {#1} {#2}
1868 }
1869 \cs_new_protected:Npn \_graphics_backend_include_auxii:nnn #1#2#3
1870 {
1871   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1872   {
1873     \_kernel_backend_literal:x
1874     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1875   }
1876   { \_graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1877 }
1878 \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!

```

1879 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1880 {
1881   \int_gincr:N \g__graphics_track_int
1882   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1883   \__kernel_backend_literal:x
1884   {
1885     pdf:#3~
1886     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1887     \int_compare:nNnT \l_graphics_page_int > 1
1888     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1889     \tl_if_empty:NF \l_graphics_pagebox_tl
1890     {
1891       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1892       bbox ~
1893         \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1894         \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1895         \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1896         \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1897     }
1898     (#1)
1899     \bool_lazy_or:nnT
1900     { \l_graphics_interpolate_bool }
1901     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1902     {
1903       <<
1904       \tl_if_empty:NF \l_graphics_decodearray_tl
1905       { /Decode~[ \l_graphics_decodearray_tl ] }
1906       \bool_if:NT \l_graphics_interpolate_bool
1907       { /Interpolate~true> }
1908     }
1909     >>
1910   }
1911 }

```

(End definition for __graphics_backend_include_eps:n and others.)

```

1912 </dvipdfmx | xetex>

```

5.4 Xe_{La}TeX backend

```

1913 <*xetex>

```

5.4.1 Images

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

```

1914 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1915 {
1916   \int_zero:N \l_graphics_page_int
1917   \tl_clear:N \l_graphics_pagebox_tl
1918   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1919 }

```

```

\__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:nnN
\__graphics_backend_getbb_auxiv:nnN
\__graphics_backend_getbb_auxiv:nnN
\__graphics_backend_getbb_auxv:nnN
\__graphics_backend_getbb_auxv:nnN
\__graphics_backend_getbb_pagebox:w

```

```

1920 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1921 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1922 {
1923   \tl_clear:N \l_graphics_decodearray_tl
1924   \bool_set_false:N \l_graphics_interpolate_bool
1925   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1926 }
1927 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
1928 {
1929   \int_compare:nNnTF \l_graphics_page_int > 1
1930     { \__graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
1931     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1932 }
1933 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
1934 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1935 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
1936 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1937 {
1938   \tl_if_empty:NTF \l_graphics_pagebox_tl
1939     { \__graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1940     { \__graphics_backend_getbb_auxv:nNnn }
1941     {#1} #2 {#3} {#4}
1942 }
1943 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
1944 {
1945   \use:x
1946   {
1947     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1948     { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1949   }
1950 }
1951 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1952 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1953 {
1954   \graphics_bb_restore:nF {#1#3}
1955   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1956 }
1957 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1958 {
1959   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1960   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1961   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1962   \graphics_bb_save:n {#1#3}
1963 }
1964 \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

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

```

1965 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1966 {

```

```

1967 \tex_XeTeXpdffile:D
1968 \__graphics_backend_include_pdf_quote:w #1 "#1" \s__graphics_stop \c_space_tl
1969 \int_compare:nNnT \l_graphics_page_int > 0
1970 { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1971 \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1972 }
1973 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \s__graphics_stop
1974 { " #2 " }

```

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

```
1975 </xetex>
```

5.5 dvisvgm backend

```
1976 <*dvisvgm>
```

__graphics_backend_getbb_eps:n Simply use the generic function.

```
1977 \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
1978 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1979 {
1980   \int_zero:N \l_graphics_page_int
1981   \tl_clear:N \l_graphics_pagebox_tl
1982   \graphics_extract_bb:n {#1}
1983 }
1984 \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

```

1985 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1986 {
1987   \tl_clear:N \l_graphics_decodearray_tl
1988   \bool_set_false:N \l_graphics_interpolate_bool
1989   \graphics_extract_bb:n {#1}
1990 }

```

(End definition for __graphics_backend_getbb_pdf:n.)

__graphics_backend_include_eps:n The special syntax is relatively clear here: remember we need PostScript sizes here. (This is the same as the dvips code.)

```

\__graphics_backend_include_pdf:n
\__graphics_backend_include:nn
1991 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1992 { __graphics_backend_include:nn { PSfile } {#1} }
1993 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1994 { __graphics_backend_include:nn { pdffile } {#1} }
1995 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
1996 {
1997   \__kernel_backend_literal:x
1998   {
1999     #1 = #2 \c_space_tl
2000     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl

```

```

2001         lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2002         urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2003         ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
2004     }
2005 }

```

(End definition for `__graphics_backend_include_eps:n`, `__graphics_backend_include_pdf:n`, and `__graphics_backend_include:nn`.)

```

\__graphics_backend_include_png:n
\__graphics_backend_include_jpg:n
\__graphics_backend_include_bitmap_quote:w

```

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

```

2006 \cs_new_protected:Npn \__graphics_backend_include_png:n #1
2007 {
2008     \__kernel_backend_literal:x
2009     {
2010         dvisvgm:img~
2011         \dim_to_decimal:n { \l_graphics_ury_dim } ~
2012         \dim_to_decimal:n { \l_graphics_ury_dim } ~
2013         \__graphics_backend_include_bitmap_quote:w #1 " #1 " \s__graphics_stop
2014     }
2015 }
2016 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_png:n
2017 \cs_new:Npn \__graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \s__graphics_stop
2018 { " #2 " }

```

(End definition for `__graphics_backend_include_png:n`, `__graphics_backend_include_jpg:n`, and `__graphics_backend_include_bitmap_quote:w`.)

```

2019 </dvisvgm>
2020 </package>

```

6 l3backend-pdf Implementation

```

2021 <*package>
2022 <@@=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

2023 \box_new:N \l__pdf_internal_box

(End definition for \l__pdf_internal_box.)

```

6.2 dvips backend

2024 `<*dvips>`

`__pdf_backend_pdfmark:n`
`__pdf_backend_pdfmark:x`

Used often enough it should be a separate function.

2025 `\cs_new_protected:Npn __pdf_backend_pdfmark:n #1`
2026 `{ __kernel_backend_postscript:n { mark #1 ~ pdfmark } }`
2027 `\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`

2028 `\cs_new_protected:Npn __pdf_backend_catalog_gput:nn #1#2`
2029 `{ __pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }`
2030 `\cs_new_protected:Npn __pdf_backend_info_gput:nn #1#2`
2031 `{ __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.

2032 `\int_new:N \g__pdf_backend_object_int`
2033 `\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`.

2034 `\cs_new_protected:Npn __pdf_backend_object_new:nn #1#2`
2035 `{`
2036 `\int_gincr:N \g__pdf_backend_object_int`
2037 `\int_const:cn`
2038 `{ c__pdf_backend_object_ \tl_to_str:n {#1} _int }`
2039 `{ \g__pdf_backend_object_int }`
2040 `\prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}`
2041 `}`
2042 `\cs_new:Npn __pdf_backend_object_ref:n #1`
2043 `{ { pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } } }`

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

`_pdf_backend_object_write:nn`
`_pdf_backend_object_write:nx`
`__pdf_backend_object_write_array:nn`
`_pdf_backend_object_write_dict:nn`
`__pdf_backend_object_write_fstream:nn`
`_pdf_backend_object_write_stream:nn`
`_pdf_backend_object_write_stream:nmm`

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

2044 `\cs_new_protected:Npn __pdf_backend_object_write:nn #1#2`
2045 `{`
2046 `__pdf_backend_pdfmark:x`
2047 `{`
2048 `/_objdef ~ __pdf_backend_object_ref:n {#1}`
2049 `/type`
2050 `\str_case_e:nn`
2051 `{ \prop_item:Nn \g__pdf_backend_object_prop {#1} }`
2052 `{`
2053 `{ array } { /array }`
2054 `{ dict } { /dict }`

```

2055         { fstream } { /stream }
2056         { stream } { /stream }
2057     }
2058     /OBJ
2059 }
2060 \use:c
2061 { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2062 { __pdf_backend_object_ref:n {#1} } {#2}
2063 }
2064 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2065 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2066 {
2067     \__pdf_backend_pdfmark:x
2068     { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2069 }
2070 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2071 {
2072     \__pdf_backend_pdfmark:x
2073     { #1 << \exp_not:n {#2} >> /PUT }
2074 }
2075 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2076 {
2077     \exp_args:Nx
2078     \__pdf_backend_object_write_fstream:nnn {#1} #2
2079 }
2080 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2081 {
2082     \__kernel_backend_postscript:n
2083     {
2084         SDict ~ begin ~
2085         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2086         mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2087         end
2088     }
2089 }
2090 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2091 {
2092     \exp_args:Nx
2093     \__pdf_backend_object_write_stream:nnn {#1} #2
2094 }
2095 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2096 {
2097     \__kernel_backend_postscript:n
2098     {
2099         mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2100         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2101     }
2102 }

```

(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.

```

2103 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2104 {

```

```

2105 \int_gincr:N \g__pdf_backend_object_int
2106 \__pdf_backend_pdfmark:x
2107 {
2108   /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2109   /type
2110   \str_case:nn
2111     {#1}
2112     {
2113       { array } { /array }
2114       { dict } { /dict }
2115       { fstream } { /stream }
2116       { stream } { /stream }
2117     }
2118   /OBJ
2119 }
2120 \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2121 { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2122 }
2123 \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.

```

2124 \cs_new:Npn \__pdf_backend_object_last:
2125   { { 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.

```

2126 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2127   { { 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.

```

2128 \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.

```

2129 \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.

```

2130 \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.

```
2131 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2132 {
2133   \exp_args:Nf \_pdf_backend_annotation_aux:nnnn
2134   { \dim_eval:n {#1} } {#2} {#3} {#4}
2135 }
2136 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2137 {
2138   \box_move_down:nn {#3}
2139   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2140   \box_move_up:nn {#2}
2141   {
2142     \hbox:n
2143     {
2144       \_kernel_kern:n {#1}
2145       \_kernel_backend_postscript:n { pdf.save.ur }
2146       \_kernel_kern:n { -#1 }
2147     }
2148   }
2149   \int_gincr:N \g__pdf_backend_object_int
2150   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2151   \_pdf_backend_pdfmark:x
2152   {
2153     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2154     pdf.rect
2155     #4 ~
2156     /ANN
2157   }
2158 }
```

(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.

```
2159 \cs_new:Npn \_pdf_backend_annotation_last:
2160 { { 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.

```
2161 \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.

```
2162 \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.

```
2163 \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.

2164 `\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.

2165 `\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.

2166 `\tl_new:N \l__pdf_breaklink_pdfmark_tl`

2167 `\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.

2168 `\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.

2169 `\cs_new_eq:NN __pdf_breaklink_usebox:N \box_use:N`

(End definition for `__pdf_breaklink_usebox:N`.)

`__pdf_backend_link_begin_goto:nnw` 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`.

`__pdf_backend_link_begin_user:nnw`

`__pdf_backend_link:nw` 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).

`__pdf_backend_link_aux:nw`

`__pdf_backend_link_end:`

`__pdf_backend_link_end_aux:`

`__pdf_backend_link_minima:` 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).

`__pdf_backend_link_outerbox:n`

`__pdf_backend_link_sf_save:`

`__pdf_backend_link_sf_restore:`

`pdf.linkdp.pad` The result should be similar to `pdfTeX` in the vast majority of foreseeable cases.

`pdf.linkht.pad` 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`.

`pdf.llx`

`pdf.lly`

`pdf.ury`

`pdf.link.dict` 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.

`pdf.outerbox`

`pdf.baselineskip`

2170 `\cs_new_protected:Npn __pdf_backend_link_begin_goto:nnw #1#2`

2171 `{`

2172 `__pdf_backend_link_begin:nw`

2173 `{ #1 /Subtype /Link /Action << /S /GoTo /D (#2) >> }`

2174 `}`

2175 `\cs_new_protected:Npn __pdf_backend_link_begin_user:nnw #1#2`

2176 `{ __pdf_backend_link_begin:nw {#1#2} }`

2177 `\cs_new_protected:Npn __pdf_backend_link_begin:nw #1`

2178 `{`

```

2179 \bool_if:NF \g__pdf_backend_link_bool
2180 { \__pdf_backend_link_begin_aux:nw {#1} }
2181 }

```

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.

```

2182 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2183 {
2184   \bool_gset_true:N \g__pdf_backend_link_bool
2185   \__kernel_backend_postscript:n
2186   { /pdf.link.dict ( #1 ) def }
2187   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2188   \__pdf_backend_link_sf_save:
2189   \mode_if_math:TF
2190     { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2191     { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2192   \hbox_set:Nw \l__pdf_backend_content_box
2193   \__pdf_backend_link_sf_restore:
2194   \bool_if:NT \g__pdf_backend_link_math_bool
2195     { \c_math_toggle_token }
2196 }
2197 \cs_new_protected:Npn \__pdf_backend_link_end:
2198 {
2199   \bool_if:NT \g__pdf_backend_link_bool
2200   { \__pdf_backend_link_end_aux: }
2201 }
2202 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2203 {
2204   \bool_if:NT \g__pdf_backend_link_math_bool
2205   { \c_math_toggle_token }
2206   \__pdf_backend_link_sf_save:
2207   \hbox_set_end:
2208   \__pdf_backend_link_minima:
2209   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2210   \exp_args:Nx \__pdf_backend_link_outerbox:n
2211   {
2212     \int_if_odd:nTF { \value { page } }
2213       { \oddsidemargin }
2214       { \evensidemargin }
2215   }
2216   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2217   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2218   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2219   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2220   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2221   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2222   {
2223     \hbox:n
2224     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2225   }
2226   \int_gincr:N \g__pdf_backend_object_int
2227   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2228   \__kernel_backend_postscript:x
2229   {

```

```

2230     mark
2231     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2232     \g__pdf_backend_link_dict_tl \c_space_tl
2233     pdf.rect
2234     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2235   }
2236   \__pdf_backend_link_sf_restore:
2237   \bool_gset_false:N \g__pdf_backend_link_bool
2238 }
2239 \cs_new_protected:Npn \__pdf_backend_link_minima:
2240 {
2241   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2242   \__kernel_backend_postscript:x
2243   {
2244     /pdf.linkdp.pad ~
2245     \dim_to_decimal:n
2246     {
2247       \dim_max:nn
2248       {
2249         \box_dp:N \l__pdf_backend_model_box
2250         - \box_dp:N \l__pdf_backend_content_box
2251       }
2252       { Opt }
2253     } ~
2254     pdf.pt.dvi ~ def
2255   /pdf.linkht.pad ~
2256   \dim_to_decimal:n
2257   {
2258     \dim_max:nn
2259     {
2260       \box_ht:N \l__pdf_backend_model_box
2261       - \box_ht:N \l__pdf_backend_content_box
2262     }
2263     { Opt }
2264   } ~
2265   pdf.pt.dvi ~ def
2266 }
2267 }
2268 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2269 {
2270   \__kernel_backend_postscript:x
2271   {
2272     /pdf.outerbox
2273     [
2274       \dim_to_decimal:n {#1} ~
2275       \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2276       \dim_to_decimal:n { #1 + \textwidth } ~
2277       \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2278     ]
2279     [ exch { pdf.pt.dvi } forall ] def
2280   /pdf.baselineskip ~
2281   \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2282   { pdf.pt.dvi ~ def }
2283   { pop ~ pop }

```

```

2284         ifelse
2285     }
2286 }
2287 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2288 {
2289     \int_gset:Nn \g__pdf_backend_link_sf_int
2290     {
2291         \mode_if_horizontal:TF
2292         { \tex_spacefactor:D }
2293         { 0 }
2294     }
2295 }
2296 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2297 {
2298     \mode_if_horizontal:T
2299     {
2300         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2301         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2302     }
2303 }

```

(End definition for `__pdf_backend_link_begin_goto:nw` and others. These functions are documented on page ??.)

`\@makecol@hook` Hooks to allow link breaking: something will be needed in format mode at some stage. At present this code is disabled as there is an open question about the name of the hook: to be resolved at the L^AT_EX 2_ε end.

```

2304 \use_none:n
2305 {
2306     \cs_if_exist:NT \@makecol@hook
2307     {
2308         \tl_put_right:Nn \@makecol@hook
2309         {
2310             \box_if_empty:NF \@cclv
2311             {
2312                 \vbox_set:Nn \@cclv
2313                 {
2314                     \__kernel_backend_postscript:n
2315                     {
2316                         pdf.globaldict /pdf.brokenlink.rect ~ known
2317                         { pdf.bordertracking.continue }
2318                         if
2319                     }
2320                     \vbox_unpack_drop:N \@cclv
2321                     \__kernel_backend_postscript:n
2322                     { pdf.bordertracking.endpage }
2323                 }
2324             }
2325         }
2326         \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2327         \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2328         \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2329     }
2330 }

```

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

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

```
2331 \cs_new:Npn \_pdf_backend_link_last:
2332 { { 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.

```
2333 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2334 {
2335   \__kernel_backend_postscript:x
2336   {
2337     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2338   }
2339 }
```

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

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

```

2371 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2372 {
2373   \vbox_to_zero:n
2374   {
2375     \__kernel_kern:n {#4}
2376     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2377     \tex_vss:D
2378   }
2379   \__kernel_kern:n {#1}
2380   \vbox_to_zero:n
2381   {
2382     \__kernel_kern:n { -#3 }
2383     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2384     \tex_vss:D
2385   }
2386   \__kernel_kern:n { -#1 }
2387   \__pdf_backend_pdfmark:n
2388   {
2389     /View
2390     [
2391       /FitR ~
2392       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2393       pdf.urx ~ pdf.ury ~ pdf.dest2device
2394     ]
2395     /Dest ( #2 ) cvn
2396     /DEST
2397   }
2398 }

```

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

```

```

2418         { pop << /CompressStreams ~ false >> setdistillerparams }
2419       if
2420     }
2421   }
2422 }

```

(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`

```

2423 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2424 {
2425   \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2426 }
2427 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2428 {
2429   \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2430 }

```

(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:`

```

2431 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2432 \cs_new:Npn \_pdf_backend_version_minor: { -1 }

```

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

6.2.5 Marked content

`_pdf_backend_bdc:nn`

Simple wrappers.

`_pdf_backend_emc:`

```

2433 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2434 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2435 \cs_new_protected:Npn \_pdf_backend_emc:
2436 { \_pdf_backend_pdfmark:n { /EMC } }

```

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

2437 \langle /dvips \rangle

6.3 LuaTeX and pdfTeX backend

2438 \langle *luatex | pdftex \rangle

6.3.1 Annotations

`_pdf_backend_annotation:nnnn`

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

```

2439 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2440 {
2441    $\langle$ *luatex $\rangle$ 
2442     \tex_pdfextension:D annot ~
2443    $\langle$ /luatex $\rangle$ 
2444    $\langle$ *pdftex $\rangle$ 
2445     \tex_pdfannot:D
2446    $\langle$ /pdftex $\rangle$ 
2447     width ~ \dim_eval:n {#1} ~
2448     height ~ \dim_eval:n {#2} ~
2449     depth ~ \dim_eval:n {#3} ~

```



```

2450     {#4}
2451   }

```

(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.

```

2452 \cs_new:Npx \_pdf_backend_annotation_last:
2453 {
2454   \exp_not:N \int_value:w
2455   <*luatex>
2456   \exp_not:N \tex_pdffeedback:D lastannot ~
2457   </luatex>
2458   <*pdftex>
2459   \exp_not:N \tex_pdflastannot:D
2460   </pdftex>
2461   \c_space_tl 0 ~ R
2462 }

```

(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 2463 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
\_pdf_backend_link_begin:nnnw      2464 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
\_pdf_backend_link_begin:nnnw      2465 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
\_pdf_backend_link_end:            2466 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2467 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2468 {
2469   <*luatex>
2470   \tex_pdfextension:D startlink ~
2471   </luatex>
2472   <*pdftex>
2473   \tex_pdfstartlink:D
2474   </pdftex>
2475   attr {#1}
2476   #2 {#3}
2477 }
2478 \cs_new_protected:Npn \_pdf_backend_link_end:
2479 {
2480   <*luatex>
2481   \tex_pdfextension:D endlink \scan_stop:
2482   </luatex>
2483   <*pdftex>
2484   \tex_pdfendlink:D
2485   </pdftex>
2486 }

```

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

`_pdf_backend_link_last:` Formatted for direct use.

```

2487 \cs_new:Npx \_pdf_backend_link_last:
2488 {
2489   \exp_not:N \int_value:w
2490   <*luatex>

```

```

2491 \exp_not:N \tex_pdffeedback:D lastlink ~
2492 </luatex>
2493 <*pdfTeX>
2494 \exp_not:N \tex_pdflastlink:D
2495 </pdfTeX>
2496 \c_space_tl 0 ~ R
2497 }

```

(End definition for _pdf_backend_link_last:.)

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

```

2498 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2499 {
2500 <*luatex>
2501 \tex_pdfvariable:D linkmargin
2502 </luatex>
2503 <*pdfTeX>
2504 \tex_pdflinkmargin:D
2505 </pdfTeX>
2506 \dim_eval:n {#1} \scan_stop:
2507 }

```

(End definition for _pdf_backend_link_margin:n.)

_pdf_backend_destination:nn 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.

_pdf_backend_destination:nnnn

```

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

```

```

2535 </luatex>
2536 <*pdftex>
2537   \tex_pdfdest:D
2538 </pdftex>
2539   name {#1}
2540   fitr ~
2541   width \dim_eval:n {#2} ~
2542   height \dim_eval:n {#3} ~
2543   depth \dim_eval:n {#4} \scan_stop:
2544 }

```

(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
2545 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2546 {
2547 <*luatex>
2548   \tex_pdfextension:D catalog
2549 </luatex>
2550 <*pdftex>
2551   \tex_pdfcatalog:D
2552 </pdftex>
2553   { / #1 ~ #2 }
2554 }
2555 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2556 {
2557 <*luatex>
2558   \tex_pdfextension:D info
2559 </luatex>
2560 <*pdftex>
2561   \tex_pdfinfo:D
2562 </pdftex>
2563   { / #1 ~ #2 }
2564 }

```

(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.

```

2565 \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

```

2566 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2
2567 {
2568 <*luatex>
2569   \tex_pdfextension:D obj ~
2570 </luatex>
2571 <*pdftex>
2572   \tex_pdfobj:D

```

```

2573 </pdfTex>
2574     reserveobjnum ~
2575     \int_const:cn
2576     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2577 <*luatex>
2578     { \tex_pdffeedback:D lastobj }
2579 </luatex>
2580 <*pdfTex>
2581     { \tex_pdflastobj:D }
2582 </pdfTex>
2583     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2584 }
2585 \cs_new:Npn \__pdf_backend_object_ref:n #1
2586 { \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:nn 2587 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
\__pdf_exp_not_i:nn           2588 {
\__pdf_exp_not_ii:nn          2589 <*luatex>
2590     \tex_immediate:D \tex_pdfextension:D obj ~
2591 </luatex>
2592 <*pdfTex>
2593     \tex_immediate:D \tex_pdfobj:D
2594 </pdfTex>
2595     useobjnum ~
2596     \int_use:c
2597     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2598     \str_case:e:nn
2599     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2600     {
2601         { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2602         { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2603         { fstream }
2604         {
2605             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2606             file ~ { \__pdf_exp_not_ii:nn #2 }
2607         }
2608         { stream }
2609         {
2610             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2611             { \__pdf_exp_not_ii:nn #2 }
2612         }
2613     }
2614 }
2615 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2616 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2617 \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 2618 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2

```

```

2619 {
2620 <*luatex>
2621 \tex_immediate:D \tex_pdfextension:D obj ~
2622 </luatex>
2623 <*pdftex>
2624 \tex_immediate:D \tex_pdfobj:D
2625 </pdftex>
2626 \str_case:nn
2627 {#1}
2628 {
2629 { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2630 { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2631 { fstream }
2632 {
2633 stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2634 file ~ { \__pdf_exp_not_ii:nn #2 }
2635 }
2636 { stream }
2637 {
2638 stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2639 { \__pdf_exp_not_ii:nn #2 }
2640 }
2641 }
2642 }
2643 \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.

```

2644 \cs_new:Npx \__pdf_backend_object_last:
2645 {
2646 \exp_not:N \int_value:w
2647 <*luatex>
2648 \exp_not:N \tex_pdffeedback:D lastobj ~
2649 </luatex>
2650 <*pdftex>
2651 \exp_not:N \tex_pdflastobj:D
2652 </pdftex>
2653 \c_space_tl 0 ~ R
2654 }

```

(End definition for __pdf_backend_object_last:.)

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

```

2655 \cs_new:Npx \__pdf_backend_pageobject_ref:n #1
2656 {
2657 \exp_not:N \int_value:w
2658 <*luatex>
2659 \exp_not:N \tex_pdffeedback:D pageref
2660 </luatex>
2661 <*pdftex>
2662 \exp_not:N \tex_pdfpageref:D
2663 </pdftex>
2664 \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2665 }

```

(End definition for `_pdf_backend_pageobject_ref:n`.)

6.3.4 Structure

Simply pass data to the engine.

```

\_pdf_backend_compresslevel:n
\_pdf_backend_compress_objects:n
\_pdf_backend_objcompresslevel:n
2666 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2667 {
2668   \tex_global:D
2669   \*luatex
2670   \tex_pdfvariable:D compresslevel
2671   \*luatex
2672   \*pdftex
2673   \tex_pdfcompresslevel:D
2674   \*pdftex
2675   \int_value:w \int_eval:n {#1} \scan_stop:
2676 }
2677 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2678 {
2679   \bool_if:nTF {#1}
2680   { \_pdf_backend_objcompresslevel:n { 2 } }
2681   { \_pdf_backend_objcompresslevel:n { 0 } }
2682 }
2683 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2684 {
2685   \tex_global:D
2686   \*luatex
2687   \tex_pdfvariable:D objcompresslevel
2688   \*luatex
2689   \*pdftex
2690   \tex_pdfobjcompresslevel:D
2691   \*pdftex
2692   #1 \scan_stop:
2693 }

```

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

`_pdf_backend_version_major_gset:n`
`_pdf_backend_version_minor_gset:n`

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

```

2694 \cs_new_protected:Npx \_pdf_backend_version_major_gset:n #1
2695 {
2696   \*luatex
2697   \int_compare:nNnT \tex_luatexversion:D > { 106 }
2698   {
2699     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2700     \exp_not:N \int_eval:n {#1} \scan_stop:
2701   }
2702   \*luatex
2703   \*pdftex
2704   \cs_if_exist:NT \tex_pdfmajorversion:D
2705   {
2706     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2707     \exp_not:N \int_eval:n {#1} \scan_stop:
2708   }
2709   \*pdftex

```

```

2710 }
2711 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2712 {
2713   \tex_global:D
2714   \*luatex
2715   \tex_pdfvariable:D minorversion
2716   \*luatex
2717   \*pdftex
2718   \tex_pdfminorversion:D
2719   \*pdftex
2720   \int_eval:n {#1} \scan_stop:
2721 }

```

(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: 2722 \cs_new:Npx \__pdf_backend_version_major:
2723 {
2724   \*luatex
2725   \int_compare:nNnTF \tex luatexversion:D > { 106 }
2726   { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2727   { 1 }
2728   \*luatex
2729   \*pdftex
2730   \cs_if_exist:NTF \tex_pdfmajorversion:D
2731   { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2732   { 1 }
2733   \*pdftex
2734 }
2735 \cs_new:Npn \__pdf_backend_version_minor:
2736 {
2737   \tex_the:D
2738   \*luatex
2739   \tex_pdfvariable:D minorversion
2740   \*luatex
2741   \*pdftex
2742   \tex_pdfminorversion:D
2743   \*pdftex
2744 }

```

(End definition for __pdf_backend_version_major: and __pdf_backend_version_minor:.)

6.3.5 Marked content

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

```

2745 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2746 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2747 \cs_new_protected:Npn \__pdf_backend_emc:
2748 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2749 \*luatex | \*pdftex

```

6.4 dvipdfmx backend

2750 `*dvipdfmx|xetex`

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

`__pdf_backend:x`

```

2751 \cs_new_protected:Npx \__pdf_backend:n #1
2752 { \__kernel_backend_literal:n { pdf: #1 } }
2753 \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
2754 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2755 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2756 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2757 { \__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` For tracking objects to allow finalisation.

`\g__pdf_backend_object_prop`

```

2758 \int_new:N \g__pdf_backend_object_int
2759 \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` Objects are tracked at the macro level, but we don't have to do anything at this stage.

`__pdf_backend_object_ref:n`

```

2760 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2761 {
2762   \int_gincr:N \g__pdf_backend_object_int
2763   \int_const:cn
2764     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2765     { \g__pdf_backend_object_int }
2766   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2767 }
2768 \cs_new:Npn \__pdf_backend_object_ref:n #1
2769 { @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` This is where we choose the actual type.

`__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`

```

2770 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2771 {
2772   \exp_args:Nx \__pdf_backend_object_write:nnn
2773     { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2774 }
2775 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2776 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2777 {
2778   \use:c { __pdf_backend_object_write_ #1 :nn }
2779     { \__pdf_backend_object_ref:n {#2} } {#3}
2780 }

```



```

2781 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2782 {
2783   \__pdf_backend:x
2784   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2785 }
2786 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2787 {
2788   \__pdf_backend:x
2789   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2790 }
2791 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2792 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2793 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2794 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2795 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2796 {
2797   \__pdf_backend:x
2798   {
2799     #1 stream ~ #2 ~
2800     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2801   }
2802 }

```

(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
2803 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2804 {
2805   \int_gincr:N \g__pdf_backend_object_int
2806   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2807   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2808   {#2}
2809 }
2810 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2811 \cs_new:Npn \__pdf_backend_object_last:
2812 { @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_TTeX.

```

2813 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2814 { @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.

```
2815 \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.

```
2816 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2817 {
2818   \int_gincr:N \g_pdf_backend_object_int
2819   \int_gset_eq:NN \g_pdf_backend_annotation_int \g_pdf_backend_object_int
2820   \_pdf_backend:x
2821   {
2822     ann ~ @pdf.obj \int_use:N \g_pdf_backend_object_int \c_space_tl
2823     width ~ \dim_eval:n {#1} ~
2824     height ~ \dim_eval:n {#2} ~
2825     depth ~ \dim_eval:n {#3} ~
2826     << /Type /Annot #4 >>
2827   }
2828 }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

`_pdf_backend_annotation_last:`

```
2829 \cs_new:Npn \_pdf_backend_annotation_last:
2830 { @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.

```
2831 \int_new:N \g_pdf_backend_link_int
```

(End definition for `\g_pdf_backend_link_int`.)

`_pdf_backend_link_begin_goto:nnw` All created using the same internals.

`_pdf_backend_link_begin_user:nnw`

`_pdf_backend_link_begin:n`

`_pdf_backend_link_end:`

```
2832 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2833 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2834 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2835 { \_pdf_backend_link_begin:n {#1#2} }
2836 \cs_new_protected:Npx \_pdf_backend_link_begin:n #1
2837 {
2838   \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2839   {
2840     \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int
2841   }
2842   \_pdf_backend:x
2843   {
2844     bann ~
2845     \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2846     {
2847       @pdf.lnk
2848       \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int
2849       \c_space_tl
2850     }
2851   }
2852 }
```

```

2851         <<
2852         /Type /Annot
2853         #1
2854         >>
2855     }
2856 }
2857 \cs_new_protected:Npn \__pdf_backend_link_end:
2858 { \__pdf_backend:n { eann } }

```

(End definition for __pdf_backend_link_begin_goto:nw and others.)

__pdf_backend_link_last: Available using the backend mechanism with a suitably-recent version.

```

2859 \cs_new:Npx \__pdf_backend_link_last:
2860 {
2861     \int_compare:nNnF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
2862     {
2863         @pdf.lnk
2864         \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
2865     }
2866 }

```

(End definition for __pdf_backend_link_last:.)

__pdf_backend_link_margin:n Pass to dvipdfmx.

```

2867 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2868 { \__kernel_backend_literal:x { dvipdfmx:config~g~ \dim_eval:n {#1} } }

```

(End definition for __pdf_backend_link_margin:n.)

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

```

2869 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2870 {
2871     \__pdf_backend:x
2872     {
2873         dest ~ ( \exp_not:n {#1} )
2874         [
2875             @thispage
2876             \str_case:nnF {#2}
2877             {
2878                 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2879                 { fit } { /Fit }
2880                 { fitb } { /FitB }
2881                 { fitbh } { /FitBH }
2882                 { fitbv } { /FitBV ~ @xpos }
2883                 { fith } { /FitH ~ @ypos }
2884                 { fitv } { /FitV ~ @xpos }
2885                 { fitr } { /Fit }
2886             }
2887             { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2888         ]
2889     }

```

```

2890 }
2891 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2892 {
2893   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2894   { \dim_eval:n {#2} } {#1} {#3} {#4}
2895 }
2896 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2897 {
2898   \vbox_to_zero:n
2899   {
2900     \__kernel_kern:n {#4}
2901     \hbox:n
2902     {
2903       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2904       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2905     }
2906     \tex_vss:D
2907   }
2908   \__kernel_kern:n {#1}
2909   \vbox_to_zero:n
2910   {
2911     \__kernel_kern:n { -#3 }
2912     \hbox:n
2913     {
2914       \__pdf_backend:n
2915       {
2916         dest ~ (#2)
2917         [
2918           @thispage
2919           /FitR ~
2920           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2921           @xpos ~ @ypos
2922         ]
2923       }
2924     }
2925     \tex_vss:D
2926   }
2927   \__kernel_kern:n { -#1 }
2928 }

```

(End definition for __pdf_backend_destination:nn, __pdf_backend_destination:nnnn, and __pdf_backend_destination_aux:nnnn.)

6.4.4 Structure

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

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2929 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2930 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2931 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2932 {
2933   \bool_if:nF {#1}
2934   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2935 }

```

(End definition for __pdf_backend_compresslevel:n and __pdf_backend_compress_objects:n.)

_pdf_backend_version_major_gset:n We start with the assumption that the default is active.

_pdf_backend_version_minor_gset:n

```

2936 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2937 {
2938   \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
2939   \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
2940 }
2941 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2942 {
2943   \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
2944   \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
2945 }

```

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

_pdf_backend_version_major: We start with the assumption that the default is active.

_pdf_backend_version_minor:

```

2946 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2947 \cs_new:Npn \__pdf_backend_version_minor: { 5 }

```

(End definition for __pdf_backend_version_major: and __pdf_backend_version_minor:.)

6.4.5 Marked content

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

_pdf_backend_emc:

```

2948 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2949 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2950 \cs_new_protected:Npn \__pdf_backend_emc:
2951 { \__kernel_backend_literal_page:n { EMC } }

```

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

2952 </dviPDFmx | xetex>

6.5 dvisvgm backend

2953 <*dvisvgm>

6.5.1 Catalogue entries

_pdf_backend_catalog_gput:nn No-op.

__pdf_backend_info_gput:nn

```

2954 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2 { }
2955 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2 { }

```

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

6.5.2 Objects

_pdf_backend_object_new:nn All no-ops here.

__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

```

2956 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2 { }
2957 \cs_new:Npn \__pdf_backend_object_ref:n #1 { }
2958 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2 { }
2959 \cs_new_protected:Npn \__pdf_backend_object_write:nx #1#2 { }
2960 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2 { }
2961 \cs_new_protected:Npn \__pdf_backend_object_now:nx #1#2 { }
2962 \cs_new:Npn \__pdf_backend_object_last: { }
2963 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1 { }

```

(End definition for __pdf_backend_object_new:nn and others.)

6.5.3 Structure

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2964 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2965 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1 { }

(End definition for \__pdf_backend_compresslevel:n and \__pdf_backend_compress_objects:n.)

\__pdf_backend_version_major_gset:n
\__pdf_backend_version_minor_gset:n
2966 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2967 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1 { }

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

\__pdf_backend_version_major:
\__pdf_backend_version_minor:
2968 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2969 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

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

\__pdf_backend_bdc:nn
\__pdf_backend_emc:
2970 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2971 \cs_new_protected:Npn \__pdf_backend_emc: { }

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

2972 </dvisvgm>
2973 </package>

```

7 l3backend-opacity Implementation

```

2974 <*package>
2975 <@@=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.

```

2976 <*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.

```

\__opacity_backend_select:n
\__opacity_backend_select_aux:n
\__opacity_backend_fill:n
\__opacity_backend_stroke:n
\__opacity_backend:nnn
\__opacity_backend:xnn
2977 \cs_new_protected:Npn \__opacity_backend_select:n #1
2978 {
2979   \exp_args:Nx \__opacity_backend_select_aux:n
2980   { \fp_eval:n { min(max(0,#1),1) } }
2981 }
2982 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
2983 {
2984   \__opacity_backend:nnn {#1} { fill } { ca }

```

```

2985     \_opacity_backend:nnn {#1} { stroke } { CA }
2986   }
2987   \cs_new_protected:Npn \_opacity_backend_fill:n #1
2988   {
2989     \_opacity_backend:xnn
2990     { \fp_eval:n { min(max(0,#1),1) } }
2991     { fill }
2992     { ca }
2993   }
2994   \cs_new_protected:Npn \_opacity_backend_stroke:n #1
2995   {
2996     \_opacity_backend:xnn
2997     { \fp_eval:n { min(max(0,#1),1) } }
2998     { stroke }
2999     { CA }
3000   }
3001   \cs_new_protected:Npn \_opacity_backend:nnn #1#2#3
3002   {
3003     \_kernel_backend_postscript:n
3004     {
3005       product ~ (Ghostscript) ~ search
3006       {
3007         pop ~ pop ~ pop ~
3008         #1 ~ .set #2 constantalpha
3009       }
3010       {
3011         pop ~
3012         mark ~
3013         /#3 ~ #1
3014         /SetTransparency ~
3015         pdfmark
3016       }
3017     }
3018     ifelse
3019   }
3020   \cs_generate_variant:Nn \_opacity_backend:nnn { x }
(End definition for \_opacity_backend_select:n and others.)
3021 </dvips>
3022 <*dvipdfmx | luatex | pdftex | xetex>

```

\c_opacity_backend_stack_int Set up a stack.

```

3023 \cs_if_exist:NT \pdfmanagement_add:nnn
3024 {
3025   \_kernel_color_backend_stack_init:Nnn \c_opacity_backend_stack_int
3026   { page ~ direct } { /opacity 1 ~ gs }
3027   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3028   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3029 }
(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
3030 \tl_new:N \l_opacity_backend_fill_tl
3031 \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 3032 \cs_new_protected:Npn \__opacity_backend_select:n #1
\__opacity_backend_reset:      3033 {
                                3034   \exp_args:Nx \__opacity_backend_select_aux:n
                                3035   { \fp_eval:n { min(max(0,#1),1) } }
                                3036 }
                                3037 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
                                3038 {
                                3039   \tl_set:Nn \l__opacity_backend_fill_tl {#1}
                                3040   \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
                                3041   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
                                3042   { opacity #1 }
                                3043   { << /ca ~ #1 /CA ~ #1 >> }
                                3044   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
                                3045   { /opacity #1 ~ gs }
                                3046   \group_insert_after:N \__opacity_backend_reset:
                                3047 }
                                3048 \cs_if_exist:NF \pdfmanagement_add:nnn
                                3049 {
                                3050   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1 { }
                                3051 }
                                3052 \cs_new_protected:Npn \__opacity_backend_reset:
                                3053 { \__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 For separate fill and stroke, we need to work out if we need to do more work or if we can stick to a single setting.

```

\__opacity_backend_stroke:n 3054 \cs_new_protected:Npn \__opacity_backend_fill:n #1
\__opacity_backend_fillstroke:nn 3055 {
\__opacity_backend_fillstroke:xx 3056   \__opacity_backend_fill_stroke:xx
                                3057   { \fp_eval:n { min(max(0,#1),1) } }
                                3058   \l__opacity_backend_stroke_tl
                                3059 }
                                3060 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
                                3061 {
                                3062   \__opacity_backend_fill_stroke:xx
                                3063   \l__opacity_backend_fill_tl
                                3064   { \fp_eval:n { min(max(0,#1),1) } }
                                3065 }
                                3066 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
                                3067 {
                                3068   \str_if_eq:nnTF {#1} {#2}
                                3069   { \__opacity_backend_select_aux:n {#1} }
                                3070   {
                                3071     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
                                3072     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
                                3073     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
                                3074     { opacity.fill #1 }
                                3075     { << /ca ~ #1 >> }
                                3076     \pdfmanagement_add:nnn { Page / Resources / ExtGState }

```



```

3077         { opacity.stroke #1 }
3078         { << /CA ~ #2 >> }
3079         \_kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3080         { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3081         \group_insert_after:N \_opacity_backend_reset:
3082     }
3083 }
3084 \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.)

3085 </dvipdfmx | luatex | pdftex | xetex>
3086 <*dvipdfmx | xdvipdfmx>

```

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

```

3087 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
3088 {
3089     \cs_gset_protected:Npn \_opacity_backend_select_aux:n #1
3090     {
3091         \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3092         \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3093         \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3094         { opacity #1 }
3095         { << /ca ~ #1 /CA ~ #1 >> }
3096         \_kernel_backend_literal_pdf:n { /opacity #1 ~ gs }
3097     }
3098     \cs_gset_protected:Npn \_opacity_backend_fill_stroke:nn #1#2
3099     {
3100         \str_if_eq:nnTF {#1} {#2}
3101         { \_opacity_backend_select_aux:n {#1} }
3102         {
3103             \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3104             \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3105             \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3106             { opacity.fill #1 }
3107             { << /ca ~ #1 >> }
3108             \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3109             { opacity.stroke #1 }
3110             { << /CA ~ #2 >> }
3111             \_kernel_backend_literal_pdf:n
3112             { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3113         }
3114     }
3115 }

(End definition for \_opacity_backend_select:n.)

3116 </dvipdfmx | xdvipdfmx>
3117 <*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
3118 \cs_new_protected:Npn \_opacity_backend_select:n #1
3119 { \_opacity_backend:nn {#1} { } }

```

```

3120 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3121 { \__opacity_backend:nn {#1} { fill- } }
3122 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3123 { \__opacity_backend:nn { {#1} } { stroke- } }
3124 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3125 { \__kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

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

3126 </dvisvgm>
3127 </package>

```

8 l3backend-header Implementation

```

3128 <*dvips & header>

color.sc Empty definition for color at the top level.
3129 /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.

```

3130 TeXDict begin
3131 /TeXcolorseparation { setcolor } def
3132 end

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

```

pdf.globaldict A small global dictionary for backend use.

```

3133 true setglobal
3134 /pdf.globaldict 4 dict def
3135 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.

```

3136 /pdf.cvs { 65534 string cvs } def
3137 /pdf.dvi.pt { 72.27 mul Resolution div } def
3138 /pdf.pt.dvi { 72.27 div Resolution mul } def
3139 /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.

```

3140 /pdf.linkmargin { 1 pdf.pt.dvi } def
3141 /pdf.linkdp.pad { 0 } def
3142 /pdf.linkht.pad { 0 } def

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

```

<code>pdf.rect</code>	Functions for marking the limits of an annotation/link, plus drawing the border. We
<code>pdf.save.ll</code>	separate links for generic annotations to support adding a margin and setting a minimal
<code>pdf.save.ur</code>	size.
<code>pdf.save.linkll</code>	3143 <code>/pdf.rect</code>
<code>pdf.save.linkur</code>	3144 <code>{ /Rect [pdf.llx pdf.lly pdf.urx pdf.ury] } def</code>
<code>pdf.llx</code>	3145 <code>/pdf.save.ll</code>
<code>pdf.lly</code>	3146 <code>{</code>
<code>pdf.urx</code>	3147 <code>currentpoint</code>
<code>pdf.ury</code>	3148 <code>/pdf.lly exch def</code>
	3149 <code>/pdf.llx exch def</code>
	3150 <code>}</code>
	3151 <code>def</code>
	3152 <code>/pdf.save.ur</code>
	3153 <code>{</code>
	3154 <code>currentpoint</code>
	3155 <code>/pdf.ury exch def</code>
	3156 <code>/pdf.urx exch def</code>
	3157 <code>}</code>
	3158 <code>def</code>
	3159 <code>/pdf.save.linkll</code>
	3160 <code>{</code>
	3161 <code>currentpoint</code>
	3162 <code>pdf.linkmargin add</code>
	3163 <code>pdf.linkdp.pad add</code>
	3164 <code>/pdf.lly exch def</code>
	3165 <code>pdf.linkmargin sub</code>
	3166 <code>/pdf.llx exch def</code>
	3167 <code>}</code>
	3168 <code>def</code>
	3169 <code>/pdf.save.linkur</code>
	3170 <code>{</code>
	3171 <code>currentpoint</code>
	3172 <code>pdf.linkmargin sub</code>
	3173 <code>pdf.linkht.pad sub</code>
	3174 <code>/pdf.ury exch def</code>
	3175 <code>pdf.linkmargin add</code>
	3176 <code>/pdf.urx exch def</code>
	3177 <code>}</code>
	3178 <code>def</code>
	(End definition for <code>pdf.rect</code> and others. These functions are documented on page ??.)
<code>pdf.dest.anchor</code>	For finding the anchor point of a destination link. We make the use case a separate
<code>pdf.dest.x</code>	function as it comes up a lot, and as this makes it easier to adjust if we need additional
<code>pdf.dest.y</code>	effects. We also need a more complex approach to convert a co-ordinate pair correctly
<code>pdf.dest.point</code>	when defining a rectangle: this can otherwise be out when using a landscape page.
<code>pdf.dest2device</code>	(Thanks to Alexander Grahn for the approach here.)
<code>pdf.dev.x</code>	3179 <code>/pdf.dest.anchor</code>
<code>pdf.dev.y</code>	3180 <code>{</code>
<code>pdf.tmpa</code>	3181 <code>currentpoint exch</code>
<code>pdf.tmpb</code>	3182 <code>pdf.dvi.pt 72 add</code>
<code>pdf.tmpc</code>	3183 <code>/pdf.dest.x exch def</code>
<code>pdf.tmpd</code>	3184 <code>pdf.dvi.pt</code>
	3185 <code>vsize 72 sub exch sub</code>

```

3186     /pdf.dest.y exch def
3187   }
3188   def
3189 /pdf.dest.point
3190 { pdf.dest.x pdf.dest.y } def
3191 /pdf.dest2device
3192 {
3193   /pdf.dest.y exch def
3194   /pdf.dest.x exch def
3195   matrix currentmatrix
3196   matrix defaultmatrix
3197   matrix invertmatrix
3198   matrix concatmatrix
3199   cvx exec
3200   /pdf.dev.y exch def
3201   /pdf.dev.x exch def
3202   /pdf.tmpd exch def
3203   /pdf.tmpc exch def
3204   /pdf.tmpb exch def
3205   /pdf.tmpa exch def
3206   pdf.dest.x pdf.tmpa mul
3207     pdf.dest.y pdf.tmpc mul add
3208     pdf.dev.x add
3209   pdf.dest.x pdf.tmpb mul
3210     pdf.dest.y pdf.tmpd mul add
3211     pdf.dev.y add
3212 }
3213 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	3214 /pdf.bordertracking false def
pdf.brokenlink.skip	3215 /pdf.bordertracking.begin
pdf.brokenlink.dict	3216 {
pdf.bordertracking.endpage	3217 SDict /pdf.bordertracking true put
pdf.bordertracking.continue	3218 SDict /pdf.leftboundary undef
pdf.originx	3219 SDict /pdf.rightboundary undef
pdf.originy	3220 /a where
	3221 {
	3222 /a
	3223 {
	3224 currentpoint pop
	3225 SDict /pdf.rightboundary known dup
	3226 {
	3227 SDict /pdf.rightboundary get 2 index lt
	3228 { not }
	3229 if
	3230 }
	3231 if
	3232 { pop }

```

3233         { SDict exch /pdf.rightboundary exch put }
3234     ifelse
3235     moveto
3236     currentpoint pop
3237     SDict /pdf.leftboundary known dup
3238     {
3239         SDict /pdf.leftboundary get 2 index gt
3240         { not }
3241         if
3242     }
3243     if
3244     { pop }
3245     { SDict exch /pdf.leftboundary exch put }
3246     ifelse
3247 }
3248 put
3249 }
3250 if
3251 }
3252 def
3253 /pdf.bordertracking.end
3254 {
3255     /a where { /a { moveto } put } if
3256     /x where { /x { 0 exch rmoveto } put } if
3257     SDict /pdf.leftboundary known
3258     { pdf.outerbox 0 pdf.leftboundary put }
3259     if
3260     SDict /pdf.rightboundary known
3261     { pdf.outerbox 2 pdf.rightboundary put }
3262     if
3263     SDict /pdf.bordertracking false put
3264 }
3265 def
3266 /pdf.bordertracking.endpage
3267 {
3268     pdf.bordertracking
3269     {
3270         pdf.bordertracking.end
3271         true setglobal
3272         pdf.globaldict
3273         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3274         pdf.globaldict
3275         /pdf.brokenlink.skip pdf.baselineskip put
3276         pdf.globaldict
3277         /pdf.brokenlink.dict
3278         pdf.link.dict pdf.cvs put
3279         false setglobal
3280         mark pdf.link.dict cvx exec /Rect
3281         [
3282             pdf.llx
3283             pdf.lly
3284             pdf.outerbox 2 get pdf.linkmargin add
3285             currentpoint exch pop
3286             pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub

```

```

3287     ]
3288     /ANN pdf.pdfmark
3289   }
3290   if
3291 }
3292   def
3293 /pdf.bordertracking.continue
3294 {
3295   /pdf.link.dict pdf.globaldict
3296   /pdf.brokenlink.dict get def
3297   /pdf.outerbox pdf.globaldict
3298   /pdf.brokenlink.rect get def
3299   /pdf.baselineskip pdf.globaldict
3300   /pdf.brokenlink.skip get def
3301   pdf.globaldict dup dup
3302   /pdf.brokenlink.dict undef
3303   /pdf.brokenlink.skip undef
3304   /pdf.brokenlink.rect undef
3305   currentpoint
3306   /pdf.originy exch def
3307   /pdf.originx exch def
3308   /a where
3309   {
3310     /a
3311     {
3312       moveto
3313       SDict
3314       begin
3315         currentpoint pdf.originy ne exch
3316         pdf.originx ne or
3317         {
3318           pdf.save.linkll
3319           /pdf.lly
3320           pdf.lly pdf.outerbox 1 get sub def
3321           pdf.bordertracking.begin
3322         }
3323         if
3324         end
3325       }
3326       put
3327     }
3328   if
3329   /x where
3330   {
3331     /x
3332     {
3333       0 exch rmoveto
3334       SDict
3335       begin
3336         currentpoint
3337         pdf.originy ne exch pdf.originx ne or
3338         {
3339           pdf.save.linkll
3340           /pdf.lly

```

```

3341         pdf.lly pdf.outerbox 1 get sub def
3342         pdf.bordertracking.begin
3343     }
3344     if
3345     end
3346 }
3347 put
3348 }
3349 if
3350 }
3351 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
pdf.breaklink.write in the dictionary, looping over key-value pairs. The first line is handled first, adjusting
pdf.count the rectangle to stay inside the text area. The second phase is a loop over the height of
pdf.currentrect 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.

```

3352 /pdf.breaklink
3353 {
3354     pop
3355     counttomark 2 mod 0 eq
3356     {
3357         counttomark /pdf.count exch def
3358         {
3359             pdf.count 0 eq { exit } if
3360             counttomark 2 roll
3361             1 index /Rect eq
3362             {
3363                 dup 4 array copy
3364                 dup dup
3365                 1 get
3366                 pdf.outerbox pdf.rect.ht
3367                 pdf.linkmargin 2 mul add sub
3368                 3 exch put
3369                 dup
3370                 pdf.outerbox 2 get
3371                 pdf.linkmargin add
3372                 2 exch put
3373                 dup dup
3374                 3 get
3375                 pdf.outerbox pdf.rect.ht
3376                 pdf.linkmargin 2 mul add add
3377                 1 exch put
3378             } /pdf.currentrect exch def
3379             pdf.breaklink.write
3380             {
3381                 pdf.currentrect
3382                 dup
3383                 pdf.outerbox 0 get
3384                 pdf.linkmargin sub
3385                 0 exch put
3386             }

```

```

3387         pdf.outerbox 2 get
3388         pdf.linkmargin add
3389         2 exch put
3390     dup dup
3391         1 get
3392         pdf.baselineskip add
3393         1 exch put
3394     dup dup
3395         3 get
3396         pdf.baselineskip add
3397         3 exch put
3398     /pdf.currentrect exch def
3399     pdf.breaklink.write
3400 }
3401 1 index 3 get
3402 pdf.linkmargin 2 mul add
3403 pdf.outerbox pdf.rect.ht add
3404 2 index 1 get sub
3405 pdf.baselineskip div round cvi 1 sub
3406 exch
3407 repeat
3408 pdf.currentrect
3409 dup
3410     pdf.outerbox 0 get
3411     pdf.linkmargin sub
3412     0 exch put
3413 dup dup
3414     1 get
3415     pdf.baselineskip add
3416     1 exch put
3417 dup dup
3418     3 get
3419     pdf.baselineskip add
3420     3 exch put
3421 dup 2 index 2 get 2 exch put
3422 /pdf.currentrect exch def
3423 pdf.breaklink.write
3424 SDict /pdf.pdfmark.good false put
3425 exit
3426 }
3427 { pdf.count 2 sub /pdf.count exch def }
3428 ifelse
3429 }
3430 loop
3431 }
3432 if
3433 /ANN
3434 }
3435 def
3436 /pdf.breaklink.write
3437 {
3438     counttomark 1 sub
3439     index /_objdef eq
3440     {

```



```

3441         counttomark -2 roll
3442         dup wcheck
3443         {
3444             readonly
3445             counttomark 2 roll
3446         }
3447         { pop pop }
3448         ifelse
3449     }
3450     if
3451     counttomark 1 add copy
3452     pop pdf.currentrect
3453     /ANN pdfmark
3454 }
3455 def

```

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

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

```

3456 /pdf.pdfmark
3457 {
3458     SDict /pdf.pdfmark.good true put
3459     dup /ANN eq
3460     {
3461         pdf.pdfmark.store
3462         pdf.pdfmark.dict
3463         begin
3464             Subtype /Link eq
3465             currentdict /Rect known and
3466             SDict /pdf.outerbox known and
3467             SDict /pdf.baselineskip known and
3468             {
3469                 Rect 3 get
3470                 pdf.linkmargin 2 mul add
3471                 pdf.outerbox pdf.rect.ht add
3472                 Rect 1 get sub
3473                 pdf.baselineskip div round cvi 0 gt
3474                 { pdf.breaklink }
3475                 if
3476             }
3477             if
3478         end
3479         SDict /pdf.outerbox undef
3480         SDict /pdf.baselineskip undef
3481         currentdict /pdf.pdfmark.dict undef
3482     }
3483     if
3484     pdf.pdfmark.good
3485     { pdfmark }
3486     { cleartomark }

```

```

3487     ifelse
3488   }
3489   def
3490 /pdf.pdfmark.store
3491 {
3492   /pdf.pdfmark.dict 65534 dict def
3493   counttomark 1 add copy
3494   pop
3495   {
3496     dup mark eq
3497     {
3498       pop
3499       exit
3500     }
3501     {
3502       pdf.pdfmark.dict
3503       begin def end
3504     }
3505     ifelse
3506   }
3507   loop
3508 }
3509 def

```

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

```

3510 </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_aux:Nn</code> ...	235, 283, 340
A			
<code>\AtBeginDvi</code>	57	<code>_box_backend_scale:Nnn</code>	252, 311, 355, 432
B			
bool commands:			
<code>\bool_gset_false:N</code>	1169, 1188, 1211, 1233, 1249, 1350, 1587, 1623, 2191, 2237	<code>\l_box_backend_sin_fp</code>	283
<code>\bool_gset_true:N</code>	1167, 1236, 1348, 1602, 2184, 2190	<code>\g_box_clip_path_int</code>	369
<code>\bool_if:NTF</code>	67, 675, 1179, 1183, 1199, 1202, 1206, 1217, 1224, 1228, 1240, 1244, 1361, 1366, 1371, 1561, 1606, 1719, 1754, 1864, 1906, 2179, 2194, 2199, 2204	C	
<code>\bool_if:nTF</code>	2413, 2679, 2933	char commands:	
<code>\bool_lazy_or:nnTF</code>	1746, 1899	<code>\char_set_catcode_space:n</code>	154
<code>\bool_new:N</code>	1170, 1237, 1351, 1603, 2164, 2165	clist commands:	
<code>\bool_set_false:N</code>	1729, 1831, 1924, 1988	<code>\clist_map_function:nN</code> ...	1257, 1381
box commands:			
<code>\box_dp:N</code>	224, 226, 274, 276, 331, 333, 380, 382, 384, 386, 2216, 2249, 2250, 2275	<code>\clist_map_function:nn</code>	1630
<code>\box_ht:N</code>	226, 276, 333, 384, 386, 1766, 1961, 2221, 2260, 2261, 2277	color internal commands:	
<code>\box_if_empty:NTF</code>	2310	<code>_color_backend:nnn</code>	1055
<code>\box_move_down:nn</code>	2138, 2216	<code>_color_backend_cmyk:w</code>	1056
<code>\box_move_up:nn</code>	2140, 2221	<code>_color_backend_devicen_init:n</code>	903
<code>\box_new:N</code>	2023, 2128, 2129	<code>_color_backend_devicen_-init:nnn</code>	821, 903
<code>\box_set_dp:Nn</code>	1686	<code>_color_backend_devicen_init:w</code>	903
<code>\box_set_ht:Nn</code>	1685	<code>_color_backend_fill:n</code>	962, 989, 1019, 1037, 1044
<code>\box_set_wd:Nn</code>	288, 1684	<code>_color_backend_fill_cmyk:n</code>	962, 996, 1019, 1044
<code>\box_use:N</code>	231, 249, 263, 279, 306, 320, 336, 352, 364, 415, 429, 448, 1301, 1496, 1687, 2169	<code>_color_backend_fill_devicen:nn</code>	988, 1010, 1036, 1106
<code>\box_wd:N</code>	225, 233, 275, 281, 332, 338, 381, 383, 1765, 1960	<code>_color_backend_fill_gray:n</code>	962, 996, 1019, 1044
box internal commands:			
<code>_box_backend_clip:N</code>	213, 268, 325, 369	<code>_color_backend_fill_rgb:n</code>	962, 996, 1019, 1044
<code>\l_box_backend_cos_fp</code>	283	<code>_color_backend_fill_separation:nn</code>	988, 996, 1036, 1106
<code>_box_backend_rotate:Nn</code>	235, 283, 340, 419	<code>\l_color_backend_fill_tl</code>	637, 647, 970, 985
		<code>\c_color_backend_main_stack_int</code>	516
		<code>_color_backend_pickup:N</code> ..	456, 479
		<code>_color_backend_pickup:w</code> ..	14, 456, 479
		<code>_color_backend_reset:</code>	619, 639, 656, 973, 986, 996, 1028, 1053
		<code>_color_backend_rgb:w</code>	1079
		<code>_color_backend_select:n</code> ..	619, 671
		<code>_color_backend_select:nn</code> ..	639, 848
		<code>_color_backend_select_cmyk:n</code> ..	619, 639, 656
		<code>_color_backend_select_devicen:nn</code>	670, 841, 847, 954
		<code>_color_backend_select_gray:n</code> ..	619, 639, 656

_color_backend_select_rgb:n . . .	\l_color_backend_stroke_tl
..... 619 , 639 , 656 637 , 648 , 972 , 983
_color_backend_select_separation:nn	\g_color_model_int 691 , 827 , 872 , 940
..... 670 , 841 , 847 , 954	\c_color_model_range_CIELAB_tl .
_color_backend_separation- 782 , 817 , 892 , 899
init:n 673 , 850 , 927 , 951	color.sc 619 , 3129
_color_backend_separation-	cs commands:
init:nnn 673	\cs_generate_variant:Nn . . 49 , 63 ,
_color_backend_separation-	66 , 99 , 138 , 143 , 170 , 201 , 207 , 569 ,
init:nnnn 673	606 , 684 , 1116 , 1311 , 1505 , 1878 ,
_color_backend_separation-	1935 , 1951 , 2027 , 2064 , 2123 , 2615 ,
init:nnnnn 673 , 843 , 850	2643 , 2753 , 2775 , 2810 , 3020 , 3084
_color_backend_separation-	\cs_gset:Npx . . 2425 , 2429 , 2938 , 2943
init:nw 673	\cs_gset_eq:NN 663 ,
_color_backend_separation-	664 , 957 , 1003 , 1004 , 1010 , 1012 , 1014
init:w 673	\cs_gset_protected:Npn
_color_backend_separation- 551 , 658 , 665 , 956 ,
init_/DeviceCMYK:nnn 673	998 , 1005 , 1007 , 1009 , 3050 , 3089 , 3098
_color_backend_separation-	\cs_if_exist:NTF
init_/DeviceGray:nnn 673 27 , 50 , 457 , 480 , 539 ,
_color_backend_separation-	866 , 934 , 2306 , 2704 , 2730 , 3023 , 3048
init_/DeviceRGB:nnn 673	\cs_if_exist_use:NTF 38 , 697
_color_backend_separation-	\cs_new:Npn 706 , 708 , 710 ,
init_aux:nnnnn 673	712 , 719 , 725 , 727 , 733 , 750 , 757 ,
_color_backend_separation-	759 , 945 , 1262 , 1386 , 1634 , 1964 ,
init_CIELAB:nnn 673 , 843 , 850	1973 , 2017 , 2042 , 2124 , 2126 , 2159 ,
_color_backend_separation-	2331 , 2431 , 2432 , 2585 , 2616 , 2617 ,
init_CIELAB:nnnnnn 844	2735 , 2768 , 2811 , 2813 , 2829 , 2946 ,
_color_backend_separation-	2947 , 2957 , 2962 , 2963 , 2968 , 2969
init_count:n 673	\cs_new:Npx
_color_backend_separation-	.. 2452 , 2487 , 2644 , 2655 , 2722 , 2859
init_count:w 673	\cs_new_eq:NN
_color_backend_separation- 46 , 57 , 59 , 672 , 849 , 951 ,
init_Device:Nn 673	992 , 993 , 1040 , 1041 , 1108 , 1109 ,
\g_color_backend_stack_int . . . 516	1115 , 1310 , 1316 , 1317 , 1504 , 1511 ,
\l_color_backend_stack_int	1696 , 1725 , 1776 , 1777 , 1819 , 1827 ,
..... 513 , 541 , 547 , 649 , 653 , 971 , 984	1849 , 1920 , 1977 , 1984 , 2016 , 2169
_color_backend_stroke:n	\cs_new_protected:Npn . . . 47 , 54 ,
..... 962 , 991 , 996	61 , 64 , 72 , 78 , 83 , 85 , 89 , 100 , 110 ,
_color_backend_stroke_cmyk:n . .	119 , 128 , 141 , 144 , 146 , 148 , 168 ,
..... 962 , 1019 , 1055	173 , 182 , 192 , 202 , 213 , 235 , 237 ,
_color_backend_stroke_cmyk:w 1055	252 , 268 , 283 , 285 , 311 , 325 , 340 ,
_color_backend_stroke_devicen:nn	342 , 355 , 369 , 419 , 432 , 456 , 474 ,
..... 988 , 1014 , 1036 , 1106	479 , 487 , 517 , 560 , 570 , 582 , 596 ,
_color_backend_stroke_gray:n . .	607 , 619 , 621 , 623 , 625 , 633 , 639 ,
..... 962 , 1019 , 1055	641 , 643 , 645 , 652 , 670 , 685 , 775 ,
_color_backend_stroke_gray-	821 , 841 , 842 , 843 , 844 , 847 , 850 ,
aux:n 1055	877 , 881 , 903 , 962 , 964 , 966 , 968 ,
_color_backend_stroke_rgb:n . . .	975 , 977 , 979 , 981 , 988 , 990 , 1019 ,
..... 962 , 1019 , 1055	1021 , 1023 , 1025 , 1030 , 1032 , 1034 ,
_color_backend_stroke_rgb:w . 1055	1036 , 1038 , 1044 , 1046 , 1048 , 1050 ,
_color_backend_stroke_separation:nn	1055 , 1057 , 1068 , 1076 , 1078 , 1080 ,
..... 988 , 996 , 1036 , 1106	1106 , 1107 , 1117 , 1122 , 1127 , 1129 ,
	1131 , 1139 , 1147 , 1156 , 1166 , 1168 ,

1171, 1173, 1190, 1195, 1213, 1235,
 1238, 1251, 1264, 1269, 1271, 1273,
 1275, 1277, 1279, 1281, 1283, 1288,
 1312, 1314, 1318, 1323, 1328, 1338,
 1347, 1349, 1352, 1354, 1356, 1358,
 1363, 1368, 1373, 1375, 1388, 1393,
 1395, 1397, 1399, 1401, 1403, 1405,
 1407, 1418, 1443, 1455, 1467, 1479,
 1486, 1506, 1512, 1517, 1522, 1533,
 1543, 1553, 1555, 1557, 1559, 1590,
 1592, 1597, 1599, 1601, 1604, 1625,
 1636, 1649, 1651, 1653, 1655, 1657,
 1659, 1661, 1663, 1665, 1673, 1697,
 1711, 1726, 1738, 1743, 1771, 1783,
 1796, 1806, 1821, 1828, 1836, 1847,
 1851, 1854, 1869, 1879, 1914, 1921,
 1927, 1933, 1936, 1943, 1952, 1957,
 1965, 1978, 1985, 1991, 1993, 1995,
 2006, 2025, 2028, 2030, 2034, 2044,
 2065, 2070, 2075, 2080, 2090, 2095,
 2103, 2131, 2136, 2168, 2170, 2175,
 2177, 2182, 2197, 2202, 2239, 2268,
 2287, 2296, 2333, 2340, 2366, 2371,
 2399, 2411, 2423, 2427, 2433, 2435,
 2439, 2463, 2465, 2467, 2478, 2498,
 2508, 2531, 2545, 2555, 2566, 2587,
 2618, 2666, 2677, 2683, 2711, 2745,
 2747, 2754, 2756, 2760, 2770, 2776,
 2781, 2786, 2791, 2793, 2795, 2803,
 2816, 2832, 2834, 2857, 2867, 2869,
 2891, 2896, 2929, 2931, 2936, 2941,
 2948, 2950, 2954, 2955, 2956, 2958,
 2959, 2960, 2961, 2964, 2965, 2966,
 2967, 2970, 2971, 2977, 2982, 2987,
 2994, 3001, 3032, 3037, 3052, 3054,
 3060, 3066, 3118, 3120, 3122, 3124
 \cs_new_protected:Npx
 520, 673, 1091, 2694, 2751, 2836
 \cs_set:Npn 152
 \cs_set_eq:NN 2327, 2328
 \cs_set_protected:Npn 459, 482

D

dim commands:

\dim_eval:n 2134, 2369,
 2447, 2448, 2449, 2506, 2541, 2542,
 2543, 2823, 2824, 2825, 2868, 2894
 \dim_max:nn 2247, 2258
 \dim_set:Nn ... 1765, 1766, 1960, 1961
 \dim_to_decimal:n .. 380, 381, 382,
 383, 384, 386, 1515, 1520, 1526,
 1527, 1528, 1529, 1538, 1539, 1540,
 1631, 1650, 2011, 2012, 2245, 2256,
 2274, 2275, 2276, 2277, 2281, 2337

\dim_to_decimal_in_bp:n
 224, 225, 226, 274, 275, 276,
 331, 332, 333, 1135, 1136, 1143,
 1144, 1151, 1152, 1160, 1161, 1162,
 1259, 1263, 1267, 1321, 1326, 1332,
 1333, 1334, 1342, 1343, 1383, 1387,
 1391, 1635, 1702, 1703, 1704, 1705,
 1841, 1842, 1843, 1844, 1893, 1894,
 1895, 1896, 2000, 2001, 2002, 2003
 draw internal commands:
 __draw_align_currentpoint... .. 35
 __draw_backend_add_to_path:n ...
 1512, 1558
 __draw_backend_begin:
 1117, 1312, 1506
 __draw_backend_box_use:Nnnnn ...
 30, 1288, 1486, 1673
 __draw_backend_cap_but:
 1251, 1375, 1625
 __draw_backend_cap_rectangle: ..
 1251, 1375, 1625
 __draw_backend_cap_round:
 1251, 1375, 1625
 __draw_backend_clip: 1171, 1352, 1557
 __draw_backend_closepath:
 1171, 1352, 1557
 __draw_backend_closestroke: ...
 1171, 1352, 1557
 __draw_backend_cm:nnnn 1283, 1296,
 1297, 1298, 1407, 1490, 1665, 1676
 __draw_backend_cm_aux:nnnn .. 1407
 __draw_backend_cm_decompose:nnnnN
 1413, 1442
 __draw_backend_cm_decompose_-
 auxi:nnnnN 1442
 __draw_backend_cm_decompose_-
 auxii:nnnnN 1442
 __draw_backend_cm_decompose_-
 auxiii:nnnnN 1442
 __draw_backend_curveto:nnnnnn ..
 1131, 1318, 1512
 __draw_backend_dash:n
 1251, 1375, 1625
 __draw_backend_dash_aux:nn .. 1625
 __draw_backend_dash_pattern:nn ..
 1251, 1375, 1625
 __draw_backend_discardpath: ...
 1171, 1352, 1557
 __draw_backend_end: 1117, 1312, 1506
 __draw_backend_evenodd_rule: ...
 1166, 1347, 1553
 __draw_backend_fill: 1171, 1352, 1557
 __draw_backend_fillstroke:
 1171, 1352, 1557

<code>__draw_backend_join_bevel:</code>	
. 1251 , 1375 , 1625	
<code>__draw_backend_join_miter:</code>	
. 1251 , 1375 , 1625	
<code>__draw_backend_join_round:</code>	
. 1251 , 1375 , 1625	
<code>__draw_backend_lineto:nn</code>	
. 1131 , 1318 , 1512	
<code>__draw_backend_linewidth:n</code>	
. 1251 , 1375 , 1625	
<code>__draw_backend_literal:n</code>	
. 1115 , 1120 , 1124 , 1128 ,	
1130 , 1133 , 1141 , 1149 , 1158 , 1172 ,	
1175 , 1176 , 1177 , 1178 , 1181 , 1187 ,	
1197 , 1204 , 1210 , 1215 , 1220 , 1221 ,	
1222 , 1223 , 1226 , 1232 , 1242 , 1248 ,	
1253 , 1266 , 1270 , 1272 , 1274 , 1276 ,	
1278 , 1280 , 1282 , 1285 , 1290 , 1291 ,	
1292 , 1293 , 1294 , 1295 , 1299 , 1300 ,	
1302 , 1303 , 1304 , 1305 , 1306 , 1310 ,	
1320 , 1325 , 1330 , 1340 , 1353 , 1355 ,	
1357 , 1360 , 1365 , 1370 , 1374 , 1377 ,	
1390 , 1394 , 1396 , 1398 , 1400 , 1402 ,	
1404 , 1406 , 1504 , 1564 , 1583 , 1609	
<code>__draw_backend_miterlimit:n</code> . . .	
. 1251 , 1375 , 1625	
<code>__draw_backend_moveto:nn</code>	
. 1131 , 1318 , 1512	
<code>__draw_backend_nonzero_rule:</code> . . .	
. 1166 , 1347 , 1553	
<code>__draw_backend_path:n</code> 1557	
<code>__draw_backend_rectangle:nnnn</code> . .	
. 1131 , 1318 , 1512	
<code>__draw_backend_scope:n</code> 1554 , 1556 ,	
1576 , 1616 , 1638 , 1650 , 1652 , 1654 ,	
1656 , 1658 , 1660 , 1662 , 1664 , 1667	
<code>__draw_backend_scope_begin:</code> . . .	
. 1127 , 1313 , 1316	
<code>__draw_backend_scope_end:</code>	
. 1127 , 1315 , 1316	
<code>__draw_backend_stroke:</code>	
. 1171 , 1352 , 1557	
<code>\g__draw_clip_path_int</code>	
. . . 1563 , 1566 , 1579 , 1608 , 1611 , 1619	
<code>\g__draw_draw_clip_bool</code> . . . 1171 , 1557	
<code>\g__draw_draw_eor_bool</code>	
. . . 1166 , 1183 , 1199 , 1206 , 1217 ,	
1228 , 1244 , 1347 , 1361 , 1366 , 1371	
<code>\g__draw_draw_path_int</code> 1557	
<code>\g__draw_draw_path_tl</code>	
. . . 1512 , 1568 , 1584 , 1586 , 1613 , 1622	
<code>\g__draw_path_int</code> 1572 , 1589	
	E
	<code>\errmessage</code> 38
	<code>\evensidemargin</code> 2214
	exp commands:
	<code>\exp_after:wN</code> 159 , 465 , 1971
	<code>\exp_args:Ne</code> 721 , 2368 , 2893
	<code>\exp_args:Nf</code> 1256 , 1380 , 2133
	<code>\exp_args:NNf</code> 236 , 284 , 341
	<code>\exp_args:Nnx</code> 2120 , 2806
	<code>\exp_args:NV</code> 461
	<code>\exp_args:Nx</code> 1789 , 1810 ,
	2077 , 2092 , 2210 , 2772 , 2979 , 3034
	<code>\exp_last_unbraced:Nx</code> 470 , 484
	<code>\exp_not:N</code> 522 , 523 ,
	531 , 533 , 679 , 2454 , 2456 , 2459 ,
	2489 , 2491 , 2494 , 2646 , 2648 , 2651 ,
	2657 , 2659 , 2662 , 2699 , 2700 , 2706 ,
	2707 , 2726 , 2731 , 2840 , 2848 , 2864
	<code>\exp_not:n</code> . . . 48 , 97 , 108 , 136 , 2068 ,
	2073 , 2362 , 2601 , 2602 , 2616 , 2617 ,
	2629 , 2630 , 2784 , 2789 , 2800 , 2873
	<code>\ExplBackendFileDate</code> 1
	F
	file commands:
	<code>\file_compare_timestamp:nNnTF</code> . 1798
	<code>\file_parse_full_name:nNNN</code> 1785 , 1808
	<code>\fmtversion</code> 52
	fp commands:
	<code>\fp_compare:nNnTF</code>
	. . . 243 , 290 , 296 , 348 , 1423 , 1436 , 1481
	<code>\fp_eval:n</code> . . . 236 , 245 , 258 , 259 , 284 ,
	301 , 316 , 318 , 341 , 350 , 361 , 362 ,
	426 , 441 , 442 , 1063 , 1064 , 1065 ,
	1073 , 1086 , 1087 , 1088 , 1425 , 1430 ,
	1431 , 1438 , 1448 , 1449 , 1450 , 1451 ,
	1460 , 1461 , 1462 , 1463 , 1472 , 1473 ,
	1474 , 1475 , 2359 , 2528 , 2887 , 2980 ,
	2990 , 2997 , 3035 , 3057 , 3064 , 3125
	<code>\fp_new:N</code> 309 , 310
	<code>\fp_set:Nn</code> 289 , 292
	<code>\fp_use:N</code> 295 , 299 , 304
	<code>\fp_zero:N</code> 291
	<code>\c_zero_fp</code> 243 , 290 , 296 , 348 , 1423 , 1436
	G
	graphics commands:
	<code>\graphics_bb_restore:nTF</code> . 1740 , 1954
	<code>\graphics_bb_save:n</code> 1769 , 1962
	<code>\l_graphics_decodearray_tl</code>
 1717 , 1718 ,
	1728 , 1748 , 1752 , 1753 , 1830 , 1862 ,
	1863 , 1901 , 1904 , 1905 , 1923 , 1987

```

\graphics_extract_bb:n .
    . . . . . 1825, 1832, 1982, 1989
\l_graphics_interpolate_bool . . .
    . . . . . 1719, 1729, 1747, 1754,
    1831, 1864, 1900, 1906, 1924, 1988
\l_graphics_llx_dim . . . . .
    . . . . . 1702, 1841, 1893, 2000
\l_graphics_lly_dim . . . . .
    . . . . . 1703, 1842, 1894, 2001
\l_graphics_name_tl . . . . . 1803
\l_graphics_page_int . . . . .
    . . . . . 1713, 1733, 1734, 1758,
    1759, 1823, 1860, 1861, 1887, 1888,
    1916, 1929, 1930, 1969, 1970, 1980
\l_graphics_pagebox_tl . . . . .
    . . . . . 51, 1714, 1732,
    1760, 1761, 1824, 1858, 1859, 1889,
    1891, 1917, 1938, 1939, 1971, 1981
\graphics_read_bb:n . 1696, 1819, 1977
\l_graphics_urx_dim . . . . .
    . . 1704, 1765, 1843, 1895, 1960, 2002
\l_graphics_ury_dim . . 1705, 1766,
    1844, 1896, 1961, 2003, 2011, 2012
graphics internal commands:
\l_graphics_backend_dir_str . 1778
\l_graphics_backend_ext_str . 1778
\l_graphics_backend_getbb_auxi:n
    . . . . . 1711
\l_graphics_backend_getbb-
    auxi:nN . . . . . 1914
\l_graphics_backend_getbb-
    auxii:n . . . . . 1711
\l_graphics_backend_getbb-
    auxii:nnN . . . . . 1914
\l_graphics_backend_getbb-
    auxiii:nNnn . . . . . 1914
\l_graphics_backend_getbb-
    auxiv:nnNnn . . . . . 1914
\l_graphics_backend_getbb-
    auxv:nNnn . . . . . 1914
\l_graphics_backend_getbb-
    auxvi:nNnn . . . . . 1955, 1957
\l_graphics_backend_getbb_eps:n .
    . . . . . 1696, 1778, 1819, 1977
\l_graphics_backend_getbb_eps:nn
    . . . . . 1778
\l_graphics_backend_getbb_eps:nn
    . . . . . 1789, 1796
\l_graphics_backend_getbb_jpg:n .
    . . . . . 1711, 1819, 1914, 1978
\l_graphics_backend_getbb-
    pagebox:w . . . . . 1914, 1971
\l_graphics_backend_getbb_pdf:n .
    . . . . . 1711, 1804, 1819, 1914, 1985
\l_graphics_backend_getbb_png:n .
    . . . . . 1711, 1819, 1914, 1978
\l_graphics_backend_include:nn 1991
\l_graphics_backend_include-
    auxi:nn . . . . . 1836
\l_graphics_backend_include-
    auxii:nnn . . . . . 1836
\l_graphics_backend_include-
    auxiii:nnn . . . . . 1836
\l_graphics_backend_include-
    bitmap_quote:w . . . . . 1965, 2006
\l_graphics_backend_include-
    eps:n . . . . . 1697, 1778, 1836, 1991
\l_graphics_backend_include-
    jpg:n . . . . . 1771, 1836, 2006
\l_graphics_backend_include-
    pdf:n . . 1771, 1810, 1836, 1965, 1991
\l_graphics_backend_include_pdf-
    quote:w . . . . . 1968, 1973
\l_graphics_backend_include-
    png:n . . . . . 1771, 1836, 2006
\l_graphics_backend_name_str . 1778
\l_graphics_graphics_attr_tl . .
    . . . . . 1710, 1715,
    1722, 1730, 1740, 1767, 1769, 1774
\l_graphics_internal_box . . . .
    . . 1763, 1765, 1766, 1959, 1960, 1961
\g_graphics_track_int . . . . .
    . . . . . 1835, 1881, 1882
group commands:
\group_begin: . . . . . 151, 179, 198
\group_end: . . . . . 164, 187
\group_insert_after:N 631, 650, 661,
    973, 986, 1001, 1028, 1053, 3046, 3081
H
hbox commands:
\hbox:n . . . . . 2139, 2142,
    2217, 2223, 2376, 2383, 2901, 2912
\hbox_overlap_right:n . . . . 231,
    263, 279, 320, 336, 364, 448, 1301, 1496
\hbox_set:Nn . . 1763, 1959, 2209, 2241
\hbox_set:Nw . . . . . 2192
\hbox_set_end: . . . . . 2207
\hbox_unpack:N . . . . . 2328
hook commands:
\hook_gput_code:nnn . . . . . 55
I
int commands:
\int_compare:nNnTF . . . . .
    516, 558, 656, 954, 996, 1733, 1758,
    1860, 1887, 1929, 1969, 2300, 2401,
    2697, 2725, 2838, 2845, 2861, 3087

```

`\int_const:Nn` 157, 163, 523,
 549, 584, 1767, 1882, 2037, 2575, 2763
`\int_eval:n`
 . 565, 575, 604, 615, 717, 726, 739,
 741, 745, 758, 2425, 2429, 2675,
 2700, 2707, 2720, 2930, 2938, 2943
`\int_gincr:N` 205, 371,
 522, 1563, 1608, 1881, 2036, 2105,
 2149, 2226, 2762, 2805, 2818, 2840
`\int_gset:Nn` 180, 199, 2289
`\int_gset_eq:NN` 188, 2150, 2227, 2819
`\int_if_exist:NTF` 1871
`\int_if_odd:nTF` 2212
`\int_new:N` 171, 172,
 418, 513, 519, 1589, 1835, 2032,
 2130, 2161, 2163, 2758, 2815, 2831
`\int_set:Nn` 541
`\int_set_eq:NN` ... 176, 195, 547, 2301
`\int_step_function:nnnN` 743
`\int_use:N` 373,
 404, 531, 542, 691, 827, 872, 940,
 1566, 1572, 1579, 1611, 1619, 1734,
 1759, 1774, 1861, 1874, 1886, 1888,
 1970, 2043, 2108, 2121, 2125, 2153,
 2160, 2231, 2332, 2586, 2596, 2769,
 2807, 2812, 2822, 2830, 2848, 2864
`\int_value:w`
 2454, 2489, 2646, 2657, 2675
`\int_zero:N` ... 1713, 1823, 1916, 1980

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, 677
`\g__kernel_backend_header_bool` ..
 67, 675
`__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, 687, 823,
 1000, 1006, 1008, 1027, 1052, 1119,
 1125, 1420, 1427, 1433, 1493, 1498,
 1699, 1838, 1873, 1883, 1997, 2008,
 2752, 2868, 2930, 2934, 2939, 2944
`__kernel_backend_literal_page:n`
 100, 144, 2746, 2748, 2949, 2951
`__kernel_backend_literal_pdf:n` .
 .. 89, 141, 271, 328, 1310, 3096, 3111

`__kernel_backend_literal_-
 postscript:n`
 61, 75, 76, 80, 217, 218, 220,
 221, 229, 241, 256, 1115, 2403, 2415
`__kernel_backend_literal_svg:n` .
 168, 175, 186, 194,
 204, 372, 374, 391, 1504, 1677, 1688
`__kernel_backend_matrix:n`
 128, 293, 314, 1410
`__kernel_backend_postscript:n` ..
 64,
 629, 1031, 1033, 1035, 1039, 2026,
 2082, 2097, 2139, 2145, 2185, 2217,
 2224, 2228, 2242, 2270, 2314, 2321,
 2327, 2335, 2342, 2376, 2383, 3003
`__kernel_backend_scope:n`
 173, 401, 406, 1093, 1509, 3125
`__kernel_backend_scope_begin:` ..
 83, 110, 146,
 173, 215, 239, 254, 270, 287, 313,
 327, 344, 357, 1316, 1488, 1508, 1675
`__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, 1317, 1500, 1511, 1689
`\g__kernel_backend_scope_int` ...
 171, 178, 180, 185, 189, 197, 199, 205
`\l__kernel_backend_scope_int` ...
 171, 177, 190, 196
`__kernel_color_backend_stack_-
 init:Nnn` 516, 582, 3025
`__kernel_color_backend_stack_-
 pop:n` 558, 596, 653, 3053
`__kernel_color_backend_stack_-
 push:nn`
 .. 558, 596, 649, 971, 984, 3044, 3079
`__kernel_dependency_version_-
 check:Nn` 1
`__kernel_dependency_version_-
 check:nn` 27, 29
`__kernel_kern:n`
 2144, 2146, 2375, 2379,
 2382, 2386, 2900, 2908, 2911, 2927
`\c__kernel_sys_dvipdfmx_version_-
 int` 151, 516, 558,
 656, 954, 996, 2838, 2845, 2861, 3087

M

`\MessageBreak` 40
 mode commands:
`\mode_if_horizontal:TF` ... 2291, 2298
`\mode_if_math:TF` 2189

O

\oddsidemargin	2213
opacity internal commands:	
_opacity_backend:nn	3118
_opacity_backend:nnn	2977
_opacity_backend_fill:n	2977, 3054, 3118
_opacity_backend_fill_stroke:nn	3056, 3062, 3066, 3084, 3098
\l_opacity_backend_fill_tl	3030, 3039, 3063, 3071, 3091, 3103
_opacity_backend_fillstroke:nn	3054
_opacity_backend_reset:	3032, 3081
_opacity_backend_select:n	2977, 3032, 3087, 3118
_opacity_backend_select_aux:n ..	2977, 3032, 3069, 3089, 3101
\c_opacity_backend_stack_int ...	3023, 3044, 3053, 3079
_opacity_backend_stroke:n	2977, 3054, 3118
\l_opacity_backend_stroke_tl ...	3030, 3040, 3058, 3072, 3092, 3104

P

pdf commands:	
\pdf_object_if_exist:nTF	883
\pdf_object_new:nn	885
\pdf_object_ref:n	898
\pdf_object_ref_last:	864, 873, 932, 941
\pdf_object_unnamed_write:nn ...	852, 879, 905
\pdf_object_write:nn	886
pdf internal commands:	
_pdf_backend:n	2751, 2755, 2757, 2783, 2788, 2797, 2820, 2842, 2858, 2871, 2903, 2904, 2914
_pdf_backend_annotation:nnnn ...	2131, 2439, 2816
_pdf_backend_annotation_au:nnnn	2133, 2136
\g_pdf_backend_annotation_int ...	2130, 2150, 2160, 2815, 2819, 2830
_pdf_backend_annotation_last: ..	2159, 2452, 2829
_pdf_backend_bdc:nn	2433, 2745, 2948, 2970
_pdf_backend_catalog_gput:nn ...	2028, 2545, 2754, 2954
_pdf_backend_compress_objects:n	2399, 2666, 2929, 2964

_pdf_backend_compresslevel:n ...	2399, 2666, 2929, 2964
\l_pdf_backend_content_box	2128, 2192, 2216, 2219, 2221, 2250, 2261
_pdf_backend_destination:nn ...	2340, 2508, 2869
_pdf_backend_destination:nnnn ...	2340, 2508, 2869
_pdf_backend_destination_au:nnnn	2340, 2869
_pdf_backend_emc:	2433, 2745, 2948, 2970
_pdf_backend_info_gput:nn	2028, 2545, 2754, 2954
_pdf_backend_link:nw	2170
_pdf_backend_link_aux:nw ...	2170
_pdf_backend_link_begin:n ...	2832
_pdf_backend_link_begin:nnnw ..	2463
_pdf_backend_link_begin:nw ...	2172, 2176, 2177
_pdf_backend_link_begin_aux:nw	2180, 2182
_pdf_backend_link_begin_goto:nnw	2170, 2463, 2832
_pdf_backend_link_begin_user:nnw	2170, 2463, 2832
\g_pdf_backend_link_bool	2165, 2179, 2184, 2199, 2237
\g_pdf_backend_link_dict_tl ...	2162, 2187, 2232
_pdf_backend_link_end:	2170, 2463, 2832
_pdf_backend_link_end_aux: ..	2170
\g_pdf_backend_link_int	2161, 2227, 2231, 2332, 2831, 2840, 2848, 2864
_pdf_backend_link_last:	2331, 2487, 2859
_pdf_backend_link_margin:n ...	2333, 2498, 2867
\g_pdf_backend_link_math_bool ...	2164, 2190, 2191, 2194, 2204
_pdf_backend_link_minima: ...	2170
_pdf_backend_link_outerbox:n ..	2170
\g_pdf_backend_link_sf_int	2163, 2289, 2300, 2301
_pdf_backend_link_sf_restore: ..	2170
_pdf_backend_link_sf_save: ..	2170
\l_pdf_backend_model_box ...	2129, 2209, 2241, 2249, 2260, 2275, 2277
_pdf_backend_objcompresslevel:n	2666
\g_pdf_backend_object_int	2032, 2036, 2039,

2105, 2108, 2121, 2125, 2149, 2150, 2153, 2226, 2227, 2758, 2762, 2765, 2805, 2807, 2812, 2818, 2819, 2822	\l__pdf_backend_object_last:	2124, 2644, 2811, 2956
__pdf_backend_object_new:nn	2034, 2566, 2760, 2956	
__pdf_backend_object_now:nn	2103, 2618, 2803, 2956	
\g__pdf_backend_object_prop	2032, 2040, 2051, 2061, 2565, 2583, 2599, 2758, 2766, 2773	
__pdf_backend_object_ref:n 2034, 2048, 2062, 2566, 2760, 2779, 2956		
__pdf_backend_object_write:nn	2044, 2587, 2770, 2956	
__pdf_backend_object_write:nnn 2770		
__pdf_backend_object_write_- array:nn	2044, 2770	
__pdf_backend_object_write_- dict:nn	2044, 2770	
__pdf_backend_object_write_- fstream:nn	2044, 2770	
__pdf_backend_object_write_- fstream:nnn	2078, 2080	
__pdf_backend_object_write_- stream:nn	2044, 2770	
__pdf_backend_object_write_- stream:nnn	2044	
__pdf_backend_object_write_- stream:nnnn	2770	
__pdf_backend_pageobject_ref:n	2126, 2655, 2813, 2956	
__pdf_backend_pdfmark:n	2025, 2029, 2031, 2046, 2067, 2072, 2106, 2151, 2343, 2387, 2434, 2436	
__pdf_backend_version_major:	2425, 2431, 2722, 2938, 2939, 2946, 2968	
__pdf_backend_version_major_- gset:n	2423, 2694, 2936, 2966	
__pdf_backend_version_minor:	2429, 2431, 2722, 2943, 2944, 2946, 2968	
__pdf_backend_version_minor_- gset:n	2423, 2694, 2936, 2966	
\l__pdf_breaklink_pdfmark_tl	2166, 2234, 2326	
__pdf_breaklink_postscript:n	2168, 2218, 2220, 2327	
__pdf_breaklink_usebox:N	2169, 2219, 2328	
__pdf_exp_not_i:nn	2587, 2633, 2638	
__pdf_exp_not_ii:nn	2587, 2634, 2639	
\l__pdf_internal_box	2023	
pdf.baselineskip	2170, 3456	
pdf.bordertracking	3214	
pdf.bordertracking.begin	3214	
pdf.bordertracking.continue	3214	
pdf.bordertracking.end	3214	
pdf.bordertracking.endpage	3214	
pdf.breaklink	3352	
pdf.breaklink.write	3352	
pdf.brokenlink.dict	3214	
pdf.brokenlink.rect	3214	
pdf.brokenlink.skip	3214	
pdf.count	3352	
pdf.currentrect	3352	
pdf.cvs	3136	
pdf.dest.anchor	3179	
pdf.dest.point	3179	
pdf.dest.x	3179	
pdf.dest.y	3179	
pdf.dest2device	3179	
pdf.dev.x	3179	
pdf.dev.y	3179	
pdf.dvi.pt	3136	
pdf.globaldict	3133	
pdf.leftboundary	3214	
pdf.link.dict	2170	
pdf.linkdp.pad	2170, 3140	
pdf.linkht.pad	2170, 3140	
pdf.linkmargin	3140	
pdf.llx	2170, 3143	
pdf.lly	2170, 3143	
pdf.originx	3214	
pdf.originy	3214	
pdf.outerbox	2170, 3456	
pdf.pdfmark	3456	
pdf.pdfmark.dict	3456	
pdf.pdfmark.good	3456	
pdf.pt.dvi	3136	
pdf.rect	3143	
pdf.rect.ht	3136	
pdf.rightboundary	3214	
pdf.save.linkll	3143	
pdf.save.linkur	3143	
pdf.save.ll	3143	
pdf.save.ur	3143	
pdf.tmpa	3179	
pdf.tmpb	3179	
pdf.tmpc	3179	
pdf.tmpd	3179	
pdf.urx	3143	
pdf.ury	2170, 3143	

pdfmanagement commands:		\@ifl@t@r	50, 52
\pdfmanagement_add:nnn	866,	\@makecol@hook	2304
870, 934, 938, 3023, 3027, 3041,		\current@color	14, 461, 465, 471, 485
3048, 3073, 3076, 3093, 3105, 3108		\special	2
prg commands:		tex commands:	
\prg_replicate:nn		\tex_baselineskip:D	2281
.	184, 715, 736, 746, 911	\tex_endinput:D	44
prop commands:		\tex_global:D	
\prop_gput:Nnn	2040, 2583, 2766	2668, 2685, 2699, 2706, 2713
\prop_item:Nn	2051, 2061, 2599, 2773	\tex_immediate:D	
\prop_new:N	2033, 2565, 2759	1745, 2590, 2593, 2621, 2624
\ProvidesExplFile	2	\tex_luatexversion:D	2697, 2725
		\tex_pdfannot:D	2445
		\tex_pdfcatalog:D	2551
		\tex_pdfcolorstack:D	602, 613
		\tex_pdfcolorstackinit:D	590
		\tex_pdfcompresslevel:D	2673
		\tex_pdfdest:D	2514, 2537
		\tex_pdfendlink:D	2484
		\tex_pdfextension:D	
		92, 103, 113, 122, 131,
			599, 610, 2442, 2470, 2481, 2511,
			2534, 2548, 2558, 2569, 2590, 2621
		\tex_pdffeedback:D	
		587, 2456, 2491, 2578, 2648, 2659
		\tex_pdfinfo:D	2561
		\tex_pdflastannot:D	2459
		\tex_pdflastlink:D	2494
		\tex_pdflastobj:D	2581, 2651
		\tex_pdflastximage:D	1764, 1768
		\tex_pdflinkmargin:D	2504
		\tex_pdfliteral:D	95, 106
		\tex_pdfmajorversion:D	
		2704, 2706, 2730, 2731
		\tex_pdfminorversion:D	2718, 2742
		\tex_pdfobj:D	2572, 2593, 2624
		\tex_pdfobjcompresslevel:D	2690
		\tex_pdfpageref:D	2662
		\tex_pdfrefximage:D	1764, 1773
		\tex_pdfrestore:D	125
		\tex_pdfsave:D	116
		\tex_pdfsetmatrix:D	134
		\tex_pdfstartlink:D	2473
		\tex_pdfvariable:D	2501,
			2670, 2687, 2699, 2715, 2726, 2739
		\tex_pdfximage:D	1745
		\tex_spacefactor:D	2292, 2301
		\tex_special:D	46
		\tex_the:D	1768, 2726, 2731, 2737
		\tex_vss:D	2377, 2384, 2906, 2925
		\tex_XeTeXpdffile:D	1925, 1967
		\tex_XeTeXpicfile:D	1918
		TeXcolorseparation	3130
		\textwidth	2270
TeX and L ^A T _E X 2 _ε commands:			
\@ccclv	2310, 2312, 2320		

tl commands:

`\c_space_tl` 295, 300, 303, 532,
 782, 985, 1548, 1701, 1702, 1703,
 1704, 1840, 1841, 1842, 1843, 1888,
 1891, 1893, 1894, 1895, 1896, 1968,
 1970, 1999, 2000, 2001, 2002, 2232,
 2461, 2496, 2653, 2664, 2822, 2849
`\tl_clear:N` 1714, 1722, 1728,
 1824, 1830, 1917, 1923, 1981, 1987
`\tl_gclear:N` 1586, 1622
`\tl_gset:Nn` 1545, 2187
`\tl_if_blank:nTF`
 533, 592, 730, 747, 754, 772, 856, 948
`\tl_if_empty:NTF` . 1548, 1717, 1752,
 1760, 1858, 1862, 1889, 1904, 1938
`\tl_if_empty:nTF` 1642
`\tl_if_empty_p:N` 1748, 1901
`\tl_if_head_is_space:nTF` 461
`\tl_new:N` 637,
 638, 1552, 1710, 2162, 2166, 3030, 3031
`\tl_put_right:Nn` 2308
`\tl_set:Nn` . 463, 475, 491, 494, 497,

501, 504, 647, 648, 970, 983, 1715,
 1730, 1803, 2167, 2326, 3039, 3040,
 3071, 3072, 3091, 3092, 3103, 3104
`\tl_to_str:n` 2038,
 2043, 2576, 2586, 2597, 2764, 2769
`\tl_use:N` 814, 891

token commands:

`\c_math_toggle_token` 2195, 2205

U

use commands:

`\use:N` 43, 2060, 2120, 2778, 2806
`\use:n` 59, 465, 501, 524,
 868, 936, 1060, 1070, 1083, 1256,
 1380, 1445, 1457, 1469, 1627, 1945
`\use_none:n` 1642, 1644, 2304

V

`\value` 2212

vbox commands:

`\vbox_set:Nn` 2312
`\vbox_to_zero:n` 2373, 2380, 2898, 2909
`\vbox_unpack_drop:N` 2320