

File I

Implementation

1 l3backend-basics Implementation

```
1 <*initex | 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 <*package>
3 \ProvidesExplFile
4 <*dvipdfmx>
5 {l3backend-dvipdfmx.def}{2019-04-06}{ }
6 {L3 backend support: dvipdfmx}
7 </dvipdfmx>
8 <*dvips>
9 {l3backend-dvips.def}{2019-04-06}{ }
10 {L3 backend support: dvips}
11 </dvips>
12 <*dvisvgm>
13 {l3backend-dvisvgm.def}{2019-04-06}{ }
14 {L3 backend support: dvisvgm}
15 </dvisvgm>
16 <*pdfmode>
17 {l3backend-pdfmode.def}{2019-04-06}{ }
18 {L3 backend support: PDF mode}
19 </pdfmode>
20 <*xdvipdfmx>
21 {l3backend-xdvipdfmx.def}{2019-04-06}{ }
22 {L3 backend support: xdvipdfmx}
23 </xdvipdfmx>
24 </package>
```

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 `pdfmode`-like.
- `pdfmode` and `(x)dvipdfmx` share drawing routines.
- `xdvipdfmx` is largely the same as `dvipdfmx` so takes most of the same code.

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.

```
25 \cs_new_eq:NN \__kernel_backend_literal:e \tex_special:D
26 \cs_new_protected:Npn \__kernel_backend_literal:n #1
27 { \__kernel_backend_literal:e { \exp_not:n {#1} } }
28 \cs_generate_variant:Nn \__kernel_backend_literal:n { x }
```

(End definition for `__kernel_backend_literal:e`.)

1.1 dvips backend

29 `<*dvips>`

`_kernel_backend_literal_postscript:n` Literal PostScript can be included using a few low-level formats. Here, we use the form
`_kernel_backend_literal_postscript:x` 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.

```
30 \cs_new_protected:Npn \__kernel_backend_literal_postscript:n #1
31   { \__kernel_backend_literal:n { ps:: #1 } }
32 \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
`_kernel_backend_postscript:x` not done automatically by `ps:` or `ps::`, in contrast to `!` or `"`).

```
33 \cs_new_protected:Npn \__kernel_backend_postscript:n #1
34   { \__kernel_backend_literal:n { ps: SDict ~ begin ~ #1 ~ end } }
35 \cs_generate_variant:Nn \__kernel_backend_postscript:n { x }
```

(End definition for `_kernel_backend_postscript:n`.)

`_kernel_backend_postscript_header: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.

```
36 \cs_new_protected:Npx \__kernel_backend_postscript_header:n #1
37 <*initex>
38   { \__kernel_backend_literal:n { ! #1 } }
39 </initex>
40 <*package>
41   {
42     \cs_if_exist:NTF \AtBeginDvi
43       { \exp_not:N \AtBeginDvi }
44       { \use:n }
45       { \__kernel_backend_literal:n { ! #1 } }
46   }
47 </package>
```

(End definition for `_kernel_backend_postscript_header:n`.)

`_kernel_backend_align_begin:` In `dvips` there is no built-in saving of the current position, and so some additional
`_kernel_backend_align_end:` 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.

```
48 \cs_new_protected:Npn \__kernel_backend_align_begin:
49   {
50     \__kernel_backend_literal:n { ps::[begin] }
51     \__kernel_backend_literal_postscript:n { currentpoint }
52     \__kernel_backend_literal_postscript:n { currentpoint-translate }
53   }
54 \cs_new_protected:Npn \__kernel_backend_align_end:
55   {
56     \__kernel_backend_literal_postscript:n { neg-exch-neg-exch-translate }
57     \__kernel_backend_literal:n { ps::[end] }
58   }
```

(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
`_kernel_backend_scope_end:` (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.

```
59 \cs_new_protected:Npn \_kernel_backend_scope_begin:
60   { \_kernel_backend_literal:n { ps:gsave } }
61 \cs_new_protected:Npn \_kernel_backend_scope_end:
62   { \_kernel_backend_literal:n { ps:grestore } }
```

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

```
63 \</dvips>
```

1.2 pdfmode backend

```
64 <*pdfmode>
```

The direct PDF backend covers both pdfTeX and LuaTeX. The latter renames and restructures the backend primitives but this can be handled at one level of abstraction. As such, we avoid using two separate backends for this material at the cost of some `x-`type definitions to get everything expanded up-front.

`_kernel_backend_literal_pdf:n` This is equivalent to `\special{pdf:}` but the engine can track it. Without the `direct`
`_kernel_backend_literal_pdf:x` 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).

```
65 \cs_new_protected:Npx \_kernel_backend_literal_pdf:n #1
66   {
67     \cs_if_exist:NTF \tex_pdfextension:D
68     { \tex_pdfextension:D literal }
69     { \tex_pdfliteral:D }
70     { \exp_not:N \exp_not:n {#1} }
71   }
72 \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.

```
73 \cs_new_protected:Npx \_kernel_backend_literal_page:n #1
74   {
75     \cs_if_exist:NTF \tex_pdfextension:D
76     { \tex_pdfextension:D literal ~ }
77     { \tex_pdfliteral:D }
78     page
79     { \exp_not:N \exp_not:n {#1} }
80   }
```

(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:
81 \cs_new_protected:Npx \_kernel_backend_scope_begin:
82   {
83     \cs_if_exist:NTF \tex_pdfextension:D
84     { \tex_pdfextension:D save \scan_stop: }
85     { \tex_pdfsave:D }
86   }
```

```

87 \cs_new_protected:Npx \__kernel_backend_scope_end:
88 {
89   \cs_if_exist:NTF \tex_pdfextension:D
90     { \tex_pdfextension:D restore \scan_stop: }
91     { \tex_pdfrestore:D }
92 }

```

(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.

```

93 \cs_new_protected:Npx \__kernel_backend_matrix:n #1
94 {
95   \cs_if_exist:NTF \tex_pdfextension:D
96     { \tex_pdfextension:D setmatrix }
97     { \tex_pdfsetmatrix:D }
98     { \exp_not:N \exp_not:n {#1} }
99 }
100 \cs_generate_variant:Nn \__kernel_backend_matrix:n { x }

```

(End definition for __kernel_backend_matrix:n.)

101 </pdfmode>

1.3 dvipdfmx backend

102 <*dvipdfmx | xdvipdfmx>

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

__kernel_backend_literal_pdf:n Equivalent to pdf:content but favored as the link to the pdfTeX primitive approach is clearer.

__kernel_backend_literal_pdf:x

```

103 \cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1
104 { \__kernel_backend_literal:n { pdf:literal~ #1 } }
105 \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!

```

106 \cs_new_protected:Npn \__kernel_backend_literal_page:n #1
107 { \__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.

__kernel_backend_scope_end:

```

108 \cs_new_protected:Npn \__kernel_backend_scope_begin:
109 { \__kernel_backend_literal:n { x:gsave } }
110 \cs_new_protected:Npn \__kernel_backend_scope_end:
111 { \__kernel_backend_literal:n { x:grestore } }

```

(End definition for __kernel_backend_scope_begin: and __kernel_backend_scope_end:.)

112 </dvipdfmx | xdvipdfmx>

1.4 dvisvgm backend

```
113 <*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.

```
114 \cs_new_protected:Npn \_kernel_backend_literal_svg:n #1
115   { \_kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }
116 \cs_generate_variant:Nn \_kernel_backend_literal_svg:n { x }
```

(End definition for `_kernel_backend_literal_svg:n`.)

`_kernel_backend_scope_begin:` A scope in SVG terms is slightly different to the other backends as operations have to be “tied” to these not simply inside them.

```
117 \cs_new_protected:Npn \_kernel_backend_scope_begin:
118   { \_kernel_backend_literal_svg:n { <g> } }
119 \cs_new_protected:Npn \_kernel_backend_scope_end:
120   { \_kernel_backend_literal_svg:n { </g> } }
```

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

`_kernel_backend_scope_begin:n` In SVG transformations, clips and so on are attached directly to scopes so we need a way or allowing for that. This is rather more useful than `_kernel_backend_scope_begin:` as a result. No assumptions are made about the nature of the scoped operation(s).

```
121 \cs_new_protected:Npn \_kernel_backend_scope_begin:n #1
122   { \_kernel_backend_literal_svg:n { <g~ #1 > } }
123 \cs_generate_variant:Nn \_kernel_backend_scope_begin:n { x }
```

(End definition for `_kernel_backend_scope_begin:n`.)

```
124 </dvisvgm>
```

```
125 </initex | package>
```

2 l3backend-box Implementation

```
126 <*initex | package>
```

```
127 <@@=box>
```

2.1 dvips backend

```
128 <*dvips>
```

`_box_backend_clip:N` The `dvips` backend scales all absolute dimensions based on the output resolution selected and any \TeX 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.

```
129 \cs_new_protected:Npn \_box_backend_clip:N #1
130   {
131     \_kernel_backend_scope_begin:
132     \_kernel_backend_align_begin:
133     \_kernel_backend_literal_postscript:n { matrix~currentmatrix }
134     \_kernel_backend_literal_postscript:n
```

```

135     { Resolution~72~div~VResolution~72~div~scale }
136   \__kernel_backend_literal_postscript:n { DVImag-dup-scale }
137   \__kernel_backend_literal_postscript:x
138     {
139     0 ~
140     \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
141     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
142     \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
143     rectclip
144     }
145   \__kernel_backend_literal_postscript:n { setmatrix }
146   \__kernel_backend_align_end:
147   \hbox_overlap_right:n { \box_use:N #1 }
148   \__kernel_backend_scope_end:
149   \skip_horizontal:n { \box_wd:N #1 }
150 }

```

(End definition for __box_backend_clip:N.)

__box_backend_rotate:Nn __box_backend_rotate_aux: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.

```

151 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
152 { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
153 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
154 {
155   \__kernel_backend_scope_begin:
156   \__kernel_backend_align_begin:
157   \__kernel_backend_literal_postscript:x
158     {
159       \fp_compare:nNnTF {#2} = \c_zero_fp
160       { 0 }
161       { \fp_eval:n { round ( -(#2) , 5 ) } } ~
162       rotate
163     }
164   \__kernel_backend_align_end:
165   \box_use:N #1
166   \__kernel_backend_scope_end:
167 }

```

(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.

```

168 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
169 {
170   \__kernel_backend_scope_begin:
171   \__kernel_backend_align_begin:
172   \__kernel_backend_literal_postscript:x
173     {
174       \fp_eval:n { round ( #2 , 5 ) } ~
175       \fp_eval:n { round ( #3 , 5 ) } ~
176       scale
177     }
178   \__kernel_backend_align_end:

```

```

179     \hbox_overlap_right:n { \box_use:N #1 }
180     \__kernel_backend_scope_end:
181 }

```

(End definition for __box_backend_scale:Nnn.)

```

182 </dvips>

```

2.2 pdfmode backend

```

183 <*pdfmode>

```

__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.

```

184 \cs_new_protected:Npn \__box_backend_clip:N #1
185 {
186     \__kernel_backend_scope_begin:
187     \__kernel_backend_literal_pdf:x
188     {
189         0~
190         \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
191         \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
192         \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
193         re~W~n
194     }
195     \hbox_overlap_right:n { \box_use:N #1 }
196     \__kernel_backend_scope_end:
197     \skip_horizontal:n { \box_wd:N #1 }
198 }

```

(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.

```

199 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
200 { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
201 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
202 {
203     \__kernel_backend_scope_begin:
204     \box_set_wd:Nn #1 { Opt }
205     \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
206     \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
207     { \fp_zero:N \l__box_backend_cos_fp }
208     \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
209     \__kernel_backend_matrix:x
210     {
211         \fp_use:N \l__box_backend_cos_fp \c_space_tl

```

```

212     \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp
213     { 0~0 }
214     {
215         \fp_use:N \l__box_backend_sin_fp
216         \c_space_tl
217         \fp_eval:n { -\l__box_backend_sin_fp }
218     }
219     \c_space_tl
220     \fp_use:N \l__box_backend_cos_fp
221 }
222 \box_use:N #1
223 \__kernel_backend_scope_end:
224 }
225 \fp_new:N \l__box_backend_cos_fp
226 \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.

```

227 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
228 {
229     \__kernel_backend_scope_begin:
230     \__kernel_backend_matrix:x
231     {
232         \fp_eval:n { round ( #2 , 5 ) } ~
233         0~0~
234         \fp_eval:n { round ( #3 , 5 ) }
235     }
236     \hbox_overlap_right:n { \box_use:N #1 }
237     \__kernel_backend_scope_end:
238 }

```

(End definition for `__box_backend_scale:Nnn`.)

239 `</pdfmode>`

2.3 dvipdfmx backend

240 `<*dvipdfmx | xdvipdfmx>`

`__box_backend_clip:N` The code here is identical to that for `pdfmode`: unlike rotation and scaling, there is no higher-level support in the backend for clipping.

```

241 \cs_new_protected:Npn \__box_backend_clip:N #1
242 {
243     \__kernel_backend_scope_begin:
244     \__kernel_backend_literal_pdf:x
245     {
246         0~
247         \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
248         \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
249         \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
250         re~W~n
251     }
252     \hbox_overlap_right:n { \box_use:N #1 }
253     \__kernel_backend_scope_end:

```



```

254     \skip_horizontal:n { \box_wd:N #1 }
255   }

```

(End definition for `_box_backend_clip:N`.)

`_box_backend_rotate:Nn` `_box_backend_rotate_aux:Nn` Rotating in (x)dvipdfmx 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.

```

256 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
257   { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
258 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
259   {
260     \_kernel_backend_scope_begin:
261     \_kernel_backend_literal:x
262     {
263       x:rotate~
264       \fp_compare:nNnTF {#2} = \c_zero_fp
265       { 0 }
266       { \fp_eval:n { round ( #2 , 5 ) } } }
267     }
268     \box_use:N #1
269     \_kernel_backend_scope_end:
270   }

```

(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.

```

271 \cs_new_protected:Npn \_box_backend_scale:Nnn #1#2#3
272   {
273     \_kernel_backend_scope_begin:
274     \_kernel_backend_literal:x
275     {
276       x:scale~
277       \fp_eval:n { round ( #2 , 5 ) } ~
278       \fp_eval:n { round ( #3 , 5 ) }
279     }
280     \hbox_overlap_right:n { \box_use:N #1 }
281     \_kernel_backend_scope_end:
282   }

```

(End definition for `_box_backend_scale:Nnn`.)

```

283 </dvipdfmx | xdvipdfmx>

```

2.4 dvisvgm backend

```

284 <*dvisvgm>

```

`_box_backend_clip:N` `\g__box_clip_path_int` Clipping in SVG is more involved than with other backends. The first issue is that the clipping path must be defined separately from where it is used, so we need to track how many paths have applied. The naming here uses l3cp as the namespace with a number

following. Rather than use a rectangular operation, we define the path manually as this allows it to have a depth: easier than the alternative approach of shifting content up and down using scopes to allow for the depth of the $\text{T}_{\text{E}}\text{X}$ box and keep the reference point the same!

```

285 \cs_new_protected:Npn \__box_backend_clip:N #1
286 {
287   \int_gincr:N \g__box_clip_path_int
288   \__kernel_backend_literal_svg:x
289   { < clipPath~id = " l3cp \int_use:N \g__box_clip_path_int " > }
290   \__kernel_backend_literal_svg:x
291   {
292     <
293     path ~ d =
294     "
295         M ~ 0 ~
296         \dim_to_decimal:n { -\box_dp:N #1 } ~
297         L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
298         \dim_to_decimal:n { -\box_dp:N #1 } ~
299         L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
300         \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
301         L ~ 0 ~
302         \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
303         Z
304     "
305     />
306   }
307   \__kernel_backend_literal_svg:n
308   { < /clipPath > }

```

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

```

309   \__kernel_backend_scope_begin:n
310   {
311     transform =
312     "
313         translate ( { ?x } , { ?y } ) ~
314         scale ( 1 , -1 )
315     "
316   }
317   \__kernel_backend_scope_begin:x
318   {
319     clip-path =
320     "url ( \c_hash_str l3cp \int_use:N \g__box_clip_path_int ) "
321   }
322   \__kernel_backend_scope_begin:n
323   {
324     transform =
325     "
326         scale ( -1 , 1 ) ~
327         translate ( { ?x } , { ?y } ) ~
328         scale ( -1 , -1 )

```

```

329         "
330     }
331     \box_use:N #1
332     \__kernel_backend_scope_end:
333     \__kernel_backend_scope_end:
334     \__kernel_backend_scope_end:
335 %     \skip_horizontal:n { \box_wd:N #1 }
336 }
337 \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.

```

338 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
339 {
340     \__kernel_backend_scope_begin:x
341     {
342         transform =
343         "
344             rotate
345             ( \fp_eval:n { round ( -#2 ) , 5 ) } , ~ { ?x } , ~ { ?y } )
346         "
347     }
348     \box_use:N #1
349     \__kernel_backend_scope_end:
350 }

```

(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).

```

351 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
352 {
353     \__kernel_backend_scope_begin:x
354     {
355         transform =
356         "
357             translate ( { ?x } , { ?y } ) ~
358             scale
359             (
360                 \fp_eval:n { round ( -#2 , 5 ) } ,
361                 \fp_eval:n { round ( -#3 , 5 ) }
362             ) ~
363             translate ( { ?x } , { ?y } ) ~
364             scale ( -1 )
365         "
366     }
367     \hbox_overlap_right:n { \box_use:N #1 }
368     \__kernel_backend_scope_end:
369 }

```

(End definition for `_box_backend_scale:Nnn`.)

```
370 </dvisvgm>
371 </initex | package>
```

3 I3backend-color Implementation

```
372 <*initex | package>
373 <@@=color>
```

Color support is split into two parts: a “general” concept and one directly linked to drawings (or rather the split between filling and stroking). General color is relatively easy to handle: we have a color stack available with all modern drivers, and can use that. Whilst (x)dvipdfmx does have its own approach to color specials, it is easier to use dvips-like ones for all cases except direct PDF output.

3.1 dvips-style

```
374 <*dvisvgm | dvipdfmx | dvips | xdvipdfmx>
```

`_color_backend_pickup:N` 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).

```
375 <*package>
376 \cs_new_protected:Npn \_color_backend_pickup:N #1 { }
377 \AtBeginDocument
378 {
379   \cs_if_exist:cT { ver@color.sty }
380   {
381     \cs_set_protected:Npn \_color_backend_pickup:N #1
382     {
383       \exp_args:NV \tl_if_head_is_space:nTF \current@color
384       {
385         \tl_set:Nx #1
386         {
387           spot ~
388           \exp_after:wN \use:n \current@color \c_space_tl 1
389         }
390       }
391       {
392         \exp_last_unbraced:Nx \_color_backend_pickup:w
393         { \current@color } \q_stop #1
394       }
395     }
396     \cs_new_protected:Npn \_color_backend_pickup:w #1 ~ #2 \q_stop #3
397     { \tl_set:Nn #3 { #1 ~ #2 } }
398   }
399 }
400 </package>
```

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

`_color_backend_cmyk:n` Push the data to the stack. In the case of dvips also reset the drawing fill color in raw PostScript.

```
401 \cs_new_protected:Npn \_color_backend_cmyk:n #1#2#3#4
402 {
```

```
\_color_backend_gray:n
\_color_backend_rgb:n
\_color_backend_spot:n
\_color_backend_select:n
\_color_backend_select:x
\_color_backend_reset:
color.fc
```

```

403   \_color_backend_select:x
404   {
405     cmyk~
406     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
407     \fp_eval:n {#3} ~ \fp_eval:n {#4}
408   }
409 }
410 \cs_new_protected:Npn \_color_backend_gray:n #1
411 { \_color_backend_select:x { gray~ \fp_eval:n {#1} } }
412 \cs_new_protected:Npn \_color_backend_rgb:nnn #1#2#3
413 {
414   \_color_backend_select:x
415   { rgb~ \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} }
416 }
417 \cs_new_protected:Npn \_color_backend_spot:nn #1#2
418 { \_color_backend_select:n { #1 } }
419 \cs_new_protected:Npn \_color_backend_select:n #1
420 {
421   \_kernel_backend_literal:n { color-push~ #1 }
422 <*dvips>
423   \_kernel_backend_postscript:n { /color.fc~{ }~def }
424 </dvips>
425   \group_insert_after:N \_color_backend_reset:
426 }
427 \cs_generate_variant:Nn \_color_backend_select:n { x }
428 \cs_new_protected:Npn \_color_backend_reset:
429 { \_kernel_backend_literal:n { color~pop } }

```

(End definition for `_color_backend_cmyk:nnnn` and others. This function is documented on page ??.)

```

430 </dvisvgm | dvipdfmx | dvips | xdvipdfmx>

```

3.2 pdfmode

```

431 <*pdfmode>

```

`_color_backend_pickup:N`
`_color_backend_pickup:w`

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

```

432 <*package>
433 \cs_new_protected:Npn \_color_backend_pickup:N #1 { }
434 \AtBeginDocument
435 {
436   \cs_if_exist:cT { ver@color.sty }
437   {
438     \cs_set_protected:Npn \_color_backend_pickup:N #1
439     {
440       \exp_last_unbraced:Nx \_color_backend_pickup:w
441       { \current@color } ~ 0 ~ 0 ~ 0 \q_stop #1
442     }
443     \cs_new_protected:Npn \_color_backend_pickup:w
444     #1 ~ #2 ~ #3 ~ #4 ~ #5 ~ #6 \q_stop #7
445     {

```

```

446     \str_if_eq:nnTF {#2} { g }
447     { \tl_set:Nn #7 { gray ~ #1 } }
448     {
449     \str_if_eq:nnTF {#4} { rg }
450     { \tl_set:Nn #7 { rgb ~ #1 ~ #2 ~ #3 } }
451     {
452     \str_if_eq:nnTF {#5} { k }
453     { \tl_set:Nn #7 { cmyk ~ #1 ~ #2 ~ #3 ~ #4 } }
454     {
455     \str_if_eq:nnTF {#2} { cs }
456     {
457     \tl_set:Nx #7 { spot ~ \use_none:n #1 ~ #5 }
458     }
459     {
460     \tl_set:Nn #7 { gray ~ 0 }
461     }
462     }
463     }
464     }
465     }
466     }
467     }
468 </package>

```

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

`\l__kernel_color_stack_int` pdfTeX and LuaTeX have multiple stacks available, and to track which one is in use a variable is required.

```

469 \int_new:N \l__kernel_color_stack_int

```

(End definition for `\l__kernel_color_stack_int`.)

`_color_backend_cmyk:n` Simply dump the data, but allowing for LuaTeX.

```

\__color_backend_cmyk_aux:nnnn
\__color_backend_gray:n
\__color_backend_gray_aux:n
\__color_backend_rgb:nnn
\__color_backend_rgb_aux:nnn
\__color_backend_spot:nn
\__color_backend_select:n
\__color_backend_select:x
\__color_backend_reset:
470 \cs_new_protected:Npn \__color_backend_cmyk:n #1#2#3#4
471 {
472     \use:x
473     {
474     \__color_backend_cmyk_aux:nnnn
475     { \fp_eval:n {#1} }
476     { \fp_eval:n {#2} }
477     { \fp_eval:n {#3} }
478     { \fp_eval:n {#4} }
479     }
480     }
481 \cs_new_protected:Npn \__color_backend_cmyk_aux:nnnn #1#2#3#4
482 {
483     \__color_backend_select:n
484     { #1 ~ #2 ~ #3 ~ #4 ~ k ~ #1 ~ #2 ~ #3 ~ #4 ~ K }
485     }
486 \cs_new_protected:Npn \__color_backend_gray:n #1
487 { \exp_args:Nx \__color_backend_gray_aux:n { \fp_eval:n {#1} } }
488 \cs_new_protected:Npn \__color_backend_gray_aux:n #1
489 { \__color_backend_select:n { #1 ~ g ~ #1 ~ G } }
490 \cs_new_protected:Npn \__color_backend_rgb:nnn #1#2#3

```

```

491 {
492   \use:x
493   {
494     \_color_backend_rgb_aux:nnn
495     { \fp_eval:n {#1} }
496     { \fp_eval:n {#2} }
497     { \fp_eval:n {#3} }
498   }
499 }
500 \cs_new_protected:Npn \_color_backend_rgb_aux:nnn #1#2#3
501 { \_color_backend_select:n { #1 ~ #2 ~ #3 ~ rg ~ #1 ~ #2 ~ #3 ~ RG } }
502 \cs_new_protected:Npn \_color_backend_spot:nn #1#2
503 { \_color_backend_select:n { /#1 ~ cs ~ /#1 ~ CS ~ #2 ~ sc ~ #2 ~ SC } }
504 \cs_new_protected:Npx \_color_backend_select:n #1
505 {
506   \cs_if_exist:NTF \tex_pdfextension:D
507   { \tex_pdfextension:D colorstack }
508   { \tex_pdfcolorstack:D }
509   \exp_not:N \l__kernel_color_stack_int push {#1}
510   \group_insert_after:N \exp_not:N \_color_backend_reset:
511 }
512 \cs_generate_variant:Nn \_color_backend_select:n { x }
513 \cs_new_protected:Npx \_color_backend_reset:
514 {
515   \cs_if_exist:NTF \tex_pdfextension:D
516   { \tex_pdfextension:D colorstack }
517   { \tex_pdfcolorstack:D }
518   \exp_not:N \l__kernel_color_stack_int pop \scan_stop:
519 }

```

(End definition for `_color_backend_cmyk:nnnn` and others.)

```

520 </pdfmode>
521 </initex | package>

```

4 I3backend-draw Implementation

```

522 <*initex | package>
523 <@@=draw>

```

4.1 dvips backend

```

524 <*dvips>

```

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

```

525 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
526 \cs_generate_variant:Nn \_draw_backend_literal:n { x }

```

(End definition for `_draw_backend_literal:n`.)

`_draw_backend_begin:` 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. The definition of `/color.fc` deals with fill color in paths. 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.

```

527 \cs_new_protected:Npn \_draw_backend_begin:
528 {
529   \_kernel_backend_literal:n { ps::[begin] }
530   \_draw_backend_literal:n { @beginspecial }
531   \_draw_backend_literal:n { SDict ~ begin ~ /color.fc ~ { } ~ def ~ end }
532 }
533 \cs_new_protected:Npn \_draw_backend_end:
534 {
535   \_draw_backend_literal:n { @endspecial }
536   \_kernel_backend_literal:n { ps::[end] }
537 }

```

(End definition for `_draw_backend_begin:`, `_draw_backend_end:`, and `color.fc`. This function is documented on page ??.)

`_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.

```

538 \cs_new_protected:Npn \_draw_backend_scope_begin:
539 { \_draw_backend_literal:n { save } }
540 \cs_new_protected:Npn \_draw_backend_scope_end:
541 { \_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.

```

542 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
543 {
544   \_draw_backend_literal:x
545   {
546     \dim_to_decimal_in_bp:n {#1} ~
547     \dim_to_decimal_in_bp:n {#2} ~ moveto
548   }
549 }
550 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
551 {
552   \_draw_backend_literal:x
553   {
554     \dim_to_decimal_in_bp:n {#1} ~
555     \dim_to_decimal_in_bp:n {#2} ~ lineto
556   }
557 }
558 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
559 {
560   \_draw_backend_literal:x
561   {
562     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~

```



```

563         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
564         moveto-dup-0~rlineto-exch-0~exch~rlineto-neg-0~rlineto-closepath
565     }
566 }
567 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
568 {
569     \__draw_backend_literal:x
570     {
571         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
572         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
573         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
574         curveto
575     }
576 }

```

(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
577 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
578 { \bool_gset_true:N \g__draw_draw_eor_bool }
579 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
580 { \bool_gset_false:N \g__draw_draw_eor_bool }
581 \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

```

```

582 \cs_new_protected:Npn \__draw_backend_closepath:
583 { \__draw_backend_literal:n { closepath } }
584 \cs_new_protected:Npn \__draw_backend_stroke:
585 {
586     \__draw_backend_literal:n { stroke }
587     \bool_if:NT \g__draw_draw_clip_bool
588     {
589         \__draw_backend_literal:x
590         {
591             \bool_if:NT \g__draw_draw_eor_bool { eo }
592             clip
593         }
594     }
595     \__draw_backend_literal:n { newpath }
596     \bool_gset_false:N \g__draw_draw_clip_bool
597 }
598 \cs_new_protected:Npn \__draw_backend_closestroke:
599 {
600     \__draw_backend_closepath:
601     \__draw_backend_stroke:
602 }

```

```

603 \cs_new_protected:Npn \__draw_backend_fill:
604 {
605   \__draw_backend_literal:n { gsave }
606   \__draw_backend_literal:n { color.fc }
607   \__draw_backend_literal:x
608   {
609     \bool_if:NT \g__draw_draw_eor_bool { eo }
610     fill
611   }
612   \__draw_backend_literal:n { grestore }
613   \bool_if:NT \g__draw_draw_clip_bool
614   {
615     \__draw_backend_literal:x
616     {
617       \bool_if:NT \g__draw_draw_eor_bool { eo }
618       clip
619     }
620   }
621   \__draw_backend_literal:n { newpath }
622   \bool_gset_false:N \g__draw_draw_clip_bool
623 }
624 \cs_new_protected:Npn \__draw_backend_fillstroke:
625 {
626   \__draw_backend_literal:n { gsave }
627   \__draw_backend_literal:n { color.fc }
628   \__draw_backend_literal:x
629   {
630     \bool_if:NT \g__draw_draw_eor_bool { eo }
631     fill
632   }
633   \__draw_backend_literal:n { grestore }
634   \__draw_backend_literal:n { stroke }
635   \bool_if:NT \g__draw_draw_clip_bool
636   {
637     \__draw_backend_literal:x
638     {
639       \bool_if:NT \g__draw_draw_eor_bool { eo }
640       clip
641     }
642   }
643   \__draw_backend_literal:n { newpath }
644   \bool_gset_false:N \g__draw_draw_clip_bool
645 }
646 \cs_new_protected:Npn \__draw_backend_clip:
647 { \bool_gset_true:N \g__draw_draw_clip_bool }
648 \bool_new:N \g__draw_draw_clip_bool
649 \cs_new_protected:Npn \__draw_backend_discardpath:
650 {
651   \bool_if:NT \g__draw_draw_clip_bool
652   {
653     \__draw_backend_literal:x
654     {
655       \bool_if:NT \g__draw_draw_eor_bool { eo }
656       clip

```

```

657     }
658   }
659   \__draw_backend_literal:n { newpath }
660   \bool_gset_false:N \g__draw_draw_clip_bool
661 }

```

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

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
662 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
663 {
664   \__draw_backend_literal:x
665   {
666     [
667       \exp_args:Nf \use:n
668       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
669     ] ~
670     \dim_to_decimal_in_bp:n {#2} ~ setdash
671   }
672 }
673 \cs_new:Npn \__draw_backend_dash:n #1
674 { ~ \dim_to_decimal_in_bp:n {#1} }
675 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
676 {
677   \__draw_backend_literal:x
678   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
679 }
680 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
681 { \__draw_backend_literal:x { \fp_eval:n {#1} ~ setmiterlimit } }
682 \cs_new_protected:Npn \__draw_backend_cap_butt:
683 { \__draw_backend_literal:n { 0 ~ setlinecap } }
684 \cs_new_protected:Npn \__draw_backend_cap_round:
685 { \__draw_backend_literal:n { 1 ~ setlinecap } }
686 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
687 { \__draw_backend_literal:n { 2 ~ setlinecap } }
688 \cs_new_protected:Npn \__draw_backend_join_miter:
689 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
690 \cs_new_protected:Npn \__draw_backend_join_round:
691 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
692 \cs_new_protected:Npn \__draw_backend_join_bevel:
693 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

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

For `dvips`, we can use the standard color stack to deal with stroke color, but for fills have to switch to raw PostScript. This is thus not handled by the stack, but the context is very restricted. See also how fills are implemented.

```

\__draw_backend_color_fill_cmyk:nnnn
\__draw_backend_color_stroke_cmyk:nnnn
\__draw_backend_color_fill_gray:n
\__draw_backend_color_stroke_gray:n
\__draw_backend_color_fill_rgb:nnm
\__draw_backend_color_stroke_rgb:nnm
\__draw_backend_color_fill:n
\__draw_backend_color_fill:x
\__draw_backend_color_stroke:n
\__draw_backend_color_stroke:x
694 \cs_new_protected:Npn \__draw_backend_color_fill_cmyk:nnnn #1#2#3#4
695 {
696   \__draw_backend_color_fill:x
697   {
698     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
699     \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
700     setcmykcolor

```

```

701     }
702   }
703   \cs_new_protected:Npn \__draw_backend_color_stroke_cmyk:nnnn #1#2#3#4
704   {
705     \__draw_backend_color_stroke:x
706     {
707       cmyk ~
708       \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
709       \fp_eval:n {#3} ~ \fp_eval:n {#4}
710     }
711   }
712   \cs_new_protected:Npn \__draw_backend_color_fill_gray:n #1
713   { \__draw_backend_color_fill:x { \fp_eval:n {#1} ~ setgray } }
714   \cs_new_protected:Npn \__draw_backend_color_stroke_gray:n #1
715   { \__draw_backend_color_stroke:x { gray ~ \fp_eval:n {#1} } }
716   \cs_new_protected:Npn \__draw_backend_color_fill_rgb:nnn #1#2#3
717   {
718     \__draw_backend_color_fill:x
719     { \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} ~ setrgbcolor }
720   }
721   \cs_new_protected:Npn \__draw_backend_color_stroke_rgb:nnn #1#2#3
722   {
723     \__draw_backend_color_stroke:x
724     { rgb ~ \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} }
725   }
726   \cs_new_protected:Npn \__draw_backend_color_fill:n #1
727   {
728     \__kernel_backend_postscript:n
729     { /color.fc ~ { #1 } ~ def }
730   }
731   \cs_generate_variant:Nn \__draw_backend_color_fill:n { x }
732   \cs_new_protected:Npn \__draw_backend_color_stroke:n #1
733   {
734     \__kernel_backend_literal:n { color~push~#1 }
735     \group_insert_after:N \__draw_color_reset:
736   }
737   \cs_generate_variant:Nn \__draw_backend_color_stroke:n { x }

```

(End definition for __draw_backend_color_fill_cmyk:nnnn 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.* (x)dvipdfmx). Thus we take the shortest path available and simply dump the matrix as given.

```

738   \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
739   {
740     \__draw_backend_literal:n
741     {
742       [
743         \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
744         \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
745         0 ~ 0
746       ] ~
747     concat

```

```

748     }
749   }

```

(End definition for `_draw_backend_cm:nnnn`.)

`_draw_backend_box_use:Nnnnn` Inside a picture `@beginspecial/@endspecial` are active, which is normally a good thing but means that the position and scaling would be off if the box was inserted directly. To deal with that, there are a number of possible approaches. The implementation here was suggested by Tom Rokici (author of `dvips`). We end the current special placement, then set the current point with a literal `[begin]`. As for general literals, we then use the stack to store the current point and move to it. To insert the required transformation, we have to flip the y -axis, once before and once after it. Then we get back to the $\text{T}_\text{E}\text{X}$ 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).

```

750 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
751 {
752   \_draw_backend_literal:n { @endspecial }
753   \_draw_backend_literal:n { [end] }
754   \_draw_backend_literal:n { [begin] }
755   \_draw_backend_literal:n { save }
756   \_draw_backend_literal:n { currentpoint }
757   \_draw_backend_literal:n { currentpoint~translate }
758   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
759   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
760   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
761   \_draw_backend_literal:n { neg~exch~neg~exch~translate }
762   \_draw_backend_literal:n { [end] }
763   \hbox_overlap_right:n { \box_use:N #1 }
764   \_draw_backend_literal:n { [begin] }
765   \_draw_backend_literal:n { restore }
766   \_draw_backend_literal:n { [end] }
767   \_draw_backend_literal:n { [begin] }
768   \_draw_backend_literal:n { @beginspecial }
769 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```

770 </dvips>

```

4.2 pdfmode and (x)dvipdfmx

Both `pdfmode` and `(x)dvipdfmx` directly produce PDF output and understand a shared set of specials for drawing commands.

```

771 <*dvipdfmx | pdfmode | xdvipdfmx>

```

4.2.1 Drawing

`_draw_backend_literal:n` Pass data through using a dedicated interface.

```

\_draw_backend_literal:x
772 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_pdf:n
773 \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: 774 \cs_new_protected:Npn \_draw_backend_begin:
775 { \_draw_backend_scope_begin: }
776 \cs_new_protected:Npn \_draw_backend_end:
777 { \_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: 778 \cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:
779 \cs_new_eq:NN \_draw_backend_scope_end: \_kernel_backend_scope_end:
```

(End definition for _draw_backend_scope_begin: and _draw_backend_scope_end:.)

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

`_draw_backend_lineto:nn`

`_draw_backend_curveto:nnnnn`
`_draw_backend_rectangle:nnnn`

```
780 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
781 {
782   \_draw_backend_literal:x
783   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
784 }
785 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
786 {
787   \_draw_backend_literal:x
788   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
789 }
790 \cs_new_protected:Npn \_draw_backend_curveto:nnnnn #1#2#3#4#5#6
791 {
792   \_draw_backend_literal:x
793   {
794     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
795     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
796     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
797     c
798   }
799 }
800 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
801 {
802   \_draw_backend_literal:x
803   {
804     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
805     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
806     re
807   }
808 }
```

(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: 809 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
\g__draw_draw_eor_bool 810 { \bool_gset_true:N \g__draw_draw_eor_bool }
811 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
812 { \bool_gset_false:N \g__draw_draw_eor_bool }
813 \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:` 814 `\cs_new_protected:Npn _draw_backend_closepath:`
`_draw_backend_closestroke:` 815 `{ _draw_backend_literal:n { h } }`
`_draw_backend_fill:` 816 `\cs_new_protected:Npn _draw_backend_stroke:`
`_draw_backend_fillstroke:` 817 `{ _draw_backend_literal:n { S } }`
`_draw_backend_clip:` 818 `\cs_new_protected:Npn _draw_backend_closestroke:`
`_draw_backend_discardpath:` 819 `{ _draw_backend_literal:n { s } }`
820 `\cs_new_protected:Npn _draw_backend_fill:`
821 `{`
822 `_draw_backend_literal:x`
823 `{ f \bool_if:NT \g__draw_draw_eor_bool * }`
824 `}`
825 `\cs_new_protected:Npn _draw_backend_fillstroke:`
826 `{`
827 `_draw_backend_literal:x`
828 `{ B \bool_if:NT \g__draw_draw_eor_bool * }`
829 `}`
830 `\cs_new_protected:Npn _draw_backend_clip:`
831 `{`
832 `_draw_backend_literal:x`
833 `{ W \bool_if:NT \g__draw_draw_eor_bool * }`
834 `}`
835 `\cs_new_protected:Npn _draw_backend_discardpath:`
836 `{ _draw_backend_literal:n { n } }`

(End definition for `_draw_backend_closepath:` and others.)

`_draw_backend_dash_pattern:nn` Converting paths to output is again a case of mapping directly to PDF operations.
`_draw_backend_dash:n` 837 `\cs_new_protected:Npn _draw_backend_dash_pattern:nn #1#2`
`_draw_backend_linewidth:n` 838 `{`
`_draw_backend_miterlimit:n` 839 `_draw_backend_literal:x`
`_draw_backend_cap_butt:` 840 `{`
`_draw_backend_cap_round:` 841 `[`
`_draw_backend_cap_rectangle:` 842 `\exp_args:Nf \use:n`
`_draw_backend_join_miter:` 843 `{ \clist_map_function:nN {#1} _draw_backend_dash:n }`
`_draw_backend_join_round:` 844 `] ~`
`_draw_backend_join_bevel:` 845 `\dim_to_decimal_in_bp:n {#2} ~ d`
846 `}`
847 `}`
848 `\cs_new:Npn _draw_backend_dash:n #1`
849 `{ ~ \dim_to_decimal_in_bp:n {#1} }`
850 `\cs_new_protected:Npn _draw_backend_linewidth:n #1`
851 `{`
852 `_draw_backend_literal:x`
853 `{ \dim_to_decimal_in_bp:n {#1} ~ w }`
854 `}`
855 `\cs_new_protected:Npn _draw_backend_miterlimit:n #1`
856 `{ _draw_backend_literal:x { \fp_eval:n {#1} ~ M } }`
857 `\cs_new_protected:Npn _draw_backend_cap_butt:`
858 `{ _draw_backend_literal:n { 0 ~ J } }`
859 `\cs_new_protected:Npn _draw_backend_cap_round:`

```

860 { \_draw_backend_literal:n { 1 ~ J } }
861 \cs_new_protected:Npn \_draw_backend_cap_rectangle:
862 { \_draw_backend_literal:n { 2 ~ J } }
863 \cs_new_protected:Npn \_draw_backend_join_miter:
864 { \_draw_backend_literal:n { 0 ~ j } }
865 \cs_new_protected:Npn \_draw_backend_join_round:
866 { \_draw_backend_literal:n { 1 ~ j } }
867 \cs_new_protected:Npn \_draw_backend_join_bevel:
868 { \_draw_backend_literal:n { 2 ~ j } }

```

(End definition for _draw_backend_dash_pattern:nn and others.)

Color has to be split between (x)dvipdfmx and the PDF engines as there is no color stack for fill/stroke separation in the former.

```

\_draw_backend_color_fill_cmyk:nnnn
\_draw_backend_color_stroke_cmyk:nnnn
  \_draw_backend_color_fill_gray:n
  \_draw_backend_color_stroke_gray:n
  \_draw_backend_color_fill_rgb:nnn
  \_draw_backend_color_stroke_rgb:nnn
  \_draw_backend_color_select:n
  \_draw_backend_color_select:x
\_draw_backend_color_reset:
869 \cs_new_protected:Npn \_draw_backend_color_fill_cmyk:nnnn #1#2#3#4
870 {
871   \_draw_backend_color_select:x
872   {
873     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
874     \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
875     k
876   }
877 }
878 \cs_new_protected:Npn \_draw_backend_color_stroke_cmyk:nnnn #1#2#3#4
879 {
880   \_draw_backend_color_select:x
881   {
882     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
883     \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
884     k
885   }
886 }
887 \cs_new_protected:Npn \_draw_backend_color_fill_gray:n #1
888 { \_draw_backend_color_select:x { \fp_eval:n {#1} ~ g } }
889 \cs_new_protected:Npn \_draw_backend_color_stroke_gray:n #1
890 { \_draw_backend_color_select:x { \fp_eval:n {#1} ~ G } }
891 \cs_new_protected:Npn \_draw_backend_color_fill_rgb:nnn #1#2#3
892 {
893   \_draw_backend_color_select:x
894   { \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} ~ rg }
895 }
896 \cs_new_protected:Npn \_draw_backend_color_stroke_rgb:nnn #1#2#3
897 {
898   \_draw_backend_color_select:x
899   { \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} ~ RG }
900 }
901 <*pdfmode>
902 \cs_new_protected:Npn \_draw_backend_color_select:n #1
903 {
904   \cs_if_exist:NTF \tex_pdfextension:D
905   { \tex_pdfextension:D colorstack }
906   { \tex_pdfcolorstack:D }
907   \exp_not:N \l__kernel_color_stack_int push {#1}
908   \group_insert_after:N \exp_not:N \_draw_backend_color_reset:

```



```

909 }
910 \cs_new_protected:Npx \__draw_backend_color_reset:
911 {
912   \cs_if_exist:NTF \tex_pdfextension:D
913     { \tex_pdfextension:D colorstack }
914     { \tex_pdfcolorstack:D }
915     \exp_not:N \l__kernel_color_stack_int pop \scan_stop:
916 }
917 </pdfmode>
918 <*dvipdfmx | xdvipdfmx>
919 \cs_new_eq:MN \__draw_backend_color_select:n \__kernel_backend_literal_pdf:n
920 </dvipdfmx | xdvipdfmx>
921 \cs_generate_variant:Nn \__draw_backend_color_select:n { x }

```

(End definition for __draw_backend_color_fill_cmyk:nnnn and others.)

__draw_backend_cm:nnnn
 __draw_backend_cm_aux:nnnn

Another split here between pdfmode and (x)dvipdfmx. In the former, we have a direct method to maintain alignment: the backend can use a matrix itself. For (x)dvipdfmx, 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 (x)dvipdfmx, but as a matched pair so not suitable for the “stand alone” transformation set up here.)

```

922 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
923 {
924 <*pdfmode>
925   \__kernel_backend_matrix:x
926   {
927     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
928     \fp_eval:n {#3} ~ \fp_eval:n {#4}
929   }
930 </pdfmode>
931 <*dvipdfmx | xdvipdfmx>
932   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
933   \__draw_backend_cm_aux:nnnn
934 </dvipdfmx | xdvipdfmx>
935 }
936 <*dvipdfmx | xdvipdfmx>
937 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
938 {
939   \__kernel_backend_literal:x
940   {
941     x:rotate~
942     \fp_compare:nNnTF {#1} = \c_zero_fp
943     { 0 }
944     { \fp_eval:n { round ( -#1 , 5 ) } }
945   }
946   \__kernel_backend_literal:x
947   {
948     x:scale~
949     \fp_eval:n { round ( #2 , 5 ) } ~
950     \fp_eval:n { round ( #3 , 5 ) }
951   }
952   \__kernel_backend_literal:x
953   {

```

```

954     x:rotate~
955     \fp_compare:nNnTF {#4} = \c_zero_fp
956       { 0 }
957       { \fp_eval:n { round ( -#4 , 5 ) } }
958   }
959 }
960 </dvipdfmx | xdvipdfmx>

```

(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.

```

961 <*dvipdfmx | xdvipdfmx>
962 \cs_new_protected:Npn \_draw_backend_cm_decompose:nnnnN #1#2#3#4#5
963   {
964     \use:x
965     {
966       \_draw_backend_cm_decompose_auxi:nnnnN
967       { \fp_eval:n { (#1 + #4) / 2 } }
968       { \fp_eval:n { (#1 - #4) / 2 } }
969       { \fp_eval:n { (#3 + #2) / 2 } }
970       { \fp_eval:n { (#3 - #2) / 2 } }
971     }
972     #5
973   }
974 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
975   {
976     \use:x

```

```

977     {
978       \__draw_backend_cm_decompose_auxii:nnnnN
979         { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
980         { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
981         { \fp_eval:n { atand ( #3 , #2 ) } }
982         { \fp_eval:n { atand ( #4 , #1 ) } }
983     }
984     #5
985 }
986 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
987 {
988   \use:x
989   {
990     \__draw_backend_cm_decompose_auxiii:nnnnN
991       { \fp_eval:n { ( #4 - #3 ) / 2 } }
992       { \fp_eval:n { ( #1 + #2 ) / 2 } }
993       { \fp_eval:n { ( #1 - #2 ) / 2 } }
994       { \fp_eval:n { ( #4 + #3 ) / 2 } }
995   }
996   #5
997 }
998 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
999 {
1000   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1001     { #5 {#1} {#2} {#3} {#4} }
1002     { #5 {#1} {#3} {#2} {#4} }
1003 }
1004 \</dvi>

```

(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.

```

1005 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1006 {
1007   \__kernel_backend_scope_begin:
1008   \<pdfmode>
1009   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1010   \</pdfmode>
1011   \<dvi>
1012   \__kernel_backend_literal:x
1013   {
1014     pdf:btrans-matrix~
1015     \fp_eval:n {#2} ~ \fp_eval:n {#3} ~
1016     \fp_eval:n {#4} ~ \fp_eval:n {#5} ~
1017     0 ~ 0
1018   }
1019   \</dvi>
1020   \hbox_overlap_right:n { \box_use:N #1 }
1021   \<dvi>
1022   \__kernel_backend_literal:n { pdf:etrans }

```

```

1023 </dvipdfmx | xdvipdfmx>
1024   \__kernel_backend_scope_end:
1025 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1026 </dvipdfmx | pdfmode | xdvipdfmx>

```

4.3 dvisvgm backend

```

1027 <*dvisvgm>

```

The same as the more general literal call.

```

\__draw_backend_literal:n The same as the more general literal call.
\__draw_backend_literal:x
1028 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1029 \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 done inside a scope, which as described below

```

\__draw_backend_end:
1030 \cs_new_protected:Npn \__draw_backend_begin:
1031 {
1032   \__draw_backend_scope_begin:
1033   \__draw_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1034 }
1035 \cs_new_protected:Npn \__draw_backend_end:
1036 { \__draw_backend_scope_end: }

```

(End definition for __draw_backend_begin: and __draw_backend_end:.)

__draw_backend_scope_begin: Several settings that with other backends are “stand alone” have to be given as part of a scope in SVG. As a result, there is a need to provide a mechanism to automatically close these extra scopes. That is done using a dedicated function and a pair of tracking variables. Within each graphics scope we use a global variable to do the work, with a group used to save the value between scopes. The result is that no direct action is needed when creating a scope.

```

\__draw_backend_scope_end:
\__draw_backend_scope:n
\__draw_backend_scope:x
\g__draw_draw_scope_int
\l__draw_draw_scope_int
1037 \cs_new_protected:Npn \__draw_backend_scope_begin:
1038 {
1039   \int_set_eq:NN
1040     \l__draw_draw_scope_int
1041     \g__draw_draw_scope_int
1042   \group_begin:
1043     \int_gzero:N \g__draw_draw_scope_int
1044 }
1045 \cs_new_protected:Npn \__draw_backend_scope_end:
1046 {
1047   \prg_replicate:nn
1048     { \g__draw_draw_scope_int }
1049     { \__draw_backend_literal:n { </g> } }
1050   \group_end:
1051   \int_gset_eq:NN
1052     \g__draw_draw_scope_int
1053     \l__draw_draw_scope_int
1054 }
1055 \cs_new_protected:Npn \__draw_backend_scope:n #1

```

```

1056 {
1057   \_draw_backend_literal:n { <g- #1 > }
1058   \int_gincr:N \g__draw_draw_scope_int
1059 }
1060 \cs_generate_variant:Nn \_draw_backend_scope:n { x }
1061 \int_new:N \g__draw_draw_scope_int
1062 \int_new:N \l__draw_draw_scope_int

```

(End definition for _draw_backend_scope_begin: and others.)

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

```

\_draw_backend_lineto:nn
  \_draw_backend_rectangle:nnnn
  \_draw_backend_curveto:nnnnnn
  \_draw_backend_add_to_path:n
\g__draw_draw_path_tl
1063 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1064 {
1065   \_draw_backend_add_to_path:n
1066   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1067 }
1068 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1069 {
1070   \_draw_backend_add_to_path:n
1071   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1072 }
1073 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1074 {
1075   \_draw_backend_add_to_path:n
1076   {
1077     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1078     h ~ \dim_to_decimal:n {#3} ~
1079     v ~ \dim_to_decimal:n {#4} ~
1080     h ~ \dim_to_decimal:n { -#3 } ~
1081     Z
1082   }
1083 }
1084 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1085 {
1086   \_draw_backend_add_to_path:n
1087   {
1088     C ~
1089     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1090     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1091     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1092   }
1093 }
1094 \cs_new_protected:Npn \_draw_backend_add_to_path:n #1
1095 {
1096   \tl_gset:Nx \g__draw_draw_path_tl
1097   {
1098     \g__draw_draw_path_tl
1099     \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1100     #1
1101   }

```

```

1102 }
1103 \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:
1104 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1105 { \__draw_backend_scope:n { fill-rule="evenodd" } }
1106 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1107 { \__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:
1108 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_clip: 1109 { \__draw_backend_add_to_path:n { Z } }
\__draw_backend_discardpath: 1110 \cs_new_protected:Npn \__draw_backend_path:n #1
\g__draw_draw_clip_bool 1111 {
\g__draw_draw_path_int 1112 \bool_if:NTF \g__draw_draw_clip_bool
1113 {
1114 \int_gincr:N \g__draw_clip_path_int
1115 \__draw_backend_literal:x
1116 {
1117 < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1118 { ?nl }
1119 <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1120 </clipPath > { ? nl }
1121 <
1122 use~xlink:href =
1123 "\c_hash_str l3path \int_use:N \g__draw_path_int " ~
1124 #1
1125 />
1126 }
1127 \__draw_backend_scope:x
1128 {
1129 clip-path =
1130 "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1131 }
1132 }
1133 {
1134 \__draw_backend_literal:x
1135 { <path ~ d=" \g__draw_draw_path_tl " ~ #1 /> }
1136 }
1137 \tl_gclear:N \g__draw_draw_path_tl
1138 \bool_gset_false:N \g__draw_draw_clip_bool
1139 }
1140 \int_new:N \g__draw_path_int
1141 \cs_new_protected:Npn \__draw_backend_stroke:
1142 { \__draw_backend_path:n { style="fill:none" } }
1143 \cs_new_protected:Npn \__draw_backend_closestroke:

```

```

1144 {
1145   \_draw_backend_closepath:
1146   \_draw_backend_stroke:
1147 }
1148 \cs_new_protected:Npn \_draw_backend_fill:
1149 { \_draw_backend_path:n { style="stroke:none" } }
1150 \cs_new_protected:Npn \_draw_backend_fillstroke:
1151 { \_draw_backend_path:n { } }
1152 \cs_new_protected:Npn \_draw_backend_clip:
1153 { \bool_gset_true:N \g__draw_draw_clip_bool }
1154 \bool_new:N \g__draw_draw_clip_bool
1155 \cs_new_protected:Npn \_draw_backend_discardpath:
1156 {
1157   \bool_if:NT \g__draw_draw_clip_bool
1158   {
1159     \int_gincr:N \g__draw_clip_path_int
1160     \_draw_backend_literal:x
1161     {
1162       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1163       { ?nl }
1164       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1165       < /clipPath >
1166     }
1167     \_draw_backend_scope:x
1168     {
1169       clip-path =
1170       "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1171     }
1172   }
1173   \tl_gclear:N \g__draw_draw_path_tl
1174   \bool_gset_false:N \g__draw_draw_clip_bool
1175 }

```

(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_but:
\_draw_backend_cap_round:
  \_draw_backend_cap_rectangle:
\_draw_backend_join_miter:
\_draw_backend_join_round:
\_draw_backend_join_bevel:
1176 \cs_new_protected:Npn \_draw_backend_dash_pattern:nn #1#2
1177 {
1178   \use:x
1179   {
1180     \_draw_backend_dash_aux:nn
1181     { \clist_map_function:nn {#1} \_draw_backend_dash:n }
1182     { \dim_to_decimal:n {#2} }
1183   }
1184 }
1185 \cs_new:Npn \_draw_backend_dash:n #1
1186 { , \dim_to_decimal_in_bp:n {#1} }
1187 \cs_new_protected:Npn \_draw_backend_dash_aux:nn #1#2
1188 {
1189   \_draw_backend_scope:x
1190   {
1191     stroke-dasharray =
1192     "

```

```

1193         \tl_if_empty:oTF { \use_none:n #1 }
1194         { none }
1195         { \use_none:n #1 }
1196     " ~
1197     stroke-offset=" #2 "
1198 }
1199 }
1200 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1201 { \__draw_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1202 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1203 { \__draw_backend_scope:x { stroke-miterlimit=" \fp_eval:n {#1} " } }
1204 \cs_new_protected:Npn \__draw_backend_cap_but:
1205 { \__draw_backend_scope:n { stroke-linecap="butt" } }
1206 \cs_new_protected:Npn \__draw_backend_cap_round:
1207 { \__draw_backend_scope:n { stroke-linecap="round" } }
1208 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1209 { \__draw_backend_scope:n { stroke-linecap="square" } }
1210 \cs_new_protected:Npn \__draw_backend_join_miter:
1211 { \__draw_backend_scope:n { stroke-linejoin="miter" } }
1212 \cs_new_protected:Npn \__draw_backend_join_round:
1213 { \__draw_backend_scope:n { stroke-linejoin="round" } }
1214 \cs_new_protected:Npn \__draw_backend_join_bevel:
1215 { \__draw_backend_scope:n { stroke-linejoin="bevel" } }

```

(End definition for __draw_backend_dash_pattern:nn and others.)

_draw_backend_color_fill_cmyk:nmnn SVG fill color has to be covered outside of the stack, as for dvips. Here, we are only allowed RGB colors so there is some conversion to do.

```

\_draw_backend_color_fill_gray:n 1216 \cs_new_protected:Npn \__draw_backend_color_fill_cmyk:nmnn #1#2#3#4
\_draw_backend_color_stroke_cmyk:nmnn {
\_draw_backend_color_fill_gray:n 1217 {
\_draw_backend_color_stroke_gray:n 1218 \use:x
\_draw_backend_color_fill_rgb:nmnn 1219 {
\_draw_backend_color_stroke_rgb:nmnn 1220 \__draw_backend_color_fill:nmnn
\_draw_backend_color_fill:nmnn 1221 { \fp_eval:n { -100 * ( (#1) * ( 1 - (#4) ) - 1 ) } }
1222 { \fp_eval:n { -100 * ( (#2) * ( 1 - (#4) ) + #4 - 1 ) } }
1223 { \fp_eval:n { -100 * ( (#3) * ( 1 - (#4) ) + #4 - 1 ) } }
1224 }
1225 }
1226 \cs_new_protected:Npn \__draw_backend_color_stroke_cmyk:nmnn #1#2#3#4
1227 {
1228 \__draw_backend_select:x
1229 {
1230 cmyk~
1231 \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
1232 \fp_eval:n {#3} ~ \fp_eval:n {#4}
1233 }
1234 }
1235 \cs_new_protected:Npn \__draw_backend_color_fill_gray:n #1
1236 {
1237 \use:x
1238 {
1239 \__draw_backend_color_gray_aux:n
1240 { \fp_eval:n { 100 * (#1) } }
1241 }

```



```

1242 }
1243 \cs_new_protected:Npn \__draw_backend_color_gray_aux:n #1
1244 { \__draw_backend_color_fill:nnn {#1} {#1} {#1} }
1245 \cs_new_protected:Npn \__draw_backend_color_stroke_gray:n #1
1246 { \__draw_backend_select:x { gray~ \fp_eval:n {#1} } }
1247 \cs_new_protected:Npn \__draw_backend_color_fill_rgb:nnn #1#2#3
1248 {
1249   \use:x
1250   {
1251     \__draw_backend_color_fill:nnn
1252     { \fp_eval:n { 100 * (#1) } }
1253     { \fp_eval:n { 100 * (#2) } }
1254     { \fp_eval:n { 100 * (#3) } }
1255   }
1256 }
1257 \cs_new_protected:Npn \__draw_backend_color_fill:nnn #1#2#3
1258 {
1259   \__draw_backend_scope:x
1260   {
1261     fill =
1262     "
1263     rgb
1264     (
1265       #1 \c_percent_str ,
1266       #2 \c_percent_str ,
1267       #3 \c_percent_str
1268     )
1269     "
1270   }
1271 }
1272 \cs_new_protected:Npn \__draw_backend_color_stroke_rgb:nnn #1#2#3
1273 {
1274   \__draw_backend_select:x
1275   { rgb~ \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} }
1276 }

```

(End definition for `__draw_backend_color_fill_cmyk:nnnn` and others.)

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

```

1277 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1278 {
1279   \__draw_backend_scope:n
1280   {
1281     transform =
1282     "
1283     matrix
1284     (
1285       \fp_eval:n {#1} , \fp_eval:n {#2} ,
1286       \fp_eval:n {#3} , \fp_eval:n {#4} ,
1287       Opt , Opt
1288     )
1289     "
1290   }
1291 }

```

(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.

```
1292 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1293 {
1294   \_kernel_backend_scope_begin:
1295   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1296   \_kernel_backend_literal_svg:n
1297   {
1298     < g~
1299     stroke="none"~
1300     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1301     >
1302   }
1303   \box_set_wd:Nn #1 { Opt }
1304   \box_set_ht:Nn #1 { Opt }
1305   \box_set_dp:Nn #1 { Opt }
1306   \box_use:N #1
1307   \_kernel_backend_literal_svg:n { </g> }
1308   \_kernel_backend_scope_end:
1309 }
```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```
1310 </dvisvgm>
1311 </initex | package>
```

5 I3backend-graphics Implementation

```
1312 <*initex | package>
1313 <@=graphics>
```

5.1 dvips backend

```
1314 <*dvips>
```

`_graphics_backend_getbb_eps:n` Simply use the generic function.

```
1315 <*initex>
1316 \use:n
1317 </initex>
1318 <*package>
1319 \AtBeginDocument
1320 </package>
1321 { \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.

```
1322 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1323 {
1324   \_kernel_backend_literal:x
1325   {
1326     PSfile = #1 \c_space_tl
1327     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
```

```

1328         lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1329         urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1330         ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1331     }
1332 }

```

(End definition for `__graphics_backend_include_eps:n`.)

```
1333 </dvips>
```

5.2 pdfmode backend

```
1334 <*pdfmode>
```

```
\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.

```
1335 \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.

```
1336 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
```

```

1337 {
1338     \int_zero:N \l_graphics_page_int
1339     \tl_clear:N \l_graphics_pagebox_tl
1340     \tl_set:Nx \l__graphics_graphics_attr_tl
1341     {
1342         \tl_if_empty:NF \l_graphics_decodearray_tl
1343         { :D \l_graphics_decodearray_tl }
1344         \bool_if:NT \l_graphics_interpolate_bool
1345         { :I }
1346     }
1347     \tl_clear:N \l__graphics_graphics_attr_tl
1348     \__graphics_backend_getbb_auxi:n {#1}
1349 }

```

```
1350 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
```

```
1351 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
```

```

1352 {
1353     \tl_clear:N \l_graphics_decodearray_tl
1354     \bool_set_false:N \l_graphics_interpolate_bool
1355     \tl_set:Nx \l__graphics_graphics_attr_tl
1356     {
1357         : \l_graphics_pagebox_tl
1358         \int_compare:nNnT \l_graphics_page_int > 1
1359         { :P \int_use:N \l_graphics_page_int }
1360     }
1361     \__graphics_backend_getbb_auxi:n {#1}
1362 }

```

```

1363 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1364 {
1365   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1366   { \__graphics_backend_getbb_auxii:n {#1} }
1367 }
1368 % \begin{macrocode}
1369 % Measuring the graphic is done by boxing up: for PDF graphics we could
1370 % use \tex_pdfximagebbox:D, but if doesn't work for other types.
1371 % As the box always starts at $(0,0)$ there is no need to worry about
1372 % the lower-left position.
1373 % \begin{macrocode}
1374 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1375 {
1376   \tex_immediate:D \tex_pdfximage:D
1377   \bool_lazy_or:nnT
1378     { \l__graphics_interpolate_bool }
1379     { ! \tl_if_empty_p:N \l__graphics_decodearray_tl }
1380     {
1381       attr ~
1382       {
1383         \tl_if_empty:NF \l__graphics_decodearray_tl
1384         { /Decode~[ \l__graphics_decodearray_tl ] }
1385         \bool_if:NT \l__graphics_interpolate_bool
1386         { /Interpolate~true }
1387       }
1388     }
1389   \int_compare:nNnT \l__graphics_page_int > 0
1390     { page ~ \int_use:N \l__graphics_page_int }
1391   \tl_if_empty:NF \l__graphics_pagebox_tl
1392     { \l__graphics_pagebox_tl }
1393   {#1}
1394   \hbox_set:Nn \l__graphics_internal_box
1395     { \tex_pdfrefximage:D \tex_pdflastximage:D }
1396   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1397   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1398   \int_const:cn { c__graphics_graphics_#1 \l__graphics_graphics_attr_tl _int }
1399     { \tex_the:D \tex_pdflastximage:D }
1400   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1401 }

```

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

`__graphics_backend_include_jpg: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.

```

1402 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1403 {
1404   \tex_pdfrefximage:D
1405   \int_use:c { c__graphics_graphics_#1 \l__graphics_graphics_attr_tl _int }
1406 }
1407 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1408 \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 EPS graphics may be included in pdfmode by conversion to PDF: this requires restricted
\__graphics_backend_getbb_eps:nm shell escape. Modelled on the epstopdf LATEX 2ε package, but simplified, conversion takes
\__graphics_backend_include_eps:n place here if we have shell access.
\l__graphics_backend_dir_str 1409 \sys_if_shell:T
\l__graphics_backend_name_str 1410 {
\l__graphics_backend_ext_str 1411 \str_new:N \l__graphics_backend_dir_str
1412 \str_new:N \l__graphics_backend_name_str
1413 \str_new:N \l__graphics_backend_ext_str
1414 \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1415 {
1416 \file_parse_full_name:nNNN {#1}
1417 \l__graphics_backend_dir_str
1418 \l__graphics_backend_name_str
1419 \l__graphics_backend_ext_str
1420 \exp_args:Nx \__graphics_backend_getbb_eps:nn
1421 {
1422 \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1423 -converted-to.pdf
1424 }
1425 {#1}
1426 }
1427 \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1428 {
1429 \file_compare_timestamp:nNnT {#2} > {#1}
1430 {
1431 \sys_shell_now:n
1432 { repstopdf ~ #2 ~ #1 }
1433 }
1434 \tl_set:Nn \l_graphics_name_tl {#1}
1435 \__graphics_backend_getbb_pdf:n {#1}
1436 }
1437 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1438 {
1439 \file_parse_full_name:nNNN {#1}
1440 \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1441 \exp_args:Nx \__graphics_backend_include_pdf:n
1442 {
1443 \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1444 -converted-to.pdf
1445 }
1446 }
1447 }
(End definition for \__graphics_backend_getbb_eps:n and others.)
1448 </pdfmode>

```

5.3 dvipdfmx backend

```

1449 <*dvipdfmx |xdvipdfmx>
\__graphics_backend_getbb_eps:n Simply use the generic functions: only for dvipdfmx in the extraction cases.
\__graphics_backend_getbb_jpg:n 1450 <*initex>
\__graphics_backend_getbb_pdf:n 1451 \use:n
\__graphics_backend_getbb_png:n 1452 </initex>

```

```

1453 <*package>
1454 \AtBeginDocument
1455 </package>
1456 { \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n }
1457 <*dvipdfmx>
1458 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1459 {
1460   \int_zero:N \l_graphics_page_int
1461   \tl_clear:N \l_graphics_pagebox_tl
1462   \graphics_extract_bb:n {#1}
1463 }
1464 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1465 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1466 {
1467   \tl_clear:N \l_graphics_decodearray_tl
1468   \bool_set_false:N \l_graphics_interpolate_bool
1469   \graphics_extract_bb:n {#1}
1470 }
1471 </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.

```

1472 \int_new:N \g__graphics_track_int

```

(End definition for `\g__graphics_track_int`.)

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

```

\__graphics_backend_include_jpg:n
\__graphics_backend_include_pdf:n
\__graphics_backend_include_png:n
\__graphics_backend_include_auxi:nn
\__graphics_backend_include_auxii:nmm
\__graphics_backend_include_auxiii:nmm
1473 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1474 {
1475   \__kernel_backend_literal:x
1476   {
1477     PSfile = #1 \c_space_tl
1478     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1479     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1480     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1481     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1482   }
1483 }
1484 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1485 { \__graphics_backend_include_auxi:nn {#1} { image } }
1486 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
1487 <*dvipdfmx>
1488 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1489 { \__graphics_backend_include_auxi:nn {#1} { epdf } }
1490 </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.

```

1491 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2

```

```

1492 {
1493   \_graphics_backend_include_auxii:xnn
1494   {
1495     \tl_if_empty:NF \l_graphics_pagebox_tl
1496     { : \l_graphics_pagebox_tl }
1497     \int_compare:nNnT \l_graphics_page_int > 1
1498     { :P \int_use:N \l_graphics_page_int }
1499     \tl_if_empty:NF \l_graphics_decodearray_tl
1500     { :D \l_graphics_decodearray_tl }
1501     \bool_if:NT \l_graphics_interpolate_bool
1502     { :I }
1503   }
1504   {#1} {#2}
1505 }
1506 \cs_new_protected:Npn \_graphics_backend_include_auxii:nnn #1#2#3
1507 {
1508   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1509   {
1510     \__kernel_backend_literal:x
1511     { pdf:useobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1512   }
1513   { \_graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1514 }
1515 \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!

```

1516 \cs_new_protected:Npn \_graphics_backend_include_auxiii:nnn #1#2#3
1517 {
1518   \int_gincr:N \g__graphics_track_int
1519   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1520   \__kernel_backend_literal:x
1521   {
1522     pdf:#3~
1523     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1524     \int_compare:nNnT \l_graphics_page_int > 1
1525     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1526     \tl_if_empty:NF \l_graphics_pagebox_tl
1527     {
1528       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1529       bbox ~
1530         \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1531         \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1532         \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1533         \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1534     }
1535     (#1)
1536     \bool_lazy_or:nnT
1537     { \l_graphics_interpolate_bool }
1538     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1539     {
1540       <<
1541       \tl_if_empty:NF \l_graphics_decodearray_tl

```

```

1542         { /Decode~[ \l_graphics_decodearray_tl ] }
1543         \bool_if:NT \l_graphics_interpolate_bool
1544         { /Interpolate~true> }
1545     >>
1546     }
1547 }
1548 }

```

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

```
1549 </dviptdpmx | xdvipdpmx>
```

5.4 xdvipdpmx backend

```
1550 <*xdvipdpmx>
```

5.4.1 Images

`_graphics_backend_getbb_jpg:n`
`_graphics_backend_getbb_pdf:n`
`_graphics_backend_getbb_png:n`
`_graphics_backend_getbb_auxi:nN`
`_graphics_backend_getbb_auxii:nNn`
`_graphics_backend_getbb_auxiii:nNnn`
`_graphics_backend_getbb_auxiv:nNnn`
`_graphics_backend_getbb_auxv:nNnn`
`_graphics_backend_getbb_pagebox:w`

For `xdvipdpmx`, there are two primitives that allow us to obtain the bounding box without needing `extractbb`. The only complexity is passing the various minor variations to a common core process. The $X_{\text{E}}\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ primitive omits the text box from the page box specification, so there is also some “trimming” to do here.

```

1551 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1552 {
1553     \int_zero:N \l_graphics_page_int
1554     \tl_clear:N \l_graphics_pagebox_tl
1555     \_graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1556 }
1557 \cs_new_eq:MN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1558 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1559 {
1560     \tl_clear:N \l_graphics_decodearray_tl
1561     \bool_set_false:N \l_graphics_interpolate_bool
1562     \_graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1563 }
1564 \cs_new_protected:Npn \_graphics_backend_getbb_auxi:nN #1#2
1565 {
1566     \int_compare:nNnTF \l_graphics_page_int > 1
1567     { \_graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
1568     { \_graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1569 }
1570 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:nNn #1#2#3
1571 { \_graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1572 \cs_generate_variant:Nn \_graphics_backend_getbb_auxii:nNn { V }
1573 \cs_new_protected:Npn \_graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1574 {
1575     \tl_if_empty:NTF \l_graphics_pagebox_tl
1576     { \_graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1577     { \_graphics_backend_getbb_auxv:nNnn }
1578     {#1} #2 {#3} {#4}
1579 }
1580 \cs_new_protected:Npn \_graphics_backend_getbb_auxiv:nNnn #1#2#3#4#5
1581 {
1582     \use:x
1583     {

```



```

1584     \_graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1585     { #5 ~ \_graphics_backend_getbb_pagebox:w #1 }
1586   }
1587 }
1588 \cs_generate_variant:Nn \_graphics_backend_getbb_auxiv:nnNnn { V }
1589 \cs_new_protected:Npn \_graphics_backend_getbb_auxv:nNnn #1#2#3#4
1590 {
1591   \graphics_bb_restore:nF {#1#3}
1592   { \_graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1593 }
1594 \cs_new_protected:Npn \_graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1595 {
1596   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1597   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1598   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1599   \graphics_bb_save:n {#1#3}
1600 }
1601 \cs_new:Npn \_graphics_backend_getbb_pagebox:w #1 box {#1}

```

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

`_graphics_backend_include_pdf:n`
`_graphics_backend_include_bitmap_quote:w`

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

```

1602 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1603 {
1604   \tex_XeTeXpdffile:D
1605   \_graphics_backend_include_pdf_quote:w #1 "#1" \q_stop \c_space_tl
1606   \int_compare:nNnT \l_graphics_page_int > 0
1607     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1608   \exp_after:wN \_graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1609 }
1610 \cs_new:Npn \_graphics_backend_include_pdf_quote:w #1 " #2 " #3 \q_stop
1611 { " #2 " }

```

(End definition for `_graphics_backend_include_pdf:n` and `_graphics_backend_include_bitmap_quote:w`.)

```
1612 </xdvipdfmx>
```

5.5 dvisvgm backend

```
1613 <*dvisvgm>
```

`_graphics_backend_getbb_eps:n`

Simply use the generic function.

```

1614 <*initex>
1615 \use:n
1616 </initex>
1617 <*package>
1618 \AtBeginDocument
1619 </package>
1620 { \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`

```

1621 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1622 {
1623   \int_zero:N \l_graphics_page_int
1624   \tl_clear:N \l_graphics_pagebox_tl
1625   \graphics_extract_bb:n {#1}
1626 }
1627 \cs_new_eq:MN \_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

```

1628 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1629 {
1630   \tl_clear:N \l_graphics_decodearray_tl
1631   \bool_set_false:N \l_graphics_interpolate_bool
1632   \graphics_extract_bb:n {#1}
1633 }

```

(End definition for `_graphics_backend_getbb_pdf:n`.)

`_graphics_backend_include_eps:n` The special syntax is relatively clear here: remember we need PostScript sizes here. (This
`_graphics_backend_include_pdf:n` is the same as the `dvips` code.)
`_graphics_backend_include:nn`

```

1634 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1635 { \_graphics_backend_include:nn { PSfile } {#1} }
1636 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1637 { \_graphics_backend_include:nn { pdffile } {#1} }
1638 \cs_new_protected:Npn \_graphics_backend_include:nn #1#2
1639 {
1640   \_kernel_backend_literal:x
1641   {
1642     #1 = #2 \c_space_tl
1643     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1644     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1645     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1646     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1647   }
1648 }

```

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

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

```

1649 \cs_new_protected:Npn \_graphics_backend_include_png:n #1
1650 {
1651   \_kernel_backend_literal:x
1652   {
1653     dvisvgm:img~
1654     \dim_to_decimal:n { \l_graphics_ury_dim } ~
1655     \dim_to_decimal:n { \l_graphics_ury_dim } ~

```

```

1656         \_graphics_backend_include_bitmap_quote:w #1 " #1 " \q_stop
1657     }
1658 }
1659 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_png:n
1660 \cs_new:Npn \_graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \q_stop
1661 { " #2 " }

```

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

```

1662 </dvisvgm>
1663 </initex | package>

```

6 I3backend-pdf Implementation

```

1664 <*initex | package>
1665 <@=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.

```

\_pdf_internal_box
1666 \box_new:N \\_pdf_internal_box
(End definition for \\_pdf_internal_box.)

```

6.2 dvips backend

```

1667 <*dvips>
\_pdf_backend_pdfmark:n Used often enough it should be a separate function.
\_pdf_backend_pdfmark:x
1668 \cs_new_protected:Npn \\_pdf_backend_pdfmark:n #1
1669 { \_kernel_backend_postscript:n { mark #1 ~ pdfmark } }
1670 \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
1671 \cs_new_protected:Npn \\_pdf_backend_catalog_gput:nn #1#2
1672 { \_pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
1673 \cs_new_protected:Npn \\_pdf_backend_info_gput:nn #1#2
1674 { \_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 For tracking objects to allow finalisation.
\g__pdf_backend_object_prop 1675 \int_new:N \g__pdf_backend_object_int
                             1676 \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 Tracking objects is similar to dvipdfmx.
\_pdf_backend_object_ref:n 1677 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2
                             1678 {
                             1679   \int_gincr:N \g__pdf_backend_object_int
                             1680   \int_const:cn
                             1681   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
                             1682   { \g__pdf_backend_object_int }
                             1683   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
                             1684 }
                             1685 \cs_new:Npn \_pdf_backend_object_ref:n #1
                             1686 { { 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: some work to get things right.
\_pdf_backend_object_write:nx 1687 \cs_new_protected:Npn \_pdf_backend_object_write:nn #1#2
\_pdf_backend_object_write_array:nn 1688 {
\_pdf_backend_object_write_dict:nn 1689   \_pdf_backend_pdfmark:x
\_pdf_backend_object_write_stream:nn 1690   {
\_pdf_backend_object_write_stream:nnn 1691     /_objdef ~ \_pdf_backend_object_ref:n {#1}
1692     /type
1693     \str_case_e:nn
1694     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
1695     {
1696       { array } { /array }
1697       { dict } { /dict }
1698       { fstream } { /stream }
1699       { stream } { /stream }
1700     }
1701     /OBJ
1702   }
1703   \use:c
1704   { \_pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
1705   { \_pdf_backend_object_ref:n {#1} } {#2}
1706 }
1707 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
1708 \cs_new_protected:Npn \_pdf_backend_object_write_array:nn #1#2
1709 {
1710   \_pdf_backend_pdfmark:x
1711   { #1 [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
1712 }
1713 \cs_new_protected:Npn \_pdf_backend_object_write_dict:nn #1#2
1714 {
1715   \_pdf_backend_pdfmark:x
1716   { #1 << \exp_not:n {#2} >> /PUT }
1717 }
1718 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nn #1#2

```

```

1719 {
1720   \exp_args:Nx
1721   \__pdf_backend_object_write_stream:nnn {#1} #2
1722 }
1723 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
1724 {
1725   \__kernel_backend_postscript:n
1726   {
1727     [nobreak]
1728     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
1729     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
1730   }
1731 }

```

(End definition for __pdf_backend_object_write:nn and others.)

__pdf_backend_object_now:nn No anonymous objects, so things are done manually.

```

\__pdf_backend_object_now:nx 1732 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
1733 {
1734   \int_gincr:N \g__pdf_backend_object_int
1735   \__pdf_backend_pdfmark:x
1736   {
1737     /_objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
1738     /type
1739     \str_case:nn
1740     {#1}
1741     {
1742       { array } { /array }
1743       { dict } { /dict }
1744       { fstream } { /stream }
1745       { stream } { /stream }
1746     }
1747     /OBJ
1748   }
1749   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
1750   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
1751 }
1752 \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.

```

1753 \cs_new:Npn \__pdf_backend_object_last:
1754 { { pdf.obj \int_use:N \g__pdf_backend_object_int } }

```

(End definition for __pdf_backend_object_last:.)

6.2.3 Annotations

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

pdf.globaldict A small global dictionary for backend use.

```
1755 \_kernel_backend_postscript_header:n
1756 {
1757     true ~ setglobal ~
1758     /pdf.globaldict ~ 4 ~ dict ~ def ~
1759     false ~ setglobal
1760 }
```

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

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

```
pdf.rect.ht 1761 \_kernel_backend_postscript_header:n
1762 {
1763     /pdf.cvs { 65534 ~ string ~ cvs } def
1764     /pdf.dvi.pt { 72.27 ~ mul ~ Resolution ~ div } def
1765     /pdf.pt.dvi { 72.27 ~ div ~ Resolution ~ mul } def
1766     /pdf.rect.ht { dup ~ 1 ~ get ~ neg ~ exch ~ 3 ~ get ~ add } def
1767 }
```

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

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

```
pdf.linkdp.pad 1768 \_kernel_backend_postscript_header:n
pdf.linkht.pad 1769 {
1770     /pdf.linkmargin { 1 ~ pdf.pt.dvi } def
1771     /pdf.linkdp.pad { 0 } def
1772     /pdf.linkht.pad { 0 } def
1773 }
```

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

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

```
pdf.save.linkll 1774 \_kernel_backend_postscript_header:n
pdf.save.linkur 1775 {
pdf.llx 1776     /pdf.rect
pdf.lly 1777     { /Rect [ pdf.llx ~ pdf.lly ~ pdf.urx ~ pdf.ury ] } def
pdf.urx 1778     /pdf.save.ll
pdf.ury 1779     {
1780         currentpoint
1781         /pdf.lly ~ exch ~ def
1782         /pdf.llx ~ exch ~ def
1783     }
1784     def
1785     /pdf.save.ur
1786     {
1787         currentpoint
1788         /pdf.ury ~ exch ~ def
1789         /pdf.urx ~ exch ~ def
1790     }
}
```

```

1791     def
1792 /pdf.save.linkll
1793 {
1794     currentpoint ~
1795     pdf.linkmargin ~ add ~
1796     pdf.linkdp.pad ~ add
1797     /pdf.lly ~ exch ~ def ~
1798     pdf.linkmargin ~ sub
1799     /pdf.llx ~ exch ~ def
1800 }
1801     def
1802 /pdf.save.linkur
1803 {
1804     currentpoint ~
1805     pdf.linkmargin ~ sub ~
1806     pdf.linkht.pad ~ sub
1807     /pdf.ury ~ exch ~ def ~
1808     pdf.linkmargin ~ add
1809     /pdf.urx ~ exch ~ def
1810 }
1811     def
1812 }

```

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

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

```

pdf.dev.x 1813 \_kernel_backend_postscript_header:n
pdf.dev.y 1814 {
pdf.tmpa 1815     /pdf.dest.anchor
pdf.tmpb 1816     {
pdf.tmpc 1817         currentpoint ~ exch ~
pdf.tmpd 1818         pdf.dvi.pt ~ 72 ~ add ~
1819         /pdf.dest.x ~ exch ~ def ~
1820         pdf.dvi.pt ~
1821         vsize ~ 72 ~ sub ~ exch ~ sub ~
1822         /pdf.dest.y ~ exch ~ def
1823     }
1824     def
1825 /pdf.dest.point
1826 { pdf.dest.x ~ pdf.dest.y } def
1827 /pdf.dest2device
1828 {
1829     /pdf.dest.y ~ exch ~ def
1830     /pdf.dest.x ~ exch ~ def ~
1831     matrix ~ currentmatrix ~
1832     matrix ~ defaultmatrix ~
1833     matrix ~ invertmatrix ~
1834     matrix ~ concatmatrix ~
1835     cvx ~ exec
1836     /pdf.dev.y ~ exch ~ def

```

```

1837     /pdf.dev.x ~ exch ~ def
1838     /pdf.tmpd ~ exch ~ def
1839     /pdf.tmpc ~ exch ~ def
1840     /pdf.tmpb ~ exch ~ def
1841     /pdf.tmpa ~ exch ~ def ~
1842     pdf.dest.x ~ pdf.tmpa ~ mul ~
1843     pdf.dest.y ~ pdf.tmpc ~ mul ~ add ~
1844     pdf.dev.x ~ add ~
1845     pdf.dest.x ~ pdf.tmpb ~ mul ~
1846     pdf.dest.y ~ pdf.tmpd ~ mul ~ add ~
1847     pdf.dev.y ~ add
1848   }
1849   def
1850 }

```

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

<pre> pdf.bordertracking pdf.bordertracking.begin pdf.bordertracking.end pdf.leftboundary pdf.rightboundary pdf.brokenlink.rect pdf.brokenlink.skip pdf.brokenlink.dict pdf.bordertracking.endpage pdf.bordertracking.continue pdf.originx pdf.originy </pre>	<p>To know where a breakable link can go, we need to track the boundary rectangle. That can be done by hooking into a and x operations: those names have to be retained. The boundary is stored at the end of the operation. Special effort is needed at the start and end of pages (or rather galleys), such that everything works properly.</p> <pre> 1851 _kernel_backend_postscript_header:n 1852 { 1853 /pdf.bordertracking ~ false ~ def 1854 /pdf.bordertracking.begin 1855 { 1856 SDict ~ /pdf.bordertracking ~ true ~ put ~ 1857 SDict ~ /pdf.leftboundary ~ undef ~ 1858 SDict ~ /pdf.rightboundary ~ undef ~ 1859 /a ~ where 1860 { 1861 /a 1862 { 1863 currentpoint ~ pop ~ 1864 SDict /pdf.rightboundary ~ known ~ dup 1865 { 1866 SDict /pdf.rightboundary ~ get ~ 2 ~ index ~ lt 1867 { not } 1868 if 1869 } 1870 if 1871 { pop } 1872 { SDict ~ exch /pdf.rightboundary ~ exch ~ put } 1873 ifelse ~ 1874 moveto ~ 1875 currentpoint ~ pop ~ 1876 SDict /pdf.leftboundary ~ known ~ dup 1877 { 1878 SDict /pdf.leftboundary ~ get ~ 2 ~ index ~ gt 1879 { not } 1880 if 1881 } 1882 if 1883 { pop } </pre>
---	---


```

1884         { SDict ~ exch /pdf.leftboundary ~ exch ~ put }
1885         ifelse
1886     }
1887     put
1888 }
1889 if
1890 }
1891 def
1892 /pdf.bordertracking.end
1893 {
1894     /a ~ where { /a { moveto } put } if
1895     /x ~ where { /x { 0 ~ exch ~ rmoveto } put } if ~
1896     SDict /pdf.leftboundary ~ known
1897     { pdf.outerbox ~ 0 ~ pdf.leftboundary ~ put }
1898     if ~
1899     SDict /pdf.rightboundary ~ known
1900     { pdf.outerbox ~ 2 ~ pdf.rightboundary ~ put }
1901     if ~
1902     SDict /pdf.bordertracking ~ false ~ put
1903 }
1904 def
1905 /pdf.bordertracking.endpage
1906 {
1907     pdf.bordertracking
1908     {
1909         pdf.bordertracking.end ~
1910         true ~ setglobal ~
1911         pdf.globaldict
1912         /pdf.brokenlink.rect [ pdf.outerbox ~ aload ~ pop ] put ~
1913         pdf.globaldict
1914         /pdf.brokenlink.skip ~ pdf.baselineskip ~ put ~
1915         pdf.globaldict
1916         /pdf.brokenlink.dict ~
1917         pdf.link.dict ~ pdf.cvs ~ put ~
1918         false ~ setglobal ~
1919         mark ~ pdf.link.dict ~ cvx ~ exec ~ /Rect
1920         [
1921             pdf.llx ~
1922             pdf.lly ~
1923             pdf.outerbox ~ 2 ~ get ~ pdf.linkmargin ~ add ~
1924             currentpoint ~ exch ~ pop ~
1925             pdf.outerbox ~ pdf.rect.ht ~ sub ~ pdf.linkmargin ~ sub
1926         ]
1927         /ANN ~ pdf.pdfmark
1928     }
1929     if
1930 }
1931 def
1932 /pdf.bordertracking.continue
1933 {
1934     /pdf.link.dict ~ pdf.globaldict
1935     /pdf.brokenlink.dict ~ get ~ def
1936     /pdf.outerbox ~ pdf.globaldict
1937     /pdf.brokenlink.rect ~ get ~ def

```

```

1938 /pdf.baselineskip ~ pdf.globaldict
1939 /pdf.brokenlink.skip ~ get ~ def ~
1940 pdf.globaldict ~ dup ~ dup
1941 /pdf.brokenlink.dict ~ undef
1942 /pdf.brokenlink.skip ~ undef
1943 /pdf.brokenlink.rect ~ undef ~
1944 currentpoint
1945 /pdf.originy ~ exch ~ def
1946 /pdf.originx ~ exch ~ def
1947 /a ~ where
1948 {
1949 /a
1950 {
1951 moveto ~
1952 SDict ~
1953 begin ~
1954 currentpoint ~ pdf.originy ~ ne ~ exch ~
1955 pdf.originx ~ ne ~ or
1956 {
1957 pdf.save.linkll
1958 /pdf.lly ~
1959 pdf.lly ~ pdf.outerbox ~ 1 ~ get ~ sub ~ def ~
1960 pdf.bordertracking.begin
1961 }
1962 if ~
1963 end
1964 }
1965 put
1966 }
1967 if
1968 /x ~ where
1969 {
1970 /x
1971 {
1972 0 ~ exch ~ rmoveto ~
1973 SDict~
1974 begin ~
1975 currentpoint ~
1976 pdf.originy ~ ne ~ exch ~ pdf.originx ~ ne ~ or
1977 {
1978 pdf.save.linkll
1979 /pdf.lly ~
1980 pdf.lly ~ pdf.outerbox ~ 1 ~ get ~ sub ~ def ~
1981 pdf.bordertracking.begin
1982 }
1983 if ~
1984 end
1985 }
1986 put
1987 }
1988 if
1989 }
1990 def
1991 }

```

(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.

```
1992 \_kernel_backend_postscript_header:n
1993 {
1994 /pdf.breaklink
1995 {
1996 pop ~
1997 counttomark ~ 2 ~ mod ~ 0 ~ eq
1998 {
1999 counttomark /pdf.count ~ exch ~ def
2000 {
2001 pdf.count ~ 0 ~ eq { exit } if ~
2002 counttomark ~ 2 ~ roll ~
2003 1 ~ index ~ /Rect ~ eq
2004 {
2005 dup ~ 4 ~ array ~ copy ~
2006 dup ~ dup ~
2007 1 ~ get ~
2008 pdf.outerbox ~ pdf.rect.ht ~
2009 pdf.linkmargin ~ 2 ~ mul ~ add ~ sub ~
2010 3 ~ exch ~ put ~
2011 dup ~
2012 pdf.outerbox ~ 2 ~ get ~
2013 pdf.linkmargin ~ add ~
2014 2 ~ exch ~ put ~
2015 dup ~ dup ~
2016 3 ~ get ~
2017 pdf.outerbox ~ pdf.rect.ht ~
2018 pdf.linkmargin ~ 2 ~ mul ~ add ~ add ~
2019 1 ~ exch ~ put
2020 /pdf.currentrect ~ exch ~ def ~
2021 pdf.breaklink.write
2022 {
2023 pdf.currentrect ~
2024 dup ~
2025 pdf.outerbox ~ 0 ~ get ~
2026 pdf.linkmargin ~ sub ~
2027 0 ~ exch ~ put ~
2028 dup ~
2029 pdf.outerbox ~ 2 ~ get ~
2030 pdf.linkmargin ~ add ~
2031 2 ~ exch ~ put ~
2032 dup ~ dup ~
2033 1 ~ get ~
2034 pdf.baselineskip ~ add ~
2035 1 ~ exch ~ put ~
2036 dup ~ dup ~
2037 3 ~ get ~
2038 pdf.baselineskip ~ add ~
```

```

2039         3 ~ exch ~ put ~
2040         /pdf.currentrect ~ exch ~ def ~
2041         pdf.breaklink.write
2042     }
2043     1 ~ index ~ 3 ~ get ~
2044     pdf.linkmargin ~ 2 ~ mul ~ add ~
2045     pdf.outerbox ~ pdf.rect.ht ~ add ~
2046     2 ~ index ~ 1 ~ get ~ sub ~
2047     pdf.baselineskip ~ div ~ round ~ cvi ~ 1 ~ sub ~
2048     exch ~
2049     repeat ~
2050     pdf.currentrect ~
2051     dup ~
2052         pdf.outerbox ~ 0 ~ get ~
2053         pdf.linkmargin ~ sub ~
2054         0 ~ exch ~ put ~
2055     dup ~ dup ~
2056         1 ~ get ~
2057         pdf.baselineskip ~ add ~
2058         1 ~ exch ~ put ~
2059     dup ~ dup ~
2060         3 ~ get ~
2061         pdf.baselineskip ~ add ~
2062         3 ~ exch ~ put ~
2063     dup ~ 2 ~ index ~ 2 ~ get ~ 2 ~ exch ~ put
2064     /pdf.currentrect ~ exch ~ def ~
2065     pdf.breaklink.write ~
2066     SDict /pdf.pdfmark.good ~ false ~ put ~
2067     exit
2068 }
2069 { pdf.count ~ 2 ~ sub /pdf.count ~ exch ~ def }
2070 ifelse
2071 }
2072 loop
2073 }
2074 if
2075 /ANN
2076 }
2077 def
2078 /pdf.breaklink.write
2079 {
2080     counttomark ~ 1 ~ sub ~
2081     index /_objdef ~ eq
2082     {
2083         counttomark ~ -2 ~ roll ~
2084         dup ~ wcheck ~
2085         {
2086             readonly ~
2087             counttomark ~ 2 ~ roll
2088         }
2089         { pop ~ pop }
2090     } ifelse
2091 }
2092 if ~

```

```

2093     counttomark ~ 1 ~ add ~ copy ~
2094     pop ~ pdf.currentrect
2095     /ANN ~ pdfmark
2096   }
2097   def
2098 }

```

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

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

```

2099 \_kernel_backend_postscript_header:n
2100 {
2101   /pdf.pdfmark
2102   {
2103     SDict /pdf.pdfmark.good ~ true ~ put ~
2104     dup /ANN ~ eq
2105     {
2106       pdf.pdfmark.store ~
2107       pdf.pdfmark.dict ~
2108       begin ~
2109         Subtype /Link ~ eq ~
2110         currentdict /Rect ~ known ~ and ~
2111         SDict /pdf.outerbox ~ known ~ and ~
2112         SDict /pdf.baselineskip ~ known ~ and ~
2113         {
2114           Rect ~ 3 ~ get ~
2115           pdf.linkmargin ~ 2 ~ mul ~ add ~
2116           pdf.outerbox ~ pdf.rect.ht ~ add ~
2117           Rect ~ 1 ~ get ~ sub ~
2118           pdf.baselineskip ~ div ~ round ~ cvi ~ 0 ~ gt
2119           { pdf.breaklink }
2120           if
2121         }
2122         if ~
2123         end ~
2124         SDict /pdf.outerbox ~ undef ~
2125         SDict /pdf.baselineskip ~ undef ~
2126         currentdict /pdf.pdfmark.dict ~ undef ~
2127       }
2128       if ~
2129       pdf.pdfmark.good
2130       { pdfmark }
2131       { cleartomark }
2132       ifelse
2133     }
2134     def
2135   /pdf.pdfmark.store
2136   {
2137     /pdf.pdfmark.dict ~ 65534 ~ dict ~ def ~
2138     counttomark ~ 1 ~ add ~ copy ~

```

```

2139     pop
2140     {
2141         dup ~ mark ~ eq
2142         {
2143             pop ~
2144             exit
2145         }
2146         {
2147             pdf.pdfmark.dict ~
2148             begin ~ def ~ end
2149         }
2150         ifelse
2151     }
2152     loop
2153 }
2154 def
2155 }

```

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

`\l__pdf_backend_content_box` The content of an annotation.
2156 `\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.
2157 `\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.
2158 `\int_new:N \g__pdf_backend_annotation_int`
(End definition for `\g__pdf_backend_annotation_int`.)

`__pdf_backend_annotation:nxxx` Annotations are objects, but we track them separately. Notably, they are not in the
`__pdf_backend_annotation_aux:nxxx` object data lists. Here, to get the co-ordinates of the annotation, we need to have the
pdf.llx data collected at the PostScript level. That requires a bit of box trickery (effectively a
pdf.lly $\LaTeX 2_{\epsilon}$ picture of zero size). Once the data is collected, use it to set up the annotation
pdf.urx border. There is a split into two parts here to allow an easy way of applying the Adobe
pdf.ury Reader fix.

```

2159 \cs_new_protected:Npn \__pdf_backend_annotation:nxxx #1#2#3#4
2160 {
2161     \__pdf_backend_annotation_aux:nxxx {#1} {#2} {#3} {#4}
2162     \int_gincr:N \g__pdf_backend_object_int
2163     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2164     \__pdf_backend_pdfmark:x
2165     {
2166
2167         /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2168         pdf.rect ~
2169         #4 ~
2170         /ANN
2171     }

```

```

2172 }
2173 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2174 {
2175   \box_move_down:nn {#3}
2176   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2177   \hbox:n {#4}
2178   \box_move_up:nn {#2}
2179   {
2180     \hbox:n
2181     {
2182       \tex_kern:D \dim_eval:n {#1} \scan_stop:
2183       \__kernel_backend_postscript:n { pdf.save.ur }
2184     }
2185   }
2186   \int_gincr:N \g__pdf_backend_object_int
2187   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2188   \__pdf_backend_pdfmark:x
2189   {
2190     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2191     pdf.rect
2192     /ANN
2193   }
2194 }

```

(End definition for __pdf_backend_annotation:nnnn and others. These functions are documented on page ??.)

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

```

2195 \cs_new:Npn \__pdf_backend_annotation_last:
2196 { { 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.

```

2197 \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.

```

2198 \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 fact we need a box.

```

2199 \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.

```

2200 \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.

```

2201 \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.

```
2202 \tl_new:N \l__pdf_breaklink_pdfmark_tl
2203 \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.

```
2204 \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.

```
2205 \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`

`__pdf_backend_link_aux:nw`

`__pdf_backend_link_end:`

`_pdf_backend_link_end_aux:`

`__pdf_backend_link_minima:`

`_pdf_backend_link_outerbox:n`

`__pdf_backend_link_sf_save:`

`_pdf_backend_link_sf_restore:`

`pdf.linkdp.pad`

`pdf.linkht.pad`

`pdf.llx`

`pdf.lly`

`pdf.ury`

`pdf.link.dict`

`pdf.outerbox`

`pdf.baselineskip`

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

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

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

```
2206 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2207 { \__pdf_backend_link_begin:nw { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2208 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
2209 { \__pdf_backend_link_begin:nw {#1#2} }
2210 \cs_new_protected:Npn \__pdf_backend_link_begin:nw #1
2211 {
2212   \bool_if:NF \g__pdf_backend_link_bool
2213     { \__pdf_backend_link_begin_aux:nw {#1} }
2214 }
2215 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2216 {
2217   \bool_gset_true:N \g__pdf_backend_link_bool
2218   \__kernel_backend_postscript:n
2219     { /pdf.link.dict ( #1 ) def }
2220   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2221   \__pdf_backend_link_sf_save:
2222   \mode_if_math:TF
2223     { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2224     { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2225   \hbox_set:Nw \l__pdf_backend_content_box
2226   \__pdf_backend_link_sf_restore:
```



```

2227     \bool_if:NT \g__pdf_backend_link_math_bool
2228     { \c_math_toggle_token }
2229   }
2230 \cs_new_protected:Npn \__pdf_backend_link_end:
2231   {
2232     \bool_if:NT \g__pdf_backend_link_bool
2233     { \__pdf_backend_link_end_aux: }
2234   }
2235 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2236   {
2237     \bool_if:NT \g__pdf_backend_link_math_bool
2238     { \c_math_toggle_token }
2239     \__pdf_backend_link_sf_save:
2240     \hbox_set_end:
2241     \__pdf_backend_link_minima:
2242     \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2243     \exp_args:Nx \__pdf_backend_link_outerbox:n
2244     {
2245     <*initex>
2246       \l_galley_total_left_margin_dim
2247     </initex>
2248     <*package>
2249       \int_if_odd:nTF { \value { page } }
2250       { \oddsidemargin }
2251       { \evensidemargin }
2252     </package>
2253   }
2254   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2255   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2256   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2257   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2258   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2259   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2260   {
2261     \hbox:n
2262     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2263   }
2264   \int_gincr:N \g__pdf_backend_object_int
2265   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2266   \__kernel_backend_postscript:x
2267   {
2268     mark
2269     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2270     \g__pdf_backend_link_dict_tl \c_space_tl
2271     pdf.rect
2272     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2273   }
2274   \__pdf_backend_link_sf_restore:
2275   \bool_gset_false:N \g__pdf_backend_link_bool
2276   }
2277 \cs_new_protected:Npn \__pdf_backend_link_minima:
2278   {
2279   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2280   \__kernel_backend_postscript:x

```

```

2281 {
2282   /pdf.linkdp.pad ~
2283   \dim_to_decimal:n
2284   {
2285     \dim_max:nn
2286     {
2287       \box_dp:N \l__pdf_backend_model_box
2288       - \box_dp:N \l__pdf_backend_content_box
2289     }
2290     { Opt }
2291   } ~
2292   pdf.pt.dvi ~ def
2293   /pdf.linkht.pad ~
2294   \dim_to_decimal:n
2295   {
2296     \dim_max:nn
2297     {
2298       \box_ht:N \l__pdf_backend_model_box
2299       - \box_ht:N \l__pdf_backend_content_box
2300     }
2301     { Opt }
2302   } ~
2303   pdf.pt.dvi ~ def
2304 }
2305 }
2306 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2307 {
2308   \__kernel_backend_postscript:x
2309   {
2310     /pdf.outerbox
2311     [
2312       \dim_to_decimal:n {#1} ~
2313       \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2314     <*initex>
2315       \dim_to_decimal:n { #1 + \l_galley_text_width_dim } ~
2316     </initex>
2317     <*package>
2318       \dim_to_decimal:n { #1 + \textwidth } ~
2319     </package>
2320     \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2321   ]
2322   [ exch { pdf.pt.dvi } forall ] def
2323   /pdf.baselineskip ~
2324   \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2325     { pdf.pt.dvi ~ def }
2326     { pop ~ pop }
2327   ifelse
2328 }
2329 }
2330 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2331 {
2332   \int_gset:Nn \g__pdf_backend_link_sf_int
2333   {
2334     \mode_if_horizontal:TF

```

```

2335         { \tex_spacefactor:D }
2336         { 0 }
2337     }
2338 }
2339 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2340 {
2341     \mode_if_horizontal:T
2342     {
2343         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2344         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2345     }
2346 }

```

(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.

```

2347 <*package>
2348 \use_none:n
2349 {
2350     \cs_if_exist:NT \@makecol@hook
2351     {
2352         \tl_put_right:Nn \@makecol@hook
2353         {
2354             \box_if_empty:NF \@cclv
2355             {
2356                 \vbox_set:Nn \@cclv
2357                 {
2358                     \__kernel_backend_postscript:n
2359                     {
2360                         pdf.globaldict /pdf.brokenlink.rect ~ known
2361                         { pdf.bordertracking.continue }
2362                     }
2363                     if
2364                     \vbox_unpack_drop:N \@cclv
2365                     \__kernel_backend_postscript:n
2366                     { pdf.bordertracking.endpage }
2367                 }
2368             }
2369         }
2370         \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2371         \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2372         \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2373     }
2374 }
2375 </package>

```

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

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

```

2376 \cs_new:Npn \__pdf_backend_link_last:
2377 { { 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.

```
2378 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2379 {
2380   \_kernel_backend_postscript:x
2381   {
2382     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2383   }
2384 }
```

(End definition for `_pdf_backend_link_margin:n`.)

`_pdf_backend_destination:nn` Here, we need to turn the zoom into a scale. We also need to know where the current
`_pdf_backend_destination_rectangle:nn` anchor point actually is: worked out in PostScript. For the rectangle version, we have a bit more PostScript: we need two points.

```
2385 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2386 {
2387   \_kernel_backend_postscript:n { pdf.dest.anchor }
2388   \_pdf_backend_pdfmark:x
2389   {
2390     /View
2391     [
2392       \str_case:nnF {#2}
2393       {
2394         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2395         { fit } { /Fit }
2396         { fitb } { /FitB }
2397         { fitbh } { /FitBH ~ pdf.dest.y }
2398         { fitbv } { /FitBV ~ pdf.dest.x }
2399         { fith } { /FitH ~ pdf.dest.y }
2400         { fitv } { /FitV ~ pdf.dest.x }
2401       }
2402       {
2403         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2404       }
2405     ]
2406     /Dest ( \exp_not:n {#1} ) cvn
2407     /DEST
2408   }
2409 }
2410 \cs_new_protected:Npn \_pdf_backend_destination_rectangle:nn #1#2
2411 {
2412   \group_begin:
2413   \hbox_set:Nn \l__pdf_internal_box {#2}
2414   \box_move_down:nn
2415   { \box_dp:N \l__pdf_internal_box }
2416   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2417   \box_use:N \l__pdf_internal_box
2418   \box_move_up:nn
2419   { \box_ht:N \l__pdf_internal_box }
2420   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ur } } }
2421   \_pdf_backend_pdfmark:n
2422   {
```

```

2423         /View
2424         [
2425             /FitR ~
2426             pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2427             pdf.urx ~ pdf.ury ~ pdf.dest2device
2428         ]
2429         /Dest ( #1 ) cvn
2430         /DEST
2431     }
2432 \group_end:
2433 }

```

(End definition for `_pdf_backend_destination:nn` and `_pdf_backend_destination_rectangle:nn`.)

6.2.4 Structure

`_pdf_backend_compresslevel:n` These are all no-ops.
`_pdf_backend_compress_objects:n`

```

2434 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1 { }
2435 \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` Data not available!
`_pdf_backend_version_minor_gset:n`

```

2436 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
2437 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1 { }

```

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

`_pdf_backend_version_major:` Data not available!
`_pdf_backend_version_minor:`

```

2438 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2439 \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:`

```

2440 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2441 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2442 \cs_new_protected:Npn \_pdf_backend_emc:
2443 { \_pdf_backend_pdfmark:n { /EMC } }

```

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

```

2444 </dvips>

```

6.3 pdfmode backend

2445 \langle *pdfmode)

6.3.1 Annotations

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

```
2446 \cs_new_protected:Npx \_pdf_backend_annotation:nnnn #1#2#3#4
2447 {
2448   \cs_if_exist:NTF \tex_pdfextension:D
2449     { \tex_pdfextension:D annot ~ }
2450     { \tex_pdfannot:D }
2451     width ~ \exp_not:N \dim_eval:n {#1} ~
2452     height ~ \exp_not:N \dim_eval:n {#2} ~
2453     depth ~ \exp_not:N \dim_eval:n {#3} ~
2454     {#4}
2455 }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

`_pdf_backend_annotation_last:` A tiny amount of extra data gets added here.

```
2456 \cs_new:Npx \_pdf_backend_annotation_last:
2457 {
2458   \exp_not:N \int_value:w
2459   \cs_if_exist:NTF \tex_pdffeedback:D
2460     { \exp_not:N \tex_pdffeedback:D lastannot ~ }
2461     { \exp_not:N \tex_pdflastannot:D }
2462   \c_space_tl 0 ~ R
2463 }
```

(End definition for `_pdf_backend_annotation_last:`.)

`_pdf_backend_link_begin_goto:nnw` Links are all created using the same internals.

`_pdf_backend_link_begin_user:nnw`

`_pdf_backend_link_begin:nnnw`

`_pdf_backend_link_end:`

```
2464 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2465 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2466 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2467 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2468 \cs_new_protected:Npx \_pdf_backend_link_begin:nnnw #1#2#3
2469 {
2470   \cs_if_exist:NTF \tex_pdfextension:D
2471     { \tex_pdfextension:D startlink ~ }
2472     { \tex_pdfstartlink:D }
2473     attr {#1}
2474     #2 {#3}
2475 }
2476 \cs_new_protected:Npx \_pdf_backend_link_end:
2477 {
2478   \cs_if_exist:NTF \tex_pdfextension:D
2479     { \tex_pdfextension:D endlink \scan_stop: }
2480     { \tex_pdfendlink:D }
2481 }
```

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

`_pdf_backend_link_last:` Formatted for direct use.

```
2482 \cs_new:Npx \_pdf_backend_link_last:
2483 {
2484   \exp_not:N \int_value:w
2485   \cs_if_exist:NTF \tex_pdffeedback:D
2486     { \exp_not:N \tex_pdffeedback:D lastlink ~ }
2487     { \exp_not:N \tex_pdflastlink:D }
2488   \c_space_tl 0 ~ R
2489 }
```

(End definition for `_pdf_backend_link_last:`.)

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

```
2490 \cs_new_protected:Npx \_pdf_backend_link_margin:n #1
2491 {
2492   \cs_if_exist:NTF \tex_pdfvariable:D
2493     { \exp_not:N \tex_pdfvariable:D linkmargin }
2494     { \exp_not:N \tex_pdflinkmargin:D }
2495     \exp_not:N \dim_eval:n {#1} \scan_stop:
2496 }
```

(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
`_pdf_backend_destination_rectangle:nn` 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.

```
2497 \cs_new_protected:Npx \_pdf_backend_destination:nn #1#2
2498 {
2499   \cs_if_exist:NTF \tex_pdfextension:D
2500     { \exp_not:N \tex_pdfextension:D dest ~ }
2501     { \exp_not:N \tex_pdfdest:D }
2502     name {#1}
2503     \exp_not:N \str_case:nnF {#2}
2504     {
2505       { xyz } { xyz }
2506       { fit } { fit }
2507       { fitb } { fitb }
2508       { fitbh } { fitbh }
2509       { fitbv } { fitbv }
2510       { fith } { fith }
2511       { fitv } { fitv }
2512     }
2513     { xyz ~ zoom \exp_not:N \fp_eval:n { #2 * 10 } }
2514     \scan_stop:
2515 }
2516 \cs_new_protected:Npx \_pdf_backend_destination_rectangle:nn #1#2
2517 {
2518   \group_begin:
2519     \hbox_set:Nn \l__pdf_internal_box {#2}
2520     \cs_if_exist:NTF \tex_pdfextension:D
2521       { \exp_not:N \tex_pdfextension:D dest ~ }
2522       { \exp_not:N \tex_pdfdest:D }
2523     name {#1}
2524     fitr ~
```

```

2525         width \exp_not:N \box_wd:N \l__pdf_internal_box
2526         height \exp_not:N \box_ht:N \l__pdf_internal_box
2527         depth \exp_not:N \box_dp:N \l__pdf_internal_box
2528         \box_use:N \l__pdf_internal_box
2529     \group_end:
2530 }

```

(End definition for `__pdf_backend_destination:nn` and `__pdf_backend_destination_rectangle:nn`.)

6.3.2 Catalogue entries

```

\__pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
2531 \cs_new_protected:Npx \__pdf_backend_catalog_gput:nn #1#2
2532 {
2533     \cs_if_exist:NTF \tex_pdfextension:D
2534     { \tex_pdfextension:D catalog }
2535     { \tex_pdfcatalog:D }
2536     { / #1 ~ #2 }
2537 }
2538 \cs_new_protected:Npx \__pdf_backend_info_gput:nn #1#2
2539 {
2540     \cs_if_exist:NTF \tex_pdfextension:D
2541     { \tex_pdfextension:D info }
2542     { \tex_pdfinfo:D }
2543     { / #1 ~ #2 }
2544 }

```

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

6.3.3 Objects

```

\g__pdf_backend_object_prop
2545 \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
2546 \group_begin:
2547 \cs_set_protected:Npn \__pdf_tmp:w #1#2
2548 {
2549     \cs_new_protected:Npx \__pdf_backend_object_new:nn ##1##2
2550     {
2551         #1 reserveobjnum ~
2552         \int_const:cn
2553         { c__pdf_backend_object_ \exp_not:N \tl_to_str:n {##1} _int }
2554         {#2}
2555         \prop_gput:Nnn \exp_not:N \g__pdf_backend_object_prop {##1} {##2}
2556     }
2557 }
2558 \cs_if_exist:NTF \tex_pdfextension:D
2559 {
2560     \__pdf_tmp:w
2561     { \tex_pdfextension:D obj ~ }
2562     { \exp_not:N \tex_pdffeedback:D lastobj }

```



```

2563     }
2564     { \_pdf_tmp:w { \tex_pdfobj:D } { \tex_pdflastobj:D } }
2565 \group_end:
2566 \cs_new:Npn \_pdf_backend_object_ref:n #1
2567   { \int_use:c { c\_pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

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

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

```

\_pdf_backend_object_write:nx
\_pdf_exp_not_i:nn
\_pdf_exp_not_ii:nn
2568 \group_begin:
2569   \cs_set_protected:Npn \_pdf_tmp:w #1
2570     {
2571       \cs_new_protected:Npn \_pdf_backend_object_write:nn ##1##2
2572         {
2573           \tex_immediate:D #1 useobjnum ~
2574           \int_use:c
2575             { c\_pdf_backend_object_ \tl_to_str:n {##1} _int }
2576           \str_case_e:nn
2577             { \prop_item:Nn \g\_pdf_backend_object_prop {##1} }
2578             {
2579               { array } { { [ ~ \exp_not:n {##2} ~ ] } }
2580               { dict } { { << ~ \exp_not:n {##2} ~ >> } }
2581               { fstream }
2582                 {
2583                   stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2584                   file ~ { \_pdf_exp_not_ii:nn ##2 }
2585                 }
2586               { stream }
2587                 {
2588                   stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2589                   { \_pdf_exp_not_ii:nn ##2 }
2590                 }
2591             }
2592         }
2593     }
2594   \cs_if_exist:NTF \tex_pdfextension:D
2595     { \_pdf_tmp:w { \tex_pdfextension:D obj ~ } }
2596     { \_pdf_tmp:w { \tex_pdfobj:D } }
2597 \group_end:
2598 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
2599 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2600 \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
2601 \group_begin:
2602   \cs_set_protected:Npn \_pdf_tmp:w #1
2603     {
2604       \cs_new_protected:Npn \_pdf_backend_object_now:nn ##1##2
2605         {
2606           \tex_immediate:D #1
2607           \str_case:nn
2608             {##1}

```

```

2609         {
2610         { array } { { [ ~ \exp_not:n {##2} ~ ] } }
2611         { dict } { { << ~ \exp_not:n {##2} ~ >> } }
2612         { fstream }
2613         {
2614             stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2615             file ~ { \_pdf_exp_not_ii:nn ##2 }
2616         }
2617         { stream }
2618         {
2619             stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2620             { \_pdf_exp_not_ii:nn ##2 }
2621         }
2622     }
2623 }
2624 }
2625 \cs_if_exist:NTF \tex_pdfextension:D
2626 { \_pdf_tmp:w { \tex_pdfextension:D obj ~ } }
2627 { \_pdf_tmp:w { \tex_pdfobj:D } }
2628 \group_end:
2629 \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.

```

2630 \cs_new:Npx \_pdf_backend_object_last:
2631 {
2632     \exp_not:N \int_value:w
2633     \cs_if_exist:NTF \tex_pdffeedback:D
2634     { \exp_not:N \tex_pdffeedback:D lastobj ~ }
2635     { \exp_not:N \tex_pdflastobj:D }
2636     \c_space_tl 0 ~ R
2637 }

```

(End definition for _pdf_backend_object_last:.)

6.3.4 Structure

_pdf_backend_compresslevel:n Simply pass data to the engine.

```

\_pdf_backend_compresslevel:n 2638 \cs_new_protected:Npx \_pdf_backend_compresslevel:n #1
\_pdf_backend_compress_objects:n 2639 {
\_pdf_backend_objcompresslevel:n 2640     \exp_not:N \tex_global:D
2641     \cs_if_exist:NTF \tex_pdfcompresslevel:D
2642     { \tex_pdfcompresslevel:D }
2643     { \tex_pdfvariable:D compresslevel }
2644     \exp_not:N \int_value:w \exp_not:N \int_eval:n {#1} \scan_stop:
2645 }
2646 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2647 {
2648     \bool_if:nTF {#1}
2649     { \_pdf_backend_objcompresslevel:n { 2 } }
2650     { \_pdf_backend_objcompresslevel:n { 0 } }
2651 }
2652 \cs_new_protected:Npx \_pdf_backend_objcompresslevel:n #1

```

```

2653 {
2654   \exp_not:N \tex_global:D
2655   \cs_if_exist:NTF \tex_pdfobjcompresslevel:D
2656     { \tex_pdfobjcompresslevel:D }
2657     { \tex_pdfvariable:D objcompresslevel }
2658     #1 \scan_stop:
2659 }

```

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

`_pdf_backend_version_major_gset:n` At present, we don't have a primitive for the major version in pdfTeX, but we anticipate one ...

`_pdf_backend_version_minor_gset:n`

```

2660 \cs_new_protected:Npx \_pdf_backend_version_major_gset:n #1
2661 {
2662   \cs_if_exist:NTF \tex_pdfvariable:D
2663     {
2664       \int_compare:nNnT \tex_luatexversion:D > { 106 }
2665       {
2666         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2667         \exp_not:N \int_eval:n {#1} \scan_stop:
2668       }
2669     }
2670     {
2671       \cs_if_exist:NT \tex_pdfmajorversion:D
2672       {
2673         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2674         \exp_not:N \int_eval:n {#1} \scan_stop:
2675       }
2676     }
2677 }
2678 \cs_new_protected:Npx \_pdf_backend_version_minor_gset:n #1
2679 {
2680   \exp_not:N \tex_global:D
2681   \cs_if_exist:NTF \tex_pdfminorversion:D
2682     { \exp_not:N \tex_pdfminorversion:D }
2683     { \tex_pdfvariable:D minorversion }
2684     \exp_not:N \int_eval:n {#1} \scan_stop:
2685 }

```

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

`_pdf_backend_version_major:` At present, we don't have a primitive for the major version!

`_pdf_backend_version_minor:`

```

2686 \cs_new:Npx \_pdf_backend_version_major:
2687 {
2688   \cs_if_exist:NTF \tex_pdfvariable:D
2689     {
2690       \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2691       { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2692       { 1 }
2693     }
2694     {
2695       \cs_if_exist:NTF \tex_pdfmajorversion:D
2696       { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2697       { 1 }

```

```

2698     }
2699   }
2700 \cs_new:Npx \__pdf_backend_version_minor:
2701   {
2702   \exp_not:N \tex_the:D
2703   \cs_if_exist:NTF \tex_pdfminorversion:D
2704     { \exp_not:N \tex_pdfminorversion:D }
2705     { \tex_pdfvariable:D minorversion }
2706   }

```

(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:

```

2707 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2708   { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2709 \cs_new_protected:Npn \__pdf_backend_emc:
2710   { \__kernel_backend_literal_page:n { EMC } }

```

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

2711 </pdfmode>

6.4 dvipdfmx backend

2712 <*dvipdfmx | xdvipdfmx>

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

```

\__pdf_backend:x
2713 \cs_new_protected:Npx \__pdf_backend:n #1
2714   { \__kernel_backend_literal:n { pdf: #1 } }
2715 \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

```

2716 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2717   { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2718 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2719   { \__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

```

2720 \int_new:N \g__pdf_backend_object_int
2721 \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 2722 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2
                          2723 {
                          2724   \int_gincr:N \g__pdf_backend_object_int
                          2725   \int_const:cn
                          2726   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
                          2727   { \g__pdf_backend_object_int }
                          2728   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
                          2729 }
2730 \cs_new:Npn \_pdf_backend_object_ref:n #1
2731 { @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 2732 \cs_new_protected:Npn \_pdf_backend_object_write:nn #1#2
\_pdf_backend_object_write:nnn 2733 {
\_pdf_backend_object_write_array:nn 2734   \exp_args:Nx \_pdf_backend_object_write:nnn
\_pdf_backend_object_write_dict:nn 2735   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
\_pdf_backend_object_write_fstream:nn 2736 }
\_pdf_backend_object_write_stream:nn 2737 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
\_pdf_backend_object_write_stream:nnn 2738 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
2739 {
2740   \use:c { __pdf_backend_object_write_ #1 :nn }
2741   { __pdf_backend_object_ref:n {#2} } {#3}
2742 }
2743 \cs_new_protected:Npn \_pdf_backend_object_write_array:nn #1#2
2744 {
2745   \__pdf_backend:x
2746   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2747 }
2748 \cs_new_protected:Npn \_pdf_backend_object_write_dict:nn #1#2
2749 {
2750   \__pdf_backend:x
2751   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2752 }
2753 \cs_new_protected:Npn \_pdf_backend_object_write_fstream:nn #1#2
2754 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2755 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nn #1#2
2756 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2757 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nnnn #1#2#3#4
2758 {
2759   \__pdf_backend:x
2760   {
2761     #1 stream ~ #2 ~
2762     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2763   }
2764 }

```

(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 2765 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2766 {

```

```

2767 \int_gincr:N \g__pdf_backend_object_int
2768 \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2769 { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2770 {#2}
2771 }
2772 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2773 \cs_new:Npn \__pdf_backend_object_last:
2774 { @pdf.obj \int_use:N \g__pdf_backend_object_int }

```

(End definition for __pdf_backend_object_last:.)

6.4.3 Annotations

\g__pdf_landscape_bool There is a bug in (x)dvipdfmx which means annotations do not rotate. As such, we need to know if landscape is active.

```

2775 \bool_new:N \g__pdf_landscape_bool
2776 <*package>
2777 \AtBeginDocument
2778 {
2779   \cs_if_exist:NT \landscape
2780   {
2781     \tl_put_right:Nn \landscape
2782     { \bool_gset_true:N \g__pdf_landscape_bool }
2783     \tl_put_left:Nn \endlandscape
2784     { \bool_gset_false:N \g__pdf_landscape_bool }
2785   }
2786 }
2787 </package>

```

(End definition for \g__pdf_landscape_bool.)

\g__pdf_backend_annotation_int Needed as objects which are not annotations could be created.

```

2788 \int_new:N \g__pdf_backend_annotation_int

```

(End definition for \g__pdf_backend_annotation_int.)

__pdf_backend_annotation:nnnn __pdf_backend_annotation_aux:nnnn Simply pass the raw data through, just dealing with evaluation of dimensions. The only wrinkle is landscape: we have to adjust by hand.

```

2789 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2790 {
2791   \bool_if:NTF \g__pdf_landscape_bool
2792   {
2793     \box_move_up:nn {#2}
2794     {
2795       \vbox:n
2796       {
2797         \__pdf_backend_annotation_aux:nnnn
2798         { #2 + #3 } {#1} { opt } {#4}
2799       }
2800     }
2801   }

```

```

2802     { \_pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4} }
2803   }
2804 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2805   {
2806     \int_gincr:N \g__pdf_backend_object_int
2807     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2808     \_pdf_backend:x
2809     {
2810       ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
2811       width ~ \dim_eval:n {#1} ~
2812       height ~ \dim_eval:n {#2} ~
2813       depth ~ \dim_eval:n {#3} ~
2814       <</Type/Annot #4 >>
2815     }
2816   }

```

(End definition for _pdf_backend_annotation:nnnn and _pdf_backend_annotation_aux:nnnn.)

_pdf_backend_annotation_last:

```

2817 \cs_new:Npn \_pdf_backend_annotation_last:
2818 { @pdf.obj \int_use:N \g__pdf_backend_annotation_int }

```

(End definition for _pdf_backend_annotation_last:.)

_pdf_backend_link_begin_goto:nmw

All created using the same internals.

_pdf_backend_link_begin_user:nmw

```

2819 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nmw #1#2

```

_pdf_backend_link_begin:n

```

2820 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }

```

_pdf_backend_link_end:

```

2821 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nmw #1#2

```

```

2822 { \_pdf_backend_link_begin:n {#1#2} }

```

```

2823 \cs_new_protected:Npn \_pdf_backend_link_begin:n #1

```

```

2824 {

```

```

2825   \_pdf_backend:n

```

```

2826   {

```

```

2827     bann

```

```

2828     <<

```

```

2829       /Type /Annot

```

```

2830       #1

```

```

2831       >>

```

```

2832     }

```

```

2833   }

```

```

2834 \cs_new_protected:Npn \_pdf_backend_link_end:

```

```

2835 { \_pdf_backend:n { eann } }

```

(End definition for _pdf_backend_link_begin_goto:nmw and others.)

_pdf_backend_link_last:

Data not available.

```

2836 \cs_new:Npn \_pdf_backend_link_last: { }

```

(End definition for _pdf_backend_link_last:.)

_pdf_backend_link_margin:n

Pass to dvipdfmx.

```

2837 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1

```

```

2838 { \_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
`_pdf_backend_destination_rectangle:nn` Grahn: the idea is to avoid needing to do any calculations in \TeX by using the backend
data for `@xpos` and `@ypos`.

```

2839 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2840 {
2841   \_pdf_backend:x
2842   {
2843     dest ~ ( \exp_not:n {#1} )
2844     [
2845       @thispage
2846       \str_case:nnF {#2}
2847       {
2848         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2849         { fit } { /Fit }
2850         { fitb } { /FitB }
2851         { fitbh } { /FitBH }
2852         { fitbv } { /FitBV ~ @xpos }
2853         { fith } { /FitH ~ @ypos }
2854         { fitv } { /FitV ~ @xpos }
2855       }
2856       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2857     ]
2858   }
2859 }
2860 \cs_new_protected:Npn \_pdf_backend_destination_rectangle:nn #1#2
2861 {
2862   \group_begin:
2863   \hbox_set:Nn \l__pdf_internal_box {#2}
2864   \box_move_down:nn { \box_dp:N \l__pdf_internal_box }
2865   {
2866     \hbox:n
2867     {
2868       \_pdf_backend:n { obj ~ @pdf_ #1 _llx ~ @xpos }
2869       \_pdf_backend:n { obj ~ @pdf_ #1 _lly ~ @ypos }
2870     }
2871   }
2872   \box_use:N \l__pdf_internal_box
2873   \box_move_up:nn { \box_ht:N \l__pdf_internal_box }
2874   {
2875     \hbox:n
2876     {
2877       \_pdf_backend:n
2878       {
2879         dest ~ (#1)
2880         [
2881           @thispage
2882           /FitR ~
2883           @pdf_ #1 _llx ~ @pdf_ #1 _lly ~
2884           @xpos ~ @ypos
2885         ]
2886       }
2887     }
2888   }
2889   \group_end:

```



```

2890 }
(End definition for \_pdf_backend_destination:nn and \_pdf_backend_destination_rectangle:nn.)

```

6.4.4 Structure

```

\_pdf_backend_compresslevel:n Pass data to the backend: these are a one-shot.
\_pdf_backend_compress_objects:n
2891 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2892 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2893 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2894 {
2895   \bool_if:nF {#1}
2896   { \__kernel_backend_literal:n { dvipdfmx:config-C-0x40 } }
2897 }

```

(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
2898 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2899 {
2900   \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2901   \__kernel_backend_literal:x { pdf:majorversion~ \_pdf_backend_version_major: }
2902 }
2903 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2904 {
2905   \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2906   \__kernel_backend_literal:x { pdf:minorversion~ \_pdf_backend_version_minor: }
2907 }

```

(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:
2908 \cs_new:Npn \_pdf_backend_version_major: { 1 }
2909 \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:
2910 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2911 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2912 \cs_new_protected:Npn \_pdf_backend_emc:
2913 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2914 </dvipdfmx | xdvipdfmx>

```

6.5 dvisvgm backend

2915 \langle *dvisvgm \rangle

6.5.1 Catalogue entries

`_pdf_backend_catalog_gput:nn` No-op.
`_pdf_backend_info_gput:nn` 2916 \backslash cs_new_protected:Npn $_pdf_backend_catalog_gput:nn$ #1#2 { }
2917 \backslash cs_new_protected:Npn $_pdf_backend_info_gput:nn$ #1#2 { }
(End definition for $_pdf_backend_catalog_gput:nn$ and $_pdf_backend_info_gput:nn$.)

6.5.2 Objects

All no-ops here.
`_pdf_backend_object_new:nn` 2918 \backslash cs_new_protected:Npn $_pdf_backend_object_new:nn$ #1#2 { }
`_pdf_backend_object_ref:n` 2919 \backslash cs_new:Npn $_pdf_backend_object_ref:n$ #1 { }
`_pdf_backend_object_write:nn` 2920 \backslash cs_new_protected:Npn $_pdf_backend_object_write:nn$ #1#2 { }
`_pdf_backend_object_write:nx` 2921 \backslash cs_new_protected:Npn $_pdf_backend_object_write:nx$ #1#2 { }
`_pdf_backend_object_now:nn` 2922 \backslash cs_new_protected:Npn $_pdf_backend_object_now:nn$ #1#2 { }
`_pdf_backend_object_now:nx` 2923 \backslash cs_new_protected:Npn $_pdf_backend_object_now:nx$ #1#2 { }
`_pdf_backend_object_last:` 2924 \backslash cs_new:Npn $_pdf_backend_object_last:$ { }
(End definition for $_pdf_backend_object_new:nn$ and others.)

6.5.3 Structure

`_pdf_backend_compresslevel:n` These are all no-ops.
`_pdf_backend_compress_objects:n` 2925 \backslash cs_new_protected:Npn $_pdf_backend_compresslevel:n$ #1 { }
2926 \backslash 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` Data not available!
`_pdf_backend_version_minor_gset:n` 2927 \backslash cs_new_protected:Npn $_pdf_backend_version_major_gset:n$ #1 { }
2928 \backslash cs_new_protected:Npn $_pdf_backend_version_minor_gset:n$ #1 { }
(End definition for $_pdf_backend_version_major_gset:n$ and $_pdf_backend_version_minor_gset:n$.)
`_pdf_backend_version_major:` Data not available!
`_pdf_backend_version_minor:` 2929 \backslash cs_new:Npn $_pdf_backend_version_major:$ { -1 }
2930 \backslash cs_new:Npn $_pdf_backend_version_minor:$ { -1 }
(End definition for $_pdf_backend_version_major:$ and $_pdf_backend_version_minor:.$)
`_pdf_backend_bdc:nn` More no-ops.
`_pdf_backend_emc:` 2931 \backslash cs_new_protected:Npn $_pdf_backend_bdc:nn$ #1#2 { }
2932 \backslash cs_new_protected:Npn $_pdf_backend_emc:$ { }
(End definition for $_pdf_backend_bdc:nn$ and $_pdf_backend_emc:.$)
2933 \langle /dvisvgm \rangle
2934 \langle /initex | package \rangle

Index

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

- A**
- `\AtBeginDocument` 377, 434, 1319, 1454, 1618, 2777
 - `\AtBeginDvi` 42, 43
- B**
- `\begin` 1368, 1373
 - bool commands:
 - `\bool_gset_false:N` 580, 596, 622, 644, 660, 812, 1138, 1174, 2224, 2275, 2784
 - `\bool_gset_true:N` 578, 647, 810, 1153, 2217, 2223, 2782
 - `\bool_if:NTF` 587, 591, 609, 613, 617, 630, 635, 639, 651, 655, 823, 828, 833, 1112, 1157, 1344, 1385, 1501, 1543, 2212, 2227, 2232, 2237, 2791
 - `\bool_if:nTF` 2648, 2895
 - `\bool_lazy_or:nnTF` 1377, 1536
 - `\bool_new:N` 581, 648, 813, 1154, 2200, 2201, 2775
 - `\bool_set_false:N` 1354, 1468, 1561, 1631
 - box commands:
 - `\box_dp:N` 140, 142, 190, 192, 247, 249, 296, 298, 300, 302, 2254, 2287, 2288, 2313, 2415, 2527, 2864
 - `\box_ht:N` 142, 192, 249, 300, 302, 1397, 1598, 2259, 2298, 2299, 2320, 2419, 2526, 2873
 - `\box_if_empty:N` 2354
 - `\box_move_down:nn` 2175, 2254, 2414, 2864
 - `\box_move_up:nn` 2178, 2259, 2418, 2793, 2873
 - `\box_new:N` 1666, 2156, 2157
 - `\box_set_dp:Nn` 1305
 - `\box_set_ht:Nn` 1304
 - `\box_set_wd:Nn` 204, 1303
 - `\box_use:N` . 147, 165, 179, 195, 222, 236, 252, 268, 280, 331, 348, 367, 763, 1020, 1306, 2205, 2417, 2528, 2872
 - `\box_wd:N` . . 141, 149, 191, 197, 248, 254, 297, 299, 335, 1396, 1597, 2525
 - box internal commands:
 - `__box_backend_clip:N` 129, 184, 241, 285
 - `\l__box_backend_cos_fp` 199
 - `__box_backend_rotate:Nn` 151, 199, 256, 338
 - `__box_backend_rotate_aux:Nn` . . . 151, 199, 256
 - `__box_backend_scale:Nnn` 168, 227, 271, 351
 - `\l__box_backend_sin_fp` 199
 - `\g__box_clip_path_int` 285
- C**
- clist commands:
 - `\clist_map_function:nN` 668, 843
 - `\clist_map_function:nn` 1181
 - color internal commands:
 - `__color_backend_cmyk:nnnn` . 401, 470
 - `__color_backend_cmyk_aux:nnnn` . 470
 - `__color_backend_gray:n` 401, 470
 - `__color_backend_gray_aux:n` . . . 470
 - `__color_backend_pickup:N` . . 375, 432
 - `__color_backend_pickup:w` 13, 375, 432
 - `__color_backend_reset:` 401, 470
 - `__color_backend_rgb:nnn` . . . 401, 470
 - `__color_backend_rgb_aux:nnn` . . 470
 - `__color_backend_select:n` . . 401, 470
 - `__color_backend_spot:nn` . . . 401, 470
 - `color.fc` 401, 527
 - cs commands:
 - `\cs_generate_variant:Nn` 28, 32, 35, 72, 100, 105, 116, 123, 427, 512, 526, 731, 737, 773, 921, 1029, 1060, 1515, 1572, 1588, 1670, 1707, 1752, 2598, 2629, 2715, 2737, 2772
 - `\cs_gset:Npx` 2900, 2905
 - `\cs_if_exist:N` . . . 42, 67, 75, 83, 89, 95, 379, 436, 506, 515, 904, 912, 2350, 2448, 2459, 2470, 2478, 2485, 2492, 2499, 2520, 2533, 2540, 2558, 2594, 2625, 2633, 2641, 2655, 2662, 2671, 2681, 2688, 2695, 2703, 2779
 - `\cs_new:Npn` 673, 848, 1185, 1601, 1610, 1660, 1685, 1753, 2195, 2376, 2438, 2439, 2566, 2599, 2600, 2730, 2773, 2817, 2836, 2908, 2909, 2919, 2924, 2929, 2930
 - `\cs_new:Npx` 2456, 2482, 2630, 2686, 2700
 - `\cs_new_eq:NN` 25, 525, 772, 778, 779, 919, 1028,

1321, 1350, 1407, 1408, 1456, 1464,
1486, 1557, 1620, 1627, 1659, 2205

`\cs_new_protected:Npn`
. . . 26, 30, 33, 48, 54, 59, 61, 103,
106, 108, 110, 114, 117, 119, 121,
129, 151, 153, 168, 184, 199, 201,
227, 241, 256, 258, 271, 285, 338,
351, 376, 396, 401, 410, 412, 417,
419, 428, 433, 443, 470, 481, 486,
488, 490, 500, 502, 527, 533, 538,
540, 542, 550, 558, 567, 577, 579,
582, 584, 598, 603, 624, 646, 649,
662, 675, 680, 682, 684, 686, 688,
690, 692, 694, 703, 712, 714, 716,
721, 726, 732, 738, 750, 774, 776,
780, 785, 790, 800, 809, 811, 814,
816, 818, 820, 825, 830, 835, 837,
850, 855, 857, 859, 861, 863, 865,
867, 869, 878, 887, 889, 891, 896,
922, 937, 962, 974, 986, 998, 1005,
1030, 1035, 1037, 1045, 1055, 1063,
1068, 1073, 1084, 1094, 1104, 1106,
1108, 1110, 1141, 1143, 1148, 1150,
1152, 1155, 1176, 1187, 1200, 1202,
1204, 1206, 1208, 1210, 1212, 1214,
1216, 1226, 1235, 1243, 1245, 1247,
1257, 1272, 1277, 1292, 1322, 1336,
1351, 1363, 1374, 1402, 1414, 1427,
1437, 1458, 1465, 1473, 1484, 1488,
1491, 1506, 1516, 1551, 1558, 1564,
1570, 1573, 1580, 1589, 1594, 1602,
1621, 1628, 1634, 1636, 1638, 1649,
1668, 1671, 1673, 1677, 1687, 1708,
1713, 1718, 1723, 1732, 2159, 2173,
2204, 2206, 2208, 2210, 2215, 2230,
2235, 2277, 2306, 2330, 2339, 2378,
2385, 2410, 2434, 2435, 2436, 2437,
2440, 2442, 2464, 2466, 2571, 2604,
2646, 2707, 2709, 2716, 2718, 2722,
2732, 2738, 2743, 2748, 2753, 2755,
2757, 2765, 2789, 2804, 2819, 2821,
2823, 2834, 2837, 2839, 2860, 2891,
2893, 2898, 2903, 2910, 2912, 2916,
2917, 2918, 2920, 2921, 2922, 2923,
2925, 2926, 2927, 2928, 2931, 2932

`\cs_new_protected:Npx`
. 36, 65, 73, 81, 87,
93, 504, 513, 902, 910, 2446, 2468,
2476, 2490, 2497, 2516, 2531, 2538,
2549, 2638, 2652, 2660, 2678, 2713

`\cs_set_eq:NN` 2371, 2372

`\cs_set_protected:Npn`
. 381, 438, 2547, 2569, 2602

D

dim commands:

`\dim_eval:n` 2182, 2451, 2452,
2453, 2495, 2811, 2812, 2813, 2838

`\dim_max:nn` 2285, 2296

`\dim_set:Nn` 1396, 1397, 1597, 1598

`\dim_to_decimal:n` 296, 297, 298, 299,
300, 302, 1066, 1071, 1077, 1078,
1079, 1080, 1089, 1090, 1091, 1182,
1201, 1654, 1655, 2283, 2294, 2312,
2313, 2315, 2318, 2320, 2324, 2382

`\dim_to_decimal_in_bp:n` . 140, 141,
142, 190, 191, 192, 247, 248, 249,
546, 547, 554, 555, 562, 563, 571,
572, 573, 670, 674, 678, 783, 788,
794, 795, 796, 804, 805, 845, 849,
853, 1186, 1327, 1328, 1329, 1330,
1478, 1479, 1480, 1481, 1530, 1531,
1532, 1533, 1643, 1644, 1645, 1646

draw internal commands:

`__draw_align_currentpoint` 21

`__draw_backend_add_to_path:n`
. 1063, 1109

`__draw_backend_begin:` 527, 774, 1030

`__draw_backend_box_use:Nnnnn`
. 16, 750, 1005, 1292

`__draw_backend_cap_but:`
. 662, 837, 1176

`__draw_backend_cap_rectangle:`
. 662, 837, 1176

`__draw_backend_cap_round:`
. 662, 837, 1176

`__draw_backend_clip:` 582, 814, 1108

`__draw_backend_closepath:`
. 582, 814, 1108

`__draw_backend_closestroke:`
. 582, 814, 1108

`__draw_backend_cm:nnnn` 738,
758, 759, 760, 922, 1009, 1277, 1295

`__draw_backend_cm_aux:nnnn` 922

`__draw_backend_cm_decompose:nnnnN`
. 932, 961

`__draw_backend_cm_decompose_-`
auxi:nnnnN 961

`__draw_backend_cm_decompose_-`
auxii:nnnnN 961

`__draw_backend_cm_decompose_-`
auxiii:nnnnN 961

`__draw_backend_color_fill:n` 694

`__draw_backend_color_fill:nnn` 1216

`__draw_backend_color_fill_-`
cmyk:nnnn 694, 869, 1216

`__draw_backend_color_fill_-`
gray:n 694, 869, 1216

<code>__draw_backend_color_fill_-</code>	<code>__draw_backend_nonzero_rule: ...</code>
<code>rgb:nnn</code> 694 , 869 , 1216 577 , 809 , 1104
<code>__draw_backend_color_gray_aux:n</code>	<code>__draw_backend_path:n</code> 1108
..... 1239 , 1243	<code>__draw_backend_rectangle:n</code> ..
<code>__draw_backend_color_reset: ..</code> 869 542 , 780 , 1063
<code>__draw_backend_color_select:n</code> .. 869	<code>__draw_backend_scope:n</code>
<code>__draw_backend_color_stroke:n</code> .. 694	... 1033 , 1037 , 1105 , 1107 , 1127 ,
<code>__draw_backend_color_stroke_-</code>	1167 , 1189 , 1201 , 1203 , 1205 , 1207 ,
<code>cmyk:n</code> 694 , 869 , 1216	1209 , 1211 , 1213 , 1215 , 1259 , 1279
<code>__draw_backend_color_stroke_-</code>	<code>__draw_backend_scope_begin: ...</code>
<code>gray:n</code> 694 , 869 , 1216 538 , 775 , 778 , 1032 , 1037
<code>__draw_backend_color_stroke_-</code>	<code>__draw_backend_scope_end:</code>
<code>rgb:nnn</code> 694 , 869 , 1216 538 , 777 , 778 , 1036 , 1037
<code>__draw_backend_curveto:n</code> 542 , 780 , 1063	<code>__draw_backend_select:n</code>
<code>__draw_backend_dash:n</code> 662 , 837 , 1176 1228 , 1246 , 1274
<code>__draw_backend_dash_aux:nn</code> .. 1176	<code>__draw_backend_stroke: 582, 814, 1108</code>
<code>__draw_backend_dash_pattern:nn</code> ..	<code>\g__draw_clip_path_int</code>
..... 662 , 837 , 1176	.. 1114 , 1117 , 1130 , 1159 , 1162 , 1170
<code>__draw_backend_discardpath: ...</code>	<code>__draw_color_reset:</code> 735
..... 582 , 814 , 1108	<code>\g__draw_draw_clip_bool</code> ... 582 , 1108
<code>__draw_backend_end: ..</code> 527 , 774 , 1030	<code>\g__draw_draw_eor_bool</code>
<code>__draw_backend_evenodd_rule: ...</code> 577 , 591 , 609 ,
..... 577 , 809 , 1104	617 , 630 , 639 , 655 , 809 , 823 , 828 , 833
<code>__draw_backend_fill: 582, 814, 1108</code>	<code>\g__draw_draw_path_int</code>
<code>__draw_backend_fillstroke:</code> 1108
..... 582 , 814 , 1108	<code>\g__draw_draw_path_tl</code>
<code>__draw_backend_join_bevel:</code>	.. 1063 , 1119 , 1135 , 1137 , 1164 , 1173
..... 662 , 837 , 1176	<code>\g__draw_draw_scope_int</code>
<code>__draw_backend_join_miter:</code> 1037
..... 662 , 837 , 1176	<code>\l__draw_draw_scope_int</code>
<code>__draw_backend_join_round:</code> 1037
..... 662 , 837 , 1176	<code>\g__draw_path_int</code>
<code>__draw_backend_lineto:nn</code> 1123 , 1140
..... 542 , 780 , 1063	
<code>__draw_backend_linewidth:n</code>	
..... 662 , 837 , 1176	
<code>__draw_backend_literal:n</code> 525 , 530 ,	
531 , 535 , 539 , 541 , 544 , 552 , 560 ,	
569 , 583 , 586 , 589 , 595 , 605 , 606 ,	
607 , 612 , 615 , 621 , 626 , 627 , 628 ,	
633 , 634 , 637 , 643 , 653 , 659 , 664 ,	
677 , 681 , 683 , 685 , 687 , 689 , 691 ,	
693 , 740 , 752 , 753 , 754 , 755 , 756 ,	
757 , 761 , 762 , 764 , 765 , 766 , 767 ,	
768 , 772 , 782 , 787 , 792 , 802 , 815 ,	
817 , 819 , 822 , 827 , 832 , 836 , 839 ,	
852 , 856 , 858 , 860 , 862 , 864 , 866 ,	
868 , 1028 , 1049 , 1057 , 1115 , 1134 , 1160	
<code>__draw_backend_miterlimit:n</code> ...	
..... 662 , 837 , 1176	
<code>__draw_backend_moveto:nn</code>	
..... 542 , 780 , 1063	
	E
	<code>\endlandscape</code>
 2783
	<code>\evensidemargin</code>
 2251
	exp commands:
	<code>\exp_after:wN</code>
 388 , 1608
	<code>\exp_args:Nf</code>
 667 , 842
	<code>\exp_args:NNf</code>
 152 , 200 , 257
	<code>\exp_args:Nnx</code>
 1749 , 2768
	<code>\exp_args:NV</code>
 383
	<code>\exp_args:Nx</code>
	... 487 , 1420 , 1441 , 1720 , 2243 , 2734
	<code>\exp_last_unbraced:Nx</code>
 392 , 440
	<code>\exp_not:N</code>
 43 ,
	70 , 79 , 98 , 509 , 510 , 518 , 907 , 908 ,
	915 , 2451 , 2452 , 2453 , 2458 , 2460 ,
	2461 , 2484 , 2486 , 2487 , 2493 , 2494 ,
	2495 , 2500 , 2501 , 2503 , 2513 , 2521 ,
	2522 , 2525 , 2526 , 2527 , 2553 , 2555 ,
	2562 , 2632 , 2634 , 2635 , 2640 , 2644 ,
	2654 , 2666 , 2667 , 2673 , 2674 , 2680 ,
	2682 , 2684 , 2691 , 2696 , 2702 , 2704
	<code>\exp_not:n</code> ... 27 , 70 , 79 , 98 , 1711 ,
	1716 , 2406 , 2579 , 2580 , 2599 , 2600 ,
	2610 , 2611 , 2746 , 2751 , 2762 , 2843

F

file commands:
`\file_compare_timestamp:nNnTF` . 1429
`\file_parse_full_name:nNNN` 1416, 1439

fp commands:
`\fp_compare:nNnTF`
 ... 159, 206, 212, 264, 942, 955, 1000
`\fp_eval:n` 152, 161, 174,
 175, 200, 217, 232, 234, 257, 266,
 277, 278, 345, 360, 361, 406, 407,
 411, 415, 475, 476, 477, 478, 487,
 495, 496, 497, 681, 698, 699, 708,
 709, 713, 715, 719, 724, 743, 744,
 856, 873, 874, 882, 883, 888, 890,
 894, 899, 927, 928, 944, 949, 950,
 957, 967, 968, 969, 970, 979, 980,
 981, 982, 991, 992, 993, 994, 1015,
 1016, 1203, 1221, 1222, 1223, 1231,
 1232, 1240, 1246, 1252, 1253, 1254,
 1275, 1285, 1286, 2403, 2513, 2856
`\fp_new:N` 225, 226
`\fp_set:Nn` 205, 208
`\fp_use:N` 211, 215, 220
`\fp_zero:N` 207
`\c_zero_fp` . 159, 206, 212, 264, 942, 955

G

galley commands:
`\l_galley_text_width_dim` 2315
`\l_galley_total_left_margin_dim` 2246

graphics commands:
`\graphics_bb_restore:nTF` . 1365, 1591
`\graphics_bb_save:n` 1400, 1599
`\l_graphics_decodearray_tl`
 1342, 1343,
 1353, 1379, 1383, 1384, 1467, 1499,
 1500, 1538, 1541, 1542, 1560, 1630
`\graphics_extract_bb:n`
 1462, 1469, 1625, 1632
`\l_graphics_interpolate_bool` ...
 1344, 1354, 1378, 1385,
 1468, 1501, 1537, 1543, 1561, 1631
`\l_graphics_llx_dim`
 1327, 1478, 1530, 1643
`\l_graphics_lly_dim`
 1328, 1479, 1531, 1644
`\l_graphics_name_tl` 1434
`\l_graphics_page_int`
 1338, 1358, 1359, 1389,
 1390, 1460, 1497, 1498, 1524, 1525,
 1553, 1566, 1567, 1606, 1607, 1623
`\l_graphics_pagebox_tl`
 41, 1339, 1357,

1391, 1392, 1461, 1495, 1496, 1526,
 1528, 1554, 1575, 1576, 1608, 1624
`\graphics_read_bb:n` . 1321, 1456, 1620
`\l_graphics_urx_dim`
 .. 1329, 1396, 1480, 1532, 1597, 1645
`\l_graphics_ury_dim` .. 1330, 1397,
 1481, 1533, 1598, 1646, 1654, 1655

graphics internal commands:
`\l__graphics_backend_dir_str` . 1409
`\l__graphics_backend_ext_str` . 1409
`__graphics_backend_getbb_auxi:n`
 1336
`__graphics_backend_getbb_-
 auxi:nN` 1551
`__graphics_backend_getbb_-
 auxii:n` 1336
`__graphics_backend_getbb_-
 auxii:nnN` 1551
`__graphics_backend_getbb_-
 auxiii:nNnn` 1551
`__graphics_backend_getbb_-
 auxiv:nnNnn` 1551
`__graphics_backend_getbb_-
 auxv:nNnn` 1551
`__graphics_backend_getbb_-
 auxvi:nNnn` 1592, 1594
`__graphics_backend_getbb_eps:n` .
 1315, 1409, 1450, 1614
`__graphics_backend_getbb_eps:nm`
 1409
`__graphics_backend_getbb_eps:nn`
 1420, 1427
`__graphics_backend_getbb_jpg:n` .
 1336, 1450, 1551, 1621
`__graphics_backend_getbb_-
 pagebox:w` 1551, 1608
`__graphics_backend_getbb_pdf:n` .
 1336, 1435, 1450, 1551, 1628
`__graphics_backend_getbb_png:n` .
 1336, 1450, 1551, 1621
`__graphics_backend_include:nn` 1634
`__graphics_backend_include_-
 auxi:nn` 1473
`__graphics_backend_include_-
 auxii:nnn` 1473
`__graphics_backend_include_-
 auxiii:nnn` 1473
`__graphics_backend_include_-
 bitmap_quote:w` 1602, 1649
`__graphics_backend_include_-
 eps:n` 1322, 1409, 1473, 1634
`__graphics_backend_include_-
 jpg:n` 1402, 1473, 1649

<code>__graphics_backend_include_-pdf:n</code> .. 1402 , 1441 , 1473 , 1602 , 1634	<code>\int_set_eq:NN</code> 1039 , 2344
<code>__graphics_backend_include_pdf_-quote:w</code> 1605 , 1610	<code>\int_use:N</code> 289 , 320 , 1117 , 1123 , 1130 , 1162 , 1170 , 1359 , 1390 , 1405 , 1498 , 1511 , 1523 , 1525 , 1607 , 1686 , 1737 , 1750 , 1754 , 2167 , 2190 , 2196 , 2269 , 2377 , 2567 , 2574 , 2731 , 2769 , 2774 , 2810 , 2818
<code>__graphics_backend_include_-png:n</code> 1402 , 1473 , 1649	<code>\int_value:w</code> .. 2458 , 2484 , 2632 , 2644
<code>\l__graphics_backend_name_str</code> . 1409	<code>\int_zero:N</code> ... 1338 , 1460 , 1553 , 1623
<code>\l__graphics_graphics_attr_tl</code> 1335 , 1340 , 1347 , 1355 , 1365 , 1398 , 1400 , 1405	
<code>\l__graphics_internal_box</code> 1394 , 1396 , 1397 , 1596 , 1597 , 1598	
<code>\g__graphics_track_int</code> 1472 , 1518 , 1519	
group commands:	
<code>\group_begin:</code> 1042 , 2412 , 2518 , 2546 , 2568 , 2601 , 2862	
<code>\group_end:</code> 1050 , 2432 , 2529 , 2565 , 2597 , 2628 , 2889	
<code>\group_insert_after:N</code> 425 , 510 , 735 , 908	
H	
hbox commands:	
<code>\hbox:n</code> 2176 , 2177 , 2180 , 2255 , 2261 , 2416 , 2420 , 2866 , 2875	
<code>\hbox_overlap_right:n</code> 147 , 179 , 195 , 236 , 252 , 280 , 367 , 763 , 1020	
<code>\hbox_set:Nn</code> 1394 , 1596 , 2242 , 2279 , 2413 , 2519 , 2863	
<code>\hbox_set:Nw</code> 2225	
<code>\hbox_set_end:</code> 2240	
<code>\hbox_unpack:N</code> 2372	
I	
int commands:	
<code>\int_compare:nNnTF</code> 1358 , 1389 , 1497 , 1524 , 1566 , 1606 , 2343 , 2664 , 2690	
<code>\int_const:Nn</code> 1398 , 1519 , 1680 , 2552 , 2725	
<code>\int_eval:n</code> 2644 , 2667 , 2674 , 2684 , 2892 , 2900 , 2905	
<code>\int_gincr:N</code> 287 , 1058 , 1114 , 1159 , 1518 , 1679 , 1734 , 2162 , 2186 , 2264 , 2724 , 2767 , 2806	
<code>\int_gset:Nn</code> 2332	
<code>\int_gset_eq:NN</code> 1051 , 2163 , 2187 , 2265 , 2807	
<code>\int_gzero:N</code> 1043	
<code>\int_if_exist:NTF</code> 1508	
<code>\int_if_odd:nTF</code> 2249	
<code>\int_new:N</code> 337 , 469 , 1061 , 1062 , 1140 , 1472 , 1675 , 2158 , 2197 , 2199 , 2720 , 2788	
	K
kernel internal commands:	
<code>__kernel_backend_align_begin:</code> 48 , 132 , 156 , 171	
<code>__kernel_backend_align_end:</code> 48 , 146 , 164 , 178	
<code>__kernel_backend_literal:n</code> 25 , 31 , 34 , 38 , 45 , 50 , 57 , 60 , 62 , 104 , 107 , 109 , 111 , 115 , 261 , 274 , 421 , 429 , 529 , 536 , 734 , 939 , 946 , 952 , 1012 , 1022 , 1324 , 1475 , 1510 , 1520 , 1640 , 1651 , 2714 , 2838 , 2892 , 2896 , 2901 , 2906	
<code>__kernel_backend_literal_page:n</code> 73 , 106 , 2708 , 2710 , 2911 , 2913	
<code>__kernel_backend_literal_pdf:n</code> 65 , 103 , 187 , 244 , 772 , 919	
<code>__kernel_backend_literal_-postscript:n</code> ... 30 , 51 , 52 , 56 , 133 , 134 , 136 , 137 , 145 , 157 , 172 , 525	
<code>__kernel_backend_literal_svg:n</code> 114 , 118 , 120 , 122 , 288 , 290 , 307 , 1028 , 1296 , 1307	
<code>__kernel_backend_matrix:n</code> 93 , 209 , 230 , 925	
<code>__kernel_backend_postscript:n</code> 33 , 423 , 728 , 1669 , 1725 , 2176 , 2183 , 2218 , 2255 , 2262 , 2266 , 2280 , 2308 , 2358 , 2365 , 2371 , 2380 , 2387 , 2416 , 2420	
<code>__kernel_backend_postscript_-header:n</code> 36 , 1755 , 1761 , 1768 , 1774 , 1813 , 1851 , 1992 , 2099	
<code>__kernel_backend_scope_begin:</code> 5 , 59 , 81 , 108 , 117 , 131 , 155 , 170 , 186 , 203 , 229 , 243 , 260 , 273 , 778 , 1007 , 1294	
<code>__kernel_backend_scope_begin:n</code> 121 , 309 , 317 , 322 , 340 , 353	
<code>__kernel_backend_scope_end:</code> 59 , 81 , 108 , 117 , 148 , 166 , 180 , 196 , 223 , 237 , 253 , 269 , 281 , 332 , 333 , 334 , 349 , 368 , 779 , 1024 , 1308	
<code>\l__kernel_color_stack_int</code> 469 , 509 , 518 , 907 , 915	

	L	
\landscape	2779, 2781
	M	
math commands:		
\c_math_toggle_token	2228, 2238
mode commands:		
\mode_if_horizontal:TF	...	2334, 2341
\mode_if_math:TF	2222
	O	
\oddsidemargin	2250
	P	
pdf internal commands:		
_pdf_backend:n	2713, 2717, 2719, 2745, 2750, 2759, 2808, 2825, 2835, 2841, 2868, 2869, 2877
_pdf_backend_annotation:nnnn
_pdf_backend_annotation_-		
aux:nnnn	2159, 2446, 2789
\g_pdf_backend_annotation_int
_pdf_backend_annotation_last:
_pdf_backend_bdc:nn
_pdf_backend_catalog_gput:nn
_pdf_backend_compress_objects:n	2434, 2638, 2891, 2925
_pdf_backend_compresslevel:n
\l_pdf_backend_content_box	2156,	2225, 2254, 2257, 2259, 2288, 2299
_pdf_backend_destination:nn
_pdf_backend_destination_-		
rectangle:nn	2385, 2497, 2839
_pdf_backend_emc:
_pdf_backend_info_gput:nn
_pdf_backend_link:nw	2206
_pdf_backend_link_aux:nw	...	2206
_pdf_backend_link_begin:n	..	2819
_pdf_backend_link_begin:nnw	2464	
_pdf_backend_link_begin:nw
_pdf_backend_link_begin_aux:nw	2213, 2215
_pdf_backend_link_begin_-		
goto:nnw	2206, 2464, 2819
_pdf_backend_link_begin_-		
user:nnw	2206, 2464, 2819
\g_pdf_backend_link_bool
\g_pdf_backend_link_dict_tl
_pdf_backend_link_end:
_pdf_backend_link_end_aux:	..	2206
\g_pdf_backend_link_int
_pdf_backend_link_last:
_pdf_backend_link_margin:n
\g_pdf_backend_link_math_bool
_pdf_backend_link_minima:	..	2206
_pdf_backend_link_outerbox:n	2206	
\g_pdf_backend_link_sf_int
_pdf_backend_link_sf_restore:	2206	
_pdf_backend_link_sf_save:	..	2206
\l_pdf_backend_model_box	..	2157, 2242, 2279, 2287, 2298, 2313, 2320
_pdf_backend_objcompresslevel:n
\g_pdf_backend_object_int
_pdf_backend_object_last:
_pdf_backend_object_new:nn
_pdf_backend_object_now:nn
\g_pdf_backend_object_prop
_pdf_backend_object_ref:n	1677,	1691, 1705, 2546, 2722, 2741, 2918
_pdf_backend_object_write:nn
_pdf_backend_object_write:nnn	2732	
_pdf_backend_object_write_-		
array:nn	1687, 2732
_pdf_backend_object_write_-		
dict:nn	1687, 2732
_pdf_backend_object_write_-		
fstream:nn	2732
_pdf_backend_object_write_-		
stream:nn	1687, 2732

sys commands:		
\sys_if_shell:TF	1409	
\sys_shell_now:n	1431	
T		
TeX and L ^A T _E X 2 _ε commands:		
\ccclv	2354, 2356, 2364	
\@makecol@hook	2347	
\current@color	13, 383, 388, 393, 441	
\special	1	
tex commands:		
\textbaseline:D	2324	
\textglobal:D		
.....	2640, 2654, 2666, 2673, 2680	
\textimmediate:D	1376, 2573, 2606	
\textkern:D	2182	
\textluatexversion:D	2664, 2690	
\textpdfannot:D	2450	
\textpdfcatalog:D	2535	
\textpdfcolorstack:D	508, 517, 906, 914	
\textpdfcompresslevel:D	2641, 2642	
\textpdfdest:D	2501, 2522	
\textpdfendlink:D	2480	
\textpdfextension:D	67, 68, 75, 76,	
83, 84, 89, 90, 95, 96, 506, 507, 515,		
516, 904, 905, 912, 913, 2448, 2449,		
2470, 2471, 2478, 2479, 2499, 2500,		
2520, 2521, 2533, 2534, 2540, 2541,		
2558, 2561, 2594, 2595, 2625, 2626		
\textpdffeedback:D	2459,	
2460, 2485, 2486, 2562, 2633, 2634		
\textpdfinfo:D	2542	
\textpdflastannot:D	2461	
\textpdflastlink:D	2487	
\textpdflastobj:D	2564, 2635	
\textpdflastximage:D	1395, 1399	
\textpdflinkmargin:D	2494	
\textpdfliteral:D	69, 77	
\textpdfmajorversion:D		
.....	2671, 2673, 2695, 2696	
\textpdfminorversion:D		
.....	2681, 2682, 2703, 2704	
\textpdfobj:D	2564, 2596, 2627	
\textpdfobjcompresslevel:D	2655, 2656	
\textpdfrefximage:D	1395, 1404	
\textpdfrestore:D	91	
\textpdfsave:D	85	
\textpdfsetmatrix:D	97	
\textpdfstartlink:D	2472	
\textpdfvariable:D		
.....	2492, 2493, 2643, 2657,	
2662, 2666, 2683, 2688, 2691, 2705		
\textpdfximage:D	1376	
\textpdfximagebbox:D	1370	
\textspacefactor:D	2335, 2344	
\textspecial:D	25	
\textthe:D	1399, 2691, 2696, 2702	
\textXeTeXpdffile:D	1562, 1604	
\textXeTeXpicfile:D	1555	
\textwidth	2318	
tl commands:		
\c_space_tl		
..	211, 216, 219, 388, 1099, 1326,	
1327, 1328, 1329, 1477, 1478, 1479,		
1480, 1525, 1528, 1530, 1531, 1532,		
1533, 1605, 1607, 1642, 1643, 1644,		
1645, 2270, 2462, 2488, 2636, 2810		
\tl_clear:N	1339, 1347, 1353,	
1461, 1467, 1554, 1560, 1624, 1630		
\tl_gclear:N	1137, 1173	
\tl_gset:Nn	1096, 2220	
\tl_if_empty:NTF	1099, 1342, 1383,	
1391, 1495, 1499, 1526, 1541, 1575		
\tl_if_empty:nTF	1193	
\tl_if_empty_p:N	1379, 1538	
\tl_if_head_is_space:nTF	383	
\tl_new:N	1103, 1335, 2198, 2202	
\tl_put_left:Nn	2783	
\tl_put_right:Nn	2352, 2781	
\tl_set:Nn	385, 397, 447, 450, 453,	
457, 460, 1340, 1355, 1434, 2203, 2370		
\tl_to_str:n	1681,	
1686, 2553, 2567, 2575, 2726, 2731		
U		
use commands:		
\use:N	1703, 1749, 2740, 2768	
\use:n	44, 388, 472, 492,	
667, 842, 964, 976, 988, 1178, 1218,		
1237, 1249, 1316, 1451, 1582, 1615		
\use_none:n	457, 1193, 1195, 2348	
V		
\value	2249	
vbox commands:		
\vbox:n	2795	
\vbox_set:Nn	2356	
\vbox_unpack_drop:N	2364	