

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-backed 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 Experimental backend support: dvipdfmx}
7 </dvipdfmx>
8 <*dvips>
9 {l3backend-dvips.def}{2019-04-06}{ }
10 {L3 Experimental backend support: dvips}
11 </dvips>
12 <*dvisvgm>
13 {l3backend-dvisvgm.def}{2019-04-06}{ }
14 {L3 Experimental backend support: dvisvgm}
15 </dvisvgm>
16 <*pdfmode>
17 {l3backend-pdfmode.def}{2019-04-06}{ }
18 {L3 Experimental backend support: PDF mode}
19 </pdfmode>
20 <*xdvipdfmx>
21 {l3backend-xdvipdfmx.def}{2019-04-06}{ }
22 {L3 Experimental 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` and `__kernel_backend_literal:n`.)

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 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 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 build-in saving of the current position, and so some additional PostScript is required to set up the transformation matrix and also to restore it afterwards. Notice the use of the stack to save the current position “up front” and to move back to it at the end of the process. Notice that the `[begin]/[end]` pair here mean that we can use a run of PostScript statements in separate lines: not *required* but does make the code and output more clear.

```
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.

```

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.

```

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 $\langle *dvisvgm \rangle$

$\backslash_kernel_backend_literal_svg:n$
 $\backslash_kernel_backend_literal_svg:x$

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

114 $\backslash cs_new_protected:Npn \backslash_kernel_backend_literal_svg:n \#1$
115 $\{ \backslash_kernel_backend_literal:n \{ dvisvgm:raw~ \#1 \{ ?nl \} \} \}$
116 $\backslash cs_generate_variant:Nn \backslash_kernel_backend_literal_svg:n \{ x \}$

(End definition for $\backslash_kernel_backend_literal_svg:n$.)

$\backslash_kernel_backend_scope_begin:$
 $\backslash_kernel_backend_scope_end:$

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 $\backslash cs_new_protected:Npn \backslash_kernel_backend_scope_begin:$
118 $\{ \backslash_kernel_backend_literal_svg:n \{ <g> \} \}$
119 $\backslash cs_new_protected:Npn \backslash_kernel_backend_scope_end:$
120 $\{ \backslash_kernel_backend_literal_svg:n \{ </g> \} \}$

(End definition for $\backslash_kernel_backend_scope_begin:$ and $\backslash_kernel_backend_scope_end:.$)

$\backslash_kernel_backend_scope_begin:n$
 $\backslash_kernel_backend_scope_begin:x$

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 $\backslash_kernel_backend_scope_begin:$ as a result. No assumptions are made about the nature of the scoped operation(s).

121 $\backslash cs_new_protected:Npn \backslash_kernel_backend_scope_begin:n \#1$
122 $\{ \backslash_kernel_backend_literal_svg:n \{ <g~ \#1 > \} \}$
123 $\backslash cs_generate_variant:Nn \backslash_kernel_backend_scope_begin:n \{ x \}$

(End definition for $\backslash_kernel_backend_scope_begin:n$.)

124 $\langle /dvisvgm \rangle$

125 $\langle /initex | package \rangle$

2 l3backend-box Implementation

126 $\langle *initex | package \rangle$

127 $\langle @@=box \rangle$

2.1 dvips backend

128 $\langle *dvips \rangle$

$\backslash_box_backend_clip:N$

The **dvips** backend scales all absolute dimensions based on the output resolution selected and any T_EX magnification. Thus for any operation involving absolute lengths there is a correction to make. See `normalscale` from `special.pro` for the variables, noting that here everything is saved on the stack rather than as a separate variable. Once all of that is done, the actual clipping is trivial.

129 $\backslash cs_new_protected:Npn \backslash_box_backend_clip:N \#1$
130 $\{$
131 $\backslash_kernel_backend_scope_begin:$
132 $\backslash_kernel_backend_align_begin:$
133 $\backslash_kernel_backend_literal_postscript:n \{ matrix-currentmatrix \}$
134 $\backslash_kernel_backend_literal_postscript:n$

```

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 \TeX 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 \TeX box is inserted to get things back on track. The clip path needs to come between those two such that it lines up with the current point, as does the \TeX box.

```

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 l3backend-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>
```

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   \@ifpackageloaded { color }
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 }
401 </package>
```

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

```
\_color_backend_cmyk:nnnn
\_color_backend_gray:n
\_color_backend_rgb:nnn
\_color_backend_spot:nn
\_color_backend_select:n
\_color_backend_select:x
\_color_backend_reset:
color.fc
```

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

```
402 \cs_new_protected:Npn \_color_backend_cmyk:nnnn #1#2#3#4
```

```

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

```

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

```

431 </dvisvgm | dvipdfmx | dvips | xdvipdfmx>

```

3.2 pdfmode

```

432 <*pdfmode>

```

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

```

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

```

```

446     {
447       \str_if_eq:nnTF {#2} { g }
448       { \tl_set:Nn #7 { gray ~ #1 } }
449       {
450         \str_if_eq:nnTF {#4} { rg }
451         { \tl_set:Nn #7 { rgb ~ #1 ~ #2 ~ #3 } }
452         {
453           \str_if_eq:nnTF {#5} { k }
454           { \tl_set:Nn #7 { cmyk ~ #1 ~ #2 ~ #3 ~ #4 } }
455           {
456             \str_if_eq:nnTF {#2} { cs }
457             {
458               \tl_set:Nx #7 { spot ~ \use_none:n #1 ~ #5 }
459             }
460             {
461               \tl_set:Nn #7 { gray ~ 0 }
462             }
463           }
464         }
465       }
466     }
467   }
468 { }
469 }
470 </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.

```

471 \int_new:N \l_kernel_color_stack_int

```

(End definition for `\l_kernel_color_stack_int`.)

`_color_backend_cmyk:nnnn` 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:
472 \cs_new_protected:Npn \__color_backend_cmyk:nnnn #1#2#3#4
473 {
474   \use:x
475   {
476     \__color_backend_cmyk_aux:nnnn
477     { \fp_eval:n {#1} }
478     { \fp_eval:n {#2} }
479     { \fp_eval:n {#3} }
480     { \fp_eval:n {#4} }
481   }
482 }
483 \cs_new_protected:Npn \__color_backend_cmyk_aux:nnnn #1#2#3#4
484 {
485   \__color_backend_select:n
486   { #1 ~ #2 ~ #3 ~ #4 ~ k ~ #1 ~ #2 ~ #3 ~ #4 ~ K }
487 }
488 \cs_new_protected:Npn \__color_backend_gray:n #1
489 { \exp_args:Nx \__color_backend_gray_aux:n { \fp_eval:n {#1} } }
490 \cs_new_protected:Npn \__color_backend_gray_aux:n #1

```

```

491 { \_color_backend_select:n { #1 ~ g ~ #1 ~ G } }
492 \cs_new_protected:Npn \_color_backend_rgb:nnn #1#2#3
493 {
494   \use:x
495   {
496     \_color_backend_rgb_aux:nnn
497     { \fp_eval:n {#1} }
498     { \fp_eval:n {#2} }
499     { \fp_eval:n {#3} }
500   }
501 }
502 \cs_new_protected:Npn \_color_backend_rgb_aux:nnn #1#2#3
503 { \_color_backend_select:n { #1 ~ #2 ~ #3 ~ rg ~ #1 ~ #2 ~ #3 ~ RG } }
504 \cs_new_protected:Npn \_color_backend_spot:nn #1#2
505 { \_color_backend_select:n { /#1 ~ cs ~ /#1 ~ CS ~ #2 ~ sc ~ #2 ~ SC } }
506 \cs_new_protected:Npx \_color_backend_select:n #1
507 {
508   \cs_if_exist:NTF \tex_pdfextension:D
509   { \tex_pdfextension:D colorstack }
510   { \tex_pdfcolorstack:D }
511   \exp_not:N \l__kernel_color_stack_int push {#1}
512   \group_insert_after:N \exp_not:N \_color_backend_reset:
513 }
514 \cs_generate_variant:Nn \_color_backend_select:n { x }
515 \cs_new_protected:Npx \_color_backend_reset:
516 {
517   \cs_if_exist:NTF \tex_pdfextension:D
518   { \tex_pdfextension:D colorstack }
519   { \tex_pdfcolorstack:D }
520   \exp_not:N \l__kernel_color_stack_int pop \scan_stop:
521 }

```

(End definition for _color_backend_cmyk:nnnn and others.)

```

522 </pdfmode>
523 </initex | package>

```

4 l3backend-draw Implementation

```

524 <*initex | package>
525 <@@=draw>

```

4.1 dvips backend

```

526 <*dvips>

```

```

\_draw_backend_literal:n The same as literal PostScript: same arguments about positioning apply her.
\_draw_backend_literal:x 527 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
528 \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
\_draw_backend_end: matching ps::[end]: contrast with ps:, which positions but where we can't split material
color.fc 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.

```

529 \cs_new_protected:Npn \__draw_backend_begin:
530 {
531   \__kernel_backend_literal:n { ps::[begin] }
532   \__draw_backend_literal:n { @beginspecial }
533   \__draw_backend_literal:n { SDict ~ begin ~ /color.fc ~ { } ~ def ~ end }
534 }
535 \cs_new_protected:Npn \__draw_backend_end:
536 {
537   \__draw_backend_literal:n { @endspecial }
538   \__kernel_backend_literal:n { ps::[end] }
539 }

```

(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.

```

540 \cs_new_protected:Npn \__draw_backend_scope_begin:
541 { \__draw_backend_literal:n { save } }
542 \cs_new_protected:Npn \__draw_backend_scope_end:
543 { \__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.

```

544 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
545 {
546   \__draw_backend_literal:x
547   {
548     \dim_to_decimal_in_bp:n {#1} ~
549     \dim_to_decimal_in_bp:n {#2} ~ moveto
550   }
551 }
552 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
553 {
554   \__draw_backend_literal:x
555   {
556     \dim_to_decimal_in_bp:n {#1} ~
557     \dim_to_decimal_in_bp:n {#2} ~ lineto
558   }
559 }
560 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
561 {
562   \__draw_backend_literal:x

```



```

563     {
564         \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
565         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
566         moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
567     }
568 }
569 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
570 {
571     \__draw_backend_literal:x
572     {
573         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
574         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
575         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
576         curveto
577     }
578 }

```

(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
579 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
580 { \bool_gset_true:N \g__draw_draw_eor_bool }
581 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
582 { \bool_gset_false:N \g__draw_draw_eor_bool }
583 \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 also desirable to have the clip keyword after a stroke or fill. To achieve those outcomes, there is some work to do. For color, the stroke color is simple but the fill one has to be inserted by hand. For clipping, the required ordering is achieved using a T_EX switch. All of the operations end with a new path instruction as they do not terminate (again in contrast to PDF).

```

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

```

```

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

```

```

657         \bool_if:NT \g__draw_draw_eor_bool { eo }
658         clip
659     }
660 }
661 \__draw_backend_literal:n { newpath }
662 \bool_gset_false:N \g__draw_draw_clip_bool
663 }

```

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

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

```

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

```

696 \cs_new_protected:Npn \__draw_backend_color_fill_cmyk:nnnn #1#2#3#4
697 {
698     \__draw_backend_color_fill:x
699     {
700         \fp_eval:n {#1} ~ \fp_eval:n {#2} ~

```

```

701         \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
702         setcmykcolor
703     }
704 }
705 \cs_new_protected:Npn \__draw_backend_color_stroke_cmyk:nnnn #1#2#3#4
706 {
707     \__draw_backend_color_stroke:x
708     {
709         cmyk ~
710         \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
711         \fp_eval:n {#3} ~ \fp_eval:n {#4}
712     }
713 }
714 \cs_new_protected:Npn \__draw_backend_color_fill_gray:n #1
715 { \__draw_backend_color_fill:x { \fp_eval:n {#1} ~ setgray } }
716 \cs_new_protected:Npn \__draw_backend_color_stroke_gray:n #1
717 { \__draw_backend_color_stroke:x { gray ~ \fp_eval:n {#1} } }
718 \cs_new_protected:Npn \__draw_backend_color_fill_rgb:nnn #1#2#3
719 {
720     \__draw_backend_color_fill:x
721     { \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} ~ setrgbcolor }
722 }
723 \cs_new_protected:Npn \__draw_backend_color_stroke_rgb:nnn #1#2#3
724 {
725     \__draw_backend_color_stroke:x
726     { rgb ~ \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} }
727 }
728 \cs_new_protected:Npn \__draw_backend_color_fill:n #1
729 {
730     \__kernel_backend_postscript:n
731     { SDict ~ begin ~ /color.fc ~ { #1 } ~ def ~ end }
732 }
733 \cs_generate_variant:Nn \__draw_backend_color_fill:n { x }
734 \cs_new_protected:Npn \__draw_backend_color_stroke:n #1
735 {
736     \__kernel_backend_literal:n { color-push~#1 }
737     \group_insert_after:N \__draw_color_reset:
738 }
739 \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.

```

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

```

```

748     ] ~
749     concat
750   }
751 }

```

(End definition for `_draw_backend_cm:nnnn`.)

`_draw_backend_box_use:Nnnnn`

Inside a picture `@beginspecial/@endspecial` are active, which is normally a good thing but means that the position and scaling would be off if the box was inserted directly. To deal with that, there are a number of possible approaches. The implementation here was suggested by Tom Rokici (author of `dvips`). We end the current special placement, then set the current point with a literal `[begin]`. As for general literals, we then use the stack to store the current point and move to it. To insert the required transformation, we have to flip the y -axis, once before and once after it. Then we get back to the \TeX reference point to insert our content. The clean up has to happen in the right places, hence the `[begin]/[end]` pair around `restore`. Finally, we can return to “normal” drawing mode. Notice that the set up here is very similar to that in `_draw_align_currentpoint_...`, but the ordering of saving and restoring is different (intermixed).

```

752 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
753 {
754   \_draw_backend_literal:n { @endspecial }
755   \_draw_backend_literal:n { [end] }
756   \_draw_backend_literal:n { [begin] }
757   \_draw_backend_literal:n { save }
758   \_draw_backend_literal:n { currentpoint }
759   \_draw_backend_literal:n { currentpoint~translate }
760   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
761   \_draw_backend_cm:nnnn { #2 } { #3 } { #4 } { #5 }
762   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
763   \_draw_backend_literal:n { neg~exch~neg~exch~translate }
764   \_draw_backend_literal:n { [end] }
765   \hbox_overlap_right:n { \box_use:N #1 }
766   \_draw_backend_literal:n { [begin] }
767   \_draw_backend_literal:n { restore }
768   \_draw_backend_literal:n { [end] }
769   \_draw_backend_literal:n { [begin] }
770   \_draw_backend_literal:n { @beginspecial }
771 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

772 `</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.

773 `<*dvipdfmx | pdfmode | xdvipdfmx>`

4.2.1 Drawing

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

`_draw_backend_literal:x` 774 `\cs_new_eq:NN _draw_backend_literal:n _kernel_backend_literal_pdf:n`
775 `\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:` 776 `\cs_new_protected:Npn _draw_backend_begin:`
777 `{ _draw_backend_scope_begin: }`
778 `\cs_new_protected:Npn _draw_backend_end:`
779 `{ _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:` 780 `\cs_new_eq:NN _draw_backend_scope_begin: _kernel_backend_scope_begin:`
781 `\cs_new_eq:NN _draw_backend_scope_end: _kernel_backend_scope_end:`

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

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

`_draw_backend_lineto:nn` to convert to bp.

`_draw_backend_curveto:nnnnnn` 782 `\cs_new_protected:Npn _draw_backend_moveto:nn #1#2`
783 `{`
784 `_draw_backend_literal:x`
785 `{ \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }`
786 `}`
787 `\cs_new_protected:Npn _draw_backend_lineto:nn #1#2`
788 `{`
789 `_draw_backend_literal:x`
790 `{ \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }`
791 `}`
792 `\cs_new_protected:Npn _draw_backend_curveto:nnnnnn #1#2#3#4#5#6`
793 `{`
794 `_draw_backend_literal:x`
795 `{`
796 `\dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~`
797 `\dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~`
798 `\dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~`
799 `c`
800 `}`
801 `}`
802 `\cs_new_protected:Npn _draw_backend_rectangle:nnnn #1#2#3#4`
803 `{`
804 `_draw_backend_literal:x`
805 `{`
806 `\dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~`
807 `\dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~`
808 `re`
809 `}`
810 `}`

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

```

\__draw_backend_evenodd_rule:
\__draw_backend_nonzero_rule:
\g__draw_draw_eor_bool

```

The even-odd rule here can be implemented as a simply switch.

```

811 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
812 { \bool_gset_true:N \g__draw_draw_eor_bool }
813 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
814 { \bool_gset_false:N \g__draw_draw_eor_bool }
815 \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:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:

```

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

```

816 \cs_new_protected:Npn \__draw_backend_closepath:
817 { \__draw_backend_literal:n { h } }
818 \cs_new_protected:Npn \__draw_backend_stroke:
819 { \__draw_backend_literal:n { S } }
820 \cs_new_protected:Npn \__draw_backend_closestroke:
821 { \__draw_backend_literal:n { s } }
822 \cs_new_protected:Npn \__draw_backend_fill:
823 {
824   \__draw_backend_literal:x
825   { f \bool_if:NT \g__draw_draw_eor_bool * }
826 }
827 \cs_new_protected:Npn \__draw_backend_fillstroke:
828 {
829   \__draw_backend_literal:x
830   { B \bool_if:NT \g__draw_draw_eor_bool * }
831 }
832 \cs_new_protected:Npn \__draw_backend_clip:
833 {
834   \__draw_backend_literal:x
835   { W \bool_if:NT \g__draw_draw_eor_bool * }
836 }
837 \cs_new_protected:Npn \__draw_backend_discardpath:
838 { \__draw_backend_literal:n { n } }

```

(End definition for __draw_backend_closepath: and others.)

```

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

```

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

```

839 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
840 {
841   \__draw_backend_literal:x
842   {
843     [
844       \exp_args:Nf \use:n
845       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
846     ] ~
847     \dim_to_decimal_in_bp:n {#2} ~ d
848   }
849 }
850 \cs_new:Npn \__draw_backend_dash:n #1
851 { ~ \dim_to_decimal_in_bp:n {#1} }
852 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
853 {
854   \__draw_backend_literal:x

```

```

855     { \dim_to_decimal_in_bp:n {#1} ~ w }
856   }
857   \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
858     { \__draw_backend_literal:x { \fp_eval:n {#1} ~ M } }
859   \cs_new_protected:Npn \__draw_backend_cap_but:
860     { \__draw_backend_literal:n { 0 ~ J } }
861   \cs_new_protected:Npn \__draw_backend_cap_round:
862     { \__draw_backend_literal:n { 1 ~ J } }
863   \cs_new_protected:Npn \__draw_backend_cap_rectangle:
864     { \__draw_backend_literal:n { 2 ~ J } }
865   \cs_new_protected:Npn \__draw_backend_join_miter:
866     { \__draw_backend_literal:n { 0 ~ j } }
867   \cs_new_protected:Npn \__draw_backend_join_round:
868     { \__draw_backend_literal:n { 1 ~ j } }
869   \cs_new_protected:Npn \__draw_backend_join_bevel:
870     { \__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:
871 \cs_new_protected:Npn \__draw_backend_color_fill_cmyk:nnnn #1#2#3#4
872 {
873   \__draw_backend_color_select:x
874   {
875     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
876     \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
877     k
878   }
879 }
880 \cs_new_protected:Npn \__draw_backend_color_stroke_cmyk:nnnn #1#2#3#4
881 {
882   \__draw_backend_color_select:x
883   {
884     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
885     \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
886     k
887   }
888 }
889 \cs_new_protected:Npn \__draw_backend_color_fill_gray:n #1
890 { \__draw_backend_color_select:x { \fp_eval:n {#1} ~ g } }
891 \cs_new_protected:Npn \__draw_backend_color_stroke_gray:n #1
892 { \__draw_backend_color_select:x { \fp_eval:n {#1} ~ G } }
893 \cs_new_protected:Npn \__draw_backend_color_fill_rgb:nnn #1#2#3
894 {
895   \__draw_backend_color_select:x
896   { \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} ~ rg }
897 }
898 \cs_new_protected:Npn \__draw_backend_color_stroke_rgb:nnn #1#2#3
899 {
900   \__draw_backend_color_select:x
901   { \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} ~ RG }
902 }
903 \<pdfmode>

```



```

904 \cs_new_protected:Npx \__draw_backend_color_select:n #1
905 {
906   \cs_if_exist:NTF \tex_pdfextension:D
907   { \tex_pdfextension:D colorstack }
908   { \tex_pdfcolorstack:D }
909   \exp_not:N \l__kernel_color_stack_int push {#1}
910   \group_insert_after:N \exp_not:N \__draw_backend_color_reset:
911 }
912 \cs_new_protected:Npx \__draw_backend_color_reset:
913 {
914   \cs_if_exist:NTF \tex_pdfextension:D
915   { \tex_pdfextension:D colorstack }
916   { \tex_pdfcolorstack:D }
917   \exp_not:N \l__kernel_color_stack_int pop \scan_stop:
918 }
919 </pdfmode>
920 <*dvipdfmx | xdvipdfmx>
921 \cs_new_eq:NN \__draw_backend_color_select:n \__kernel_backend_literal_pdf:n
922 </dvipdfmx | xdvipdfmx>
923 \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 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.)

```

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

```

```

949     {
950       x:scale~
951       \fp_eval:n { round ( #2 , 5 ) } ~
952       \fp_eval:n { round ( #3 , 5 ) }
953     }
954     \__kernel_backend_literal:x
955     {
956       x:rotate~
957       \fp_compare:nNnTF {#4} = \c_zero_fp
958       { 0 }
959       { \fp_eval:n { round ( -#4 , 5 ) } }
960     }
961   }
962   </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.

```

963   <*dvipdfmx | xdvipdfmx>
964   \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
965   {
966     \use:x
967     {
968       \__draw_backend_cm_decompose_auxi:nnnnN
969       { \fp_eval:n { (#1 + #4) / 2 } }
970       { \fp_eval:n { (#1 - #4) / 2 } }
971       { \fp_eval:n { (#3 + #2) / 2 } }

```

```

972         { \fp_eval:n { (#3 - #2) / 2 } }
973     }
974     #5
975 }
976 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
977 {
978     \use:x
979     {
980         \__draw_backend_cm_decompose_auxii:nnnnN
981         { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
982         { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
983         { \fp_eval:n { atand ( #3 , #2 ) } }
984         { \fp_eval:n { atand ( #4 , #1 ) } }
985     }
986     #5
987 }
988 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
989 {
990     \use:x
991     {
992         \__draw_backend_cm_decompose_auxiii:nnnnN
993         { \fp_eval:n { ( #4 - #3 ) / 2 } }
994         { \fp_eval:n { ( #1 + #2 ) / 2 } }
995         { \fp_eval:n { ( #1 - #2 ) / 2 } }
996         { \fp_eval:n { ( #4 + #3 ) / 2 } }
997     }
998     #5
999 }
1000 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1001 {
1002     \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1003     { #5 {#1} {#2} {#3} {#4} }
1004     { #5 {#1} {#3} {#2} {#4} }
1005 }
1006 </dviPDFmx | xdvipdfmx>

```

(End definition for __draw_backend_cm_decompose:nnnnN and others.)

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

```

1007 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1008 {
1009     \__kernel_backend_scope_begin:
1010     <*pdfmode>
1011     \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1012     </pdfmode>
1013     <*dviPDFmx | xdvipdfmx>
1014     \__kernel_backend_literal:x
1015     {
1016         pdf:btrans-matrix~
1017         \fp_eval:n {#2} ~ \fp_eval:n {#3} ~

```

```

1018         \fp_eval:n {#4} ~ \fp_eval:n {#5} ~
1019         0 ~ 0
1020     }
1021 </dvipdfmx | xdvipdfmx>
1022     \hbox_overlap_right:n { \box_use:N #1 }
1023 <*dvipdfmx | xdvipdfmx>
1024     \__kernel_backend_literal:n { pdf:etrans }
1025 </dvipdfmx | xdvipdfmx>
1026     \__kernel_backend_scope_end:
1027 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1028 </dvipdfmx | pdfmode | xdvipdfmx>

```

4.3 dvisvgm backend

```

1029 <*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
1030 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1031 \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

```

1032 \cs_new_protected:Npn \__draw_backend_begin:
1033 {
1034     \__draw_backend_scope_begin:
1035     \__draw_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1036 }
1037 \cs_new_protected:Npn \__draw_backend_end:
1038 { \__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.

```

1039 \cs_new_protected:Npn \__draw_backend_scope_begin:
1040 {
1041     \int_set_eq:NN
1042     \l__draw_draw_scope_int
1043     \g__draw_draw_scope_int
1044     \group_begin:
1045     \int_gzero:N \g__draw_draw_scope_int
1046 }
1047 \cs_new_protected:Npn \__draw_backend_scope_end:
1048 {
1049     \prg_replicate:nn
1050     { \g__draw_draw_scope_int }

```

```

1051         { \_draw_backend_literal:n { </g> } }
1052     \group_end:
1053     \int_gset_eq:NN
1054         \g__draw_draw_scope_int
1055         \l__draw_draw_scope_int
1056 }
1057 \cs_new_protected:Npn \_draw_backend_scope:n #1
1058 {
1059     \_draw_backend_literal:n { <g~ #1 > }
1060     \int_gincr:N \g__draw_draw_scope_int
1061 }
1062 \cs_generate_variant:Nn \_draw_backend_scope:n { x }
1063 \int_new:N \g__draw_draw_scope_int
1064 \int_new:N \l__draw_draw_scope_int

```

(End definition for _draw_backend_scope_begin: and others.)

```

\_draw_backend_moveto:nn
\_draw_backend_lineto:nn
    \_draw_backend_rectangle:nnnn
    \_draw_backend_curveto:nnnnnn
    \_draw_backend_add_to_path:n
\g__draw_draw_path_tl

```

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.

```

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

```

```

1097 {
1098   \tl_gset:Nx \g__draw_draw_path_tl
1099   {
1100     \g__draw_draw_path_tl
1101     \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1102     #1
1103   }
1104 }
1105 \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:
1106 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1107 { \__draw_backend_scope:n { fill-rule="evenodd" } }
1108 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1109 { \__draw_backend_scope:n { fill-rule="nonzero" } }

```

(End definition for `__draw_backend_evenodd_rule:` and `__draw_backend_nonzero_rule:.`)

```

\__draw_backend_path:n Setting fill and stroke effects and doing clipping all has to be done using scopes. This
\__draw_backend_closepath: means setting up the various requirements in a shared auxiliary which deals with the
\__draw_backend_stroke: bits and pieces. Clipping paths are reused for path drawing: not essential but avoids
\__draw_backend_closestroke: constructing them twice. Discarding a path needs a separate function as it's not quite
\__draw_backend_fill: the same.
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int
1110 \cs_new_protected:Npn \__draw_backend_closepath:
1111 { \__draw_backend_add_to_path:n { Z } }
1112 \cs_new_protected:Npn \__draw_backend_path:n #1
1113 {
1114   \bool_if:NTF \g__draw_draw_clip_bool
1115   {
1116     \int_gincr:N \g__draw_clip_path_int
1117     \__draw_backend_literal:x
1118     {
1119       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1120       { ?nl }
1121       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1122       < /clipPath > { ? nl }
1123       <
1124       use~xlink:href =
1125       "\c_hash_str l3path \int_use:N \g__draw_path_int " ~
1126       #1
1127     } />
1128   }
1129   \__draw_backend_scope:x
1130   {
1131     clip-path =
1132     "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int )"
1133   }
1134 }
1135 {
1136   \__draw_backend_literal:x
1137   { <path ~ d=" \g__draw_draw_path_tl " ~ #1 /> }
1138 }

```

```

1139     \tl_gclear:N \g__draw_draw_path_tl
1140     \bool_gset_false:N \g__draw_draw_clip_bool
1141   }
1142   \int_new:N \g__draw_path_int
1143   \cs_new_protected:Npn \__draw_backend_stroke:
1144     { \__draw_backend_path:n { style="fill:none" } }
1145   \cs_new_protected:Npn \__draw_backend_closestroke:
1146     {
1147       \__draw_backend_closepath:
1148       \__draw_backend_stroke:
1149     }
1150   \cs_new_protected:Npn \__draw_backend_fill:
1151     { \__draw_backend_path:n { style="stroke:none" } }
1152   \cs_new_protected:Npn \__draw_backend_fillstroke:
1153     { \__draw_backend_path:n { } }
1154   \cs_new_protected:Npn \__draw_backend_clip:
1155     { \bool_gset_true:N \g__draw_draw_clip_bool }
1156   \bool_new:N \g__draw_draw_clip_bool
1157   \cs_new_protected:Npn \__draw_backend_discardpath:
1158     {
1159       \bool_if:NT \g__draw_draw_clip_bool
1160       {
1161         \int_gincr:N \g__draw_clip_path_int
1162         \__draw_backend_literal:x
1163         {
1164           < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1165           { ?nl }
1166           <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1167           < /clipPath >
1168         }
1169         \__draw_backend_scope:x
1170         {
1171           clip-path =
1172             "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1173         }
1174       }
1175     \tl_gclear:N \g__draw_draw_path_tl
1176     \bool_gset_false:N \g__draw_draw_clip_bool
1177   }

```

(End definition for __draw_backend_path:n and others.)

```

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

```

1178 \cs_new_protected:Npn __draw_backend_dash_pattern:nn #1#2
1179 {
1180 \use:x
1181 {
1182 __draw_backend_dash_aux:nn
1183 { \clist_map_function:nn {#1} __draw_backend_dash:n }
1184 { \dim_to_decimal:n {#2} }
1185 }
1186 }
1187 \cs_new:Npn __draw_backend_dash:n #1

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

```

1188 { , \dim_to_decimal_in_bp:n {#1} }
1189 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1190 {
1191   \__draw_backend_scope:x
1192   {
1193     stroke-dasharray =
1194     "
1195     \tl_if_empty:oTF { \use_none:n #1 }
1196     { none }
1197     { \use_none:n #1 }
1198     " ~
1199     stroke-offset=" #2 "
1200   }
1201 }
1202 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1203 { \__draw_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1204 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1205 { \__draw_backend_scope:x { stroke-miterlimit=" \fp_eval:n {#1} " } }
1206 \cs_new_protected:Npn \__draw_backend_cap_but:
1207 { \__draw_backend_scope:n { stroke-linecap="butt" } }
1208 \cs_new_protected:Npn \__draw_backend_cap_round:
1209 { \__draw_backend_scope:n { stroke-linecap="round" } }
1210 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1211 { \__draw_backend_scope:n { stroke-linecap="square" } }
1212 \cs_new_protected:Npn \__draw_backend_join_miter:
1213 { \__draw_backend_scope:n { stroke-linejoin="miter" } }
1214 \cs_new_protected:Npn \__draw_backend_join_round:
1215 { \__draw_backend_scope:n { stroke-linejoin="round" } }
1216 \cs_new_protected:Npn \__draw_backend_join_bevel:
1217 { \__draw_backend_scope:n { stroke-linejoin="bevel" } }

```

(End definition for __draw_backend_dash_pattern:nn and others.)

_draw_backend_color_fill_cmyk:nnnn 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 1218 \cs_new_protected:Npn \__draw_backend_color_fill_cmyk:nnnn #1#2#3#4
\_draw_backend_color_stroke_gray:n 1219 {
\_draw_backend_color_fill_rgb:nnn 1220   \use:x
\_draw_backend_color_stroke_rgb:nnn 1221   {
\_draw_backend_color_fill:nnn 1222     \__draw_backend_color_fill:nnn
1223     { \fp_eval:n { -100 * ( (#1) * ( 1 - (#4) ) - 1 ) } }
1224     { \fp_eval:n { -100 * ( (#2) * ( 1 - (#4) ) + #4 - 1 ) } }
1225     { \fp_eval:n { -100 * ( (#3) * ( 1 - (#4) ) + #4 - 1 ) } }
1226   }
1227 }
1228 \cs_new_protected:Npn \__draw_backend_color_stroke_cmyk:nnnn #1#2#3#4
1229 {
1230   \__draw_backend_select:x
1231   {
1232     cmyk~
1233     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
1234     \fp_eval:n {#3} ~ \fp_eval:n {#4}
1235   }
1236 }

```



```

1237 \cs_new_protected:Npn \__draw_backend_color_fill_gray:n #1
1238 {
1239   \use:x
1240   {
1241     \__draw_backend_color_gray_aux:n
1242     { \fp_eval:n { 100 * (#1) } }
1243   }
1244 }
1245 \cs_new_protected:Npn \__draw_backend_color_gray_aux:n #1
1246 { \__draw_backend_color_fill:nnn {#1} {#1} {#1} }
1247 \cs_new_protected:Npn \__draw_backend_color_stroke_gray:n #1
1248 { \__draw_backend_select:x { gray~ \fp_eval:n {#1} } }
1249 \cs_new_protected:Npn \__draw_backend_color_fill_rgb:nnn #1#2#3
1250 {
1251   \use:x
1252   {
1253     \__draw_backend_color_fill:nnn
1254     { \fp_eval:n { 100 * (#1) } }
1255     { \fp_eval:n { 100 * (#2) } }
1256     { \fp_eval:n { 100 * (#3) } }
1257   }
1258 }
1259 \cs_new_protected:Npn \__draw_backend_color_fill:nnn #1#2#3
1260 {
1261   \__draw_backend_scope:x
1262   {
1263     fill =
1264     "
1265     rgb
1266     (
1267       #1 \c_percent_str ,
1268       #2 \c_percent_str ,
1269       #3 \c_percent_str
1270     )
1271     "
1272   }
1273 }
1274 \cs_new_protected:Npn \__draw_backend_color_stroke_rgb:nnn #1#2#3
1275 {
1276   \__draw_backend_select:x
1277   { rgb~ \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} }
1278 }

```

(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.

```

1279 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1280 {
1281   \__draw_backend_scope:n
1282   {
1283     transform =
1284     "
1285     matrix

```

```

1286      (
1287        \fp_eval:n {#1} , \fp_eval:n {#2} ,
1288        \fp_eval:n {#3} , \fp_eval:n {#4} ,
1289        Opt , Opt
1290      )
1291      "
1292    }
1293  }

```

(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.

```

1294 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1295 {
1296   \_kernel_backend_scope_begin:
1297   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1298   \_kernel_backend_literal_svg:n
1299   {
1300     < g~
1301     stroke="none"~
1302     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1303     >
1304   }
1305   \box_set_wd:Nn #1 { Opt }
1306   \box_set_ht:Nn #1 { Opt }
1307   \box_set_dp:Nn #1 { Opt }
1308   \box_use:N #1
1309   \_kernel_backend_literal_svg:n { </g> }
1310   \_kernel_backend_scope_end:
1311 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```

1312 </dvisvgm>
1313 </initex | package>

```

5 l3backend-graphics Implementation

```

1314 <*initex | package>
1315 <@@=graphics>

```

5.1 dvips backend

```

1316 <*dvips>

```

`_graphics_backend_getbb_eps:n` Simply use the generic function.

```

1317 <*initex>
1318 \use:n
1319 </initex>
1320 <*package>
1321 \AtBeginDocument
1322 </package>
1323 { \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.

```

1324 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1325 {
1326   \_kernel_backend_literal:x
1327   {
1328     PSfile = #1 \c_space_tl
1329     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1330     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1331     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1332     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1333   }
1334 }

```

(End definition for _graphics_backend_include_eps:n.)

```

1335 </dvips>

```

5.2 pdfmode backend

```

1336 <{*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.

```

1337 \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.

```

1338 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1339 {
1340   \int_zero:N \l_graphics_page_int
1341   \tl_clear:N \l_graphics_pagebox_tl
1342   \tl_set:Nx \l_graphics_graphics_attr_tl
1343   {
1344     \tl_if_empty:NF \l_graphics_decodearray_tl
1345     { :D \l_graphics_decodearray_tl }
1346     \bool_if:NT \l_graphics_interpolate_bool
1347     { :I }
1348   }
1349   \tl_clear:N \l_graphics_graphics_attr_tl
1350   \_graphics_backend_getbb_auxi:n {#1}
1351 }
1352 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1353 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1354 {

```

```

1355 \tl_clear:N \l_graphics_decodearray_tl
1356 \bool_set_false:N \l_graphics_interpolate_bool
1357 \tl_set:Nx \l__graphics_graphics_attr_tl
1358 {
1359   : \l_graphics_pagebox_tl
1360   \int_compare:nNnT \l_graphics_page_int > 1
1361     { :P \int_use:N \l_graphics_page_int }
1362 }
1363 \__graphics_backend_getbb_auxi:n {#1}
1364 }
1365 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1366 {
1367   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1368   { \__graphics_backend_getbb_auxii:n {#1} }
1369 }
1370 % \begin{macrocode}
1371 % Measuring the graphic is done by boxing up: for PDF graphics we could
1372 % use \tex_pdfximagebbox:D/, but if doesn't work for other types.
1373 % As the box always starts at $(0,0)$ there is no need to worry about
1374 % the lower-left position.
1375 % \begin{macrocode}
1376 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1377 {
1378   \tex_immediate:D \tex_pdfximage:D
1379   \bool_lazy_or:nnT
1380     { \l_graphics_interpolate_bool }
1381     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1382     {
1383       attr ~
1384       {
1385         \tl_if_empty:NF \l_graphics_decodearray_tl
1386         { /Decode~[ \l_graphics_decodearray_tl ] }
1387         \bool_if:NT \l_graphics_interpolate_bool
1388         { /Interpolate~true }
1389       }
1390     }
1391   \int_compare:nNnT \l_graphics_page_int > 0
1392     { page ~ \int_use:N \l_graphics_page_int }
1393   \tl_if_empty:NF \l_graphics_pagebox_tl
1394     { \l_graphics_pagebox_tl }
1395   {#1}
1396   \hbox_set:Nn \l__graphics_internal_box
1397     { \tex_pdfrefximage:D \tex_pdflastximage:D }
1398   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1399   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1400   \int_const:cn { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1401     { \tex_the:D \tex_pdflastximage:D }
1402   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1403 }

```

(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 straight-forward, with only any attributes to worry about. The latter carry through from deter-

_graphics_backend_include_pdf:n

_graphics_backend_include_png:n

mination of the bounding box.

```

1404 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1405 {
1406   \tex_pdfrefximage:D
1407   \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl_int }
1408 }
1409 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1410 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

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

```

EPS graphics may be included in pdfmode by conversion to PDF: this requires restricted shell escape. Modelled on the `epstopdf` L^AT_EX 2_ε package, but simplified, conversion takes place here if we have shell access.

```

\__graphics_backend_getbb_eps:n
\__graphics_backend_getbb_eps:nm
\__graphics_backend_include_eps:n
\l__graphics_backend_dir_str
\l__graphics_backend_name_str
\l__graphics_backend_ext_str
1411 \sys_if_shell:T
1412 {
1413   \str_new:N \l__graphics_backend_dir_str
1414   \str_new:N \l__graphics_backend_name_str
1415   \str_new:N \l__graphics_backend_ext_str
1416   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1417   {
1418     \file_parse_full_name:nNNN {#1}
1419     \l__graphics_backend_dir_str
1420     \l__graphics_backend_name_str
1421     \l__graphics_backend_ext_str
1422     \exp_args:Nx \__graphics_backend_getbb_eps:nm
1423     {
1424       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1425       -converted-to.pdf
1426     }
1427     {#1}
1428   }
1429   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1430   {
1431     \file_compare_timestamp:nNnT {#2} > {#1}
1432     {
1433       \sys_shell_now:n
1434       { repstopdf ~ #2 ~ #1 }
1435     }
1436     \tl_set:Nn \l__graphics_name_tl {#1}
1437     \__graphics_backend_getbb_pdf:n {#1}
1438   }
1439   \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1440   {
1441     \file_parse_full_name:nNNN {#1}
1442     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1443     \exp_args:Nx \__graphics_backend_include_pdf:n
1444     {
1445       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1446       -converted-to.pdf
1447     }
1448   }
1449 }

```

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

1450 `</pdfmode>`

5.3 dvipdfmx backend

1451 `<*dvipdfmx | xdvipdfmx>`

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

```

\__graphics_backend_getbb_eps:n 1452 <*initex>
\__graphics_backend_getbb_jpg:n 1453 \use:n
\__graphics_backend_getbb_pdf:n 1454 </initex>
\__graphics_backend_getbb_png:n 1455 <*package>
1456 \AtBeginDocument
1457 </package>
1458 { \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n }
1459 <*dvipdfmx>
1460 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1461 {
1462   \int_zero:N \l_graphics_page_int
1463   \tl_clear:N \l_graphics_pagebox_tl
1464   \graphics_extract_bb:n {#1}
1465 }
1466 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1467 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1468 {
1469   \tl_clear:N \l_graphics_decodearray_tl
1470   \bool_set_false:N \l_graphics_interpolate_bool
1471   \graphics_extract_bb:n {#1}
1472 }
1473 </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.

1474 `\int_new:N \g__graphics_track_int`

(End definition for `\g__graphics_track_int`.)

The special syntax depends on the file type. There is a difference in how PDF graphics are best handled between dvipdfmx and 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_eps:n 1475 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
\__graphics_backend_include_jpg:n 1476 {
\__graphics_backend_include_pdf:n 1477   \__kernel_backend_literal:x
\__graphics_backend_include_png:n 1478 {
\__graphics_backend_include_auxi:nn 1479     PSfile = #1 \c_space_tl
\__graphics_backend_include_auxii:nnn 1480     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
\__graphics_backend_include_auxii:xnn 1481     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
\__graphics_backend_include_auxiii:nnn 1482     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1483     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1484   }
1485 }
1486 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1487 { \__graphics_backend_include_auxi:nn {#1} { image } }

```

```

1488 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
1489 <*dvipdfmx>
1490 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1491 { \__graphics_backend_include_auxi:nn {#1} { epdf } }
1492 </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.

```

1493 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1494 {
1495   \__graphics_backend_include_auxii:xnn
1496   {
1497     \tl_if_empty:NF \l_graphics_pagebox_tl
1498     { : \l_graphics_pagebox_tl }
1499     \int_compare:nNnT \l_graphics_page_int > 1
1500     { :P \int_use:N \l_graphics_page_int }
1501     \tl_if_empty:NF \l_graphics_decodearray_tl
1502     { :D \l_graphics_decodearray_tl }
1503     \bool_if:NT \l_graphics_interpolate_bool
1504     { :I }
1505   }
1506   {#1} {#2}
1507 }
1508 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1509 {
1510   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1511   {
1512     \__kernel_backend_literal:x
1513     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1514   }
1515   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1516 }
1517 \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!

```

1518 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1519 {
1520   \int_gincr:N \g__graphics_track_int
1521   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1522   \__kernel_backend_literal:x
1523   {
1524     pdf:#3~
1525     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1526     \int_compare:nNnT \l_graphics_page_int > 1
1527     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1528     \tl_if_empty:NF \l_graphics_pagebox_tl
1529     {
1530       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1531       bbox ~
1532       \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl

```

```

1533         \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1534         \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1535         \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1536     }
1537     (#1)
1538     \bool_lazy_or:nnT
1539     { \l_graphics_interpolate_bool }
1540     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1541     {
1542         <<
1543         \tl_if_empty:NF \l_graphics_decodearray_tl
1544         { /Decode~[ \l_graphics_decodearray_tl ] }
1545         \bool_if:NT \l_graphics_interpolate_bool
1546         { /Interpolate~true> }
1547         >>
1548     }
1549 }
1550 }

```

(End definition for _graphics_backend_include_eps:n and others.)

```
1551 </dviPDFmx | xdvipdfmx>
```

5.4 xdvipdfmx backend

```
1552 < *xdvipdfmx>
```

5.4.1 Images

For `xdvipdfmx`, 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 XeTeX primitive omits the text box from the page box specification, so there is also some “trimming” to do here.

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
\__graphics_backend_getbb_auxi:nN
\__graphics_backend_getbb_auxii:nNn
\__graphics_backend_getbb_auxiii:nNnn
\__graphics_backend_getbb_auxiv:nNnn
\__graphics_backend_getbb_auxv:nNnn
\__graphics_backend_getbb_auxv:nNnn
\__graphics_backend_getbb_pagebox:w

```

```

1553 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1554 {
1555     \int_zero:N \l_graphics_page_int
1556     \tl_clear:N \l_graphics_pagebox_tl
1557     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1558 }
1559 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1560 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1561 {
1562     \tl_clear:N \l_graphics_decodearray_tl
1563     \bool_set_false:N \l_graphics_interpolate_bool
1564     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdfvfile:D
1565 }
1566 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
1567 {
1568     \int_compare:nNnTF \l_graphics_page_int > 1
1569     { \__graphics_backend_getbb_auxii:nNn \l_graphics_page_int {#1} #2 }
1570     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1571 }
1572 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nNn #1#2#3
1573 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1574 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nNn { V }

```



```

1575 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1576 {
1577   \tl_if_empty:NTF \l_graphics_pagebox_tl
1578   { \__graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1579   { \__graphics_backend_getbb_auxv:nNnn }
1580   {#1} #2 {#3} {#4}
1581 }
1582 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
1583 {
1584   \use:x
1585   {
1586     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1587     { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1588   }
1589 }
1590 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1591 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1592 {
1593   \graphics_bb_restore:nF {#1#3}
1594   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1595 }
1596 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1597 {
1598   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1599   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1600   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1601   \graphics_bb_save:n {#1#3}
1602 }
1603 \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

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

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

```

1604 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1605 {
1606   \tex_XeTeXpdffile:D
1607   \__graphics_backend_include_pdf_quote:w #1 "#1" \q_stop \c_space_tl
1608   \int_compare:nNnT \l_graphics_page_int > 0
1609   { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1610   \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1611 }
1612 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \q_stop
1613 { " #2 " }

```

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

```

1614 </xdvipdmtx>

```

5.5 dvisvgm backend

```

1615 < *dvisvgm>

```

`_graphics_backend_getbb_png:n`
`_graphics_backend_getbb_jpg:n`

These can be included by extracting the bounding box data.

```

1616 <*initex>
1617 \use:n
1618 </initex>
1619 <*package>
1620 \AtBeginDocument
1621 </package>
1622 {
1623   \cs_new_eq:NN \_graphics_backend_getbb_png:n \graphics_extract_bb:n
1624   \cs_new_eq:NN \_graphics_backend_getbb_jpg:n \graphics_extract_bb:n
1625 }

```

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

`_graphics_backend_include_png:n`
`_graphics_backend_include_jpg:n`
`_graphics_backend_include_bitmap_quote:w`

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

The only issue is that `#1` must be quote-corrected. The `dvisvgm:img` operation quotes the file name, but if it is already quoted (contains spaces) then we have an issue: we simply strip off any quotes as a result.

```

1626 \cs_new_protected:Npn \_graphics_backend_include_png:n #1
1627 {
1628   \_kernel_backend_literal:x
1629   {
1630     dvisvgm:img~
1631     \dim_to_decimal:n { \l_graphics_ury_dim } ~
1632     \dim_to_decimal:n { \l_graphics_ury_dim } ~
1633     \_graphics_backend_include_bitmap_quote:w #1 " #1 " \q_stop
1634   }
1635 }
1636 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_png:n
1637 \cs_new:Npn \_graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \q_stop
1638 { " #2 " }

```

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

```

1639 </dvisvgm>
1640 </initex | package>

```

6 l3backend-pdf Implementation

```

1641 <*initex | package>
1642 <@@=pdf>

```

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

6.1 Shared code

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

`\l__pdf_internal_boxf`

```
1643 \box_new:N \l__pdf_internal_box
```

(End definition for `\l__pdf_internal_boxf`.)

6.2 dvips backend

```
1644 ⟨*dvips⟩
```

`__pdf_backend_pdfmark:n`

Used often enough it should be a separate function.

`__pdf_backend_pdfmark:x`

```
1645 \cs_new_protected:Npn \__pdf_backend_pdfmark:n #1
1646 { \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }
1647 \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`

```
1648 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
1649 { \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
1650 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
1651 { \__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`

```
1652 \int_new:N \g__pdf_backend_object_int
1653 \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`

```
1654 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
1655 {
1656   \int_gincr:N \g__pdf_backend_object_int
1657   \int_const:cn
1658   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
1659   { \g__pdf_backend_object_int }
1660   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
1661 }
1662 \cs_new:Npn \__pdf_backend_object_ref:n #1
1663 { { 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`

`_pdf_backend_object_write_array:nn`

```
1664 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
1665 {
1666   \__pdf_backend_pdfmark:x
1667   {
1668     /_objdef ~ \__pdf_backend_object_ref:n {#1}
1669     /type
```

`_pdf_backend_object_write_dict:nn`

`_pdf_backend_object_write_stream:nn`

`_pdf_backend_object_write_stream:nnn`

```

1670     \str_case:e:nn
1671     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
1672     {
1673       { array }    { /array }
1674       { dict }     { /dict }
1675       { fstream }  { /stream }
1676       { stream }   { /stream }
1677     }
1678   /OBJ
1679 }
1680 \use:c
1681 { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
1682 { \__pdf_backend_object_ref:n {#1} } {#2}
1683 }
1684 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
1685 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
1686 {
1687   \__pdf_backend_pdfmark:x
1688   { #1 [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
1689 }
1690 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
1691 {
1692   \__pdf_backend_pdfmark:x
1693   { #1 << \exp_not:n {#2} >> /PUT }
1694 }
1695 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
1696 {
1697   \exp_args:Nx
1698   \__pdf_backend_object_write_stream:nnn {#1} #2
1699 }
1700 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
1701 {
1702   \__pdf_postscript:n
1703   {
1704     [nobreak]
1705     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
1706     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
1707   }
1708 }

```

(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
1709 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
1710 {
1711   \int_gincr:N \g__pdf_backend_object_int
1712   \__pdf_backend_pdfmark:x
1713   {
1714     /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
1715     /type
1716     \str_case:nn
1717     {#1}
1718     {
1719       { array }    { /array }

```

```

1720         { dict }    { /dict }
1721         { fstream } { /stream }
1722         { stream }  { /stream }
1723     }
1724     /OBJ
1725 }
1726 \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
1727 { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
1728 }
1729 \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.

```

1730 \cs_new:Npn \__pdf_backend_object_last:
1731 { { 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.

```

1732 \__kernel_backend_postscript_header:n
1733 {
1734     true ~ setglobal ~
1735     /pdf.globaldict ~ 4 ~ dict ~ def ~
1736     false ~ setglobal
1737 }

```

(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 to allow for Resolution. The total height of a rectangle (an array) needs a little maths, in contrast to simply extracting a value.

```

1738 \__kernel_backend_postscript_header:n
1739 {
1740     /pdf.cvs { 65534 ~ string ~ cvs } def
1741     /pdf.dvi.pt { 72.27 ~ mul ~ Resolution ~ div } def
1742     /pdf.pt.dvi { 72.27 ~ div ~ Resolution ~ mul } def
1743     /pdf.rect.ht { dup ~ 1 ~ get ~ neg ~ exch ~ 3 ~ get ~ add } def
1744 }

```

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

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

```

1745 \__kernel_backend_postscript_header:n
1746 {
1747     /pdf.linkmargin { 1 ~ pdf.pt.dvi } def
1748     /pdf.linkdp.pad { 0 } def
1749     /pdf.linkht.pad { 0 } def
1750 }

```

(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 1751 \_kernel_backend_postscript_header:n
pdf.save.linkur 1752 {
    pdf.llx      1753     /pdf.rect
    pdf.lly      1754     { /Rect [ pdf.llx ~ pdf.lly ~ pdf.urx ~ pdf.ury ] } def
    pdf.urx      1755     /pdf.save.ll
    pdf.ury      1756     {
        1757         currentpoint
        1758         /pdf.lly ~ exch ~ def
        1759         /pdf.llx ~ exch ~ def
        1760     }
        1761     def
    1762     /pdf.save.ur
    1763     {
        1764         currentpoint
        1765         /pdf.ury ~ exch ~ def
        1766         /pdf.urx ~ exch ~ def
        1767     }
        1768     def
    1769     /pdf.save.linkll
    1770     {
        1771         currentpoint ~
        1772         pdf.linkmargin ~ add ~
        1773         pdf.linkdp.pad ~ add
        1774         /pdf.lly ~ exch ~ def ~
        1775         pdf.linkmargin ~ sub
        1776         /pdf.llx ~ exch ~ def
        1777     }
        1778     def
    1779     /pdf.save.linkur
    1780     {
        1781         currentpoint ~
        1782         pdf.linkmargin ~ sub ~
        1783         pdf.linkht.pad ~ sub
        1784         /pdf.ury ~ exch ~ def ~
        1785         pdf.linkmargin ~ add
        1786         /pdf.urx ~ exch ~ def
        1787     }
        1788     def
    1789 }
```

(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
`pdf.dest.x` function as it comes up a lot, and as this makes it easier to adjust if we need additional
`pdf.dest.y` effects. We also need a more complex approach to convert a co-ordinate pair correctly
`pdf.dest.point` when defining a rectangle: this can otherwise be out when using a landscape page.
`pdf.dest2device` (Thanks to Alexander Grahn for the approach here.)

```
pdf.dev.x 1790 \_kernel_backend_postscript_header:n
pdf.dev.y
pdf.tmpa
pdf.tmpb
pdf.tmpc
pdf.tmpd
```

```

1791 {
1792   /pdf.dest.anchor
1793   {
1794     currentpoint ~ exch ~
1795     pdf.dvi.pt ~ 72 ~ add ~
1796     /pdf.dest.x ~ exch ~ def ~
1797     pdf.dvi.pt ~
1798     vsize ~ 72 ~ sub ~ exch ~ sub ~
1799     /pdf.dest.y ~ exch ~ def
1800   }
1801   def
1802   /pdf.dest.point
1803   { pdf.dest.x ~ pdf.dest.y } def
1804   /pdf.dest2device
1805   {
1806     /pdf.dest.y ~ exch ~ def
1807     /pdf.dest.x ~ exch ~ def ~
1808     matrix ~ currentmatrix ~
1809     matrix ~ defaultmatrix ~
1810     matrix ~ invertmatrix ~
1811     matrix ~ concatmatrix ~
1812     cvx ~ exec
1813     /pdf.dev.y ~ exch ~ def
1814     /pdf.dev.x ~ exch ~ def
1815     /pdf.tmpd ~ exch ~ def
1816     /pdf.tmpc ~ exch ~ def
1817     /pdf.tmpb ~ exch ~ def
1818     /pdf.tmpa ~ exch ~ def ~
1819     pdf.dest.x ~ pdf.tmpa ~ mul ~
1820     pdf.dest.y ~ pdf.tmpc ~ mul ~ add ~
1821     pdf.dev.x ~ add ~
1822     pdf.dest.x ~ pdf.tmpb ~ mul ~
1823     pdf.dest.y ~ pdf.tmpd ~ mul ~ add ~
1824     pdf.dev.y ~ add
1825   }
1826   def
1827 }

```

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

pdf.bordertracking	To know where a breakable link can go, we need to track the boundary rectangle. That
pdf.bordertracking.begin	can be done by hooking into a and x operations: those names have to be retained. The
pdf.bordertracking.end	boundary is stored at the end of the operation. Special effort is needed at the start and
pdf.leftboundary	end of pages (or rather galleys), such that everything works properly.
pdf.rightboundary	
pdf.brokenlink.rect	1828 _kernel_backend_postscript_header:n
pdf.brokenlink.skip	1829 {
pdf.brokenlink.dict	1830 /pdf.bordertracking ~ false ~ def
pdf.bordertracking.endpage	1831 /pdf.bordertracking.begin
pdf.bordertracking.continue	1832 {
pdf.originx	1833 SDict ~ /pdf.bordertracking ~ true ~ put ~
pdf.originy	1834 SDict ~ /pdf.leftboundary ~ undef ~
	1835 SDict ~ /pdf.rightboundary ~ undef ~
	1836 /a ~ where
	1837 {

```

1838     /a
1839     {
1840         currentpoint ~ pop ~
1841         SDict /pdf.rightboundary ~ known ~ dup
1842         {
1843             SDict /pdf.rightboundary ~ get ~ 2 ~ index ~ lt
1844             { not }
1845             if
1846         }
1847         if
1848         { pop }
1849         { SDict ~ exch /pdf.rightboundary ~ exch ~ put }
1850         ifelse ~
1851         moveto ~
1852         currentpoint ~ pop ~
1853         SDict /pdf.leftboundary ~ known ~ dup
1854         {
1855             SDict /pdf.leftboundary ~ get ~ 2 ~ index ~ gt
1856             { not }
1857             if
1858         }
1859         if
1860         { pop }
1861         { SDict ~ exch /pdf.leftboundary ~ exch ~ put }
1862         ifelse
1863     }
1864     put
1865 }
1866 if
1867 }
1868 def
1869 /pdf.bordertracking.end
1870 {
1871     /a ~ where { /a { moveto } put } if
1872     /x ~ where { /x { 0 ~ exch ~ rmoveto } put } if ~
1873     SDict /pdf.leftboundary ~ known
1874     { pdf.outerbox ~ 0 ~ pdf.leftboundary ~ put }
1875     if ~
1876     SDict /pdf.rightboundary ~ known
1877     { pdf.outerbox ~ 2 ~ pdf.rightboundary ~ put }
1878     if ~
1879     SDict /pdf.bordertracking ~ false ~ put
1880 }
1881 def
1882 /pdf.bordertracking.endpage
1883 {
1884     pdf.bordertracking
1885     {
1886         pdf.bordertracking.end ~
1887         true ~ setglobal ~
1888         pdf.globaldict
1889         /pdf.brokenlink.rect [ pdf.outerbox ~ aload ~ pop ] put ~
1890         pdf.globaldict
1891         /pdf.brokenlink.skip ~ pdf.baselineskip ~ put ~

```



```

1892 pdf.globaldict
1893 /pdf.brokenlink.dict ~
1894 pdf.link.dict ~ pdf.cvs ~ put ~
1895 false ~ setglobal ~
1896 mark ~ pdf.link.dict ~ cvx ~ exec ~ /Rect
1897 [
1898 pdf.llx ~
1899 pdf.lly ~
1900 pdf.outerbox ~ 2 ~ get ~ pdf.linkmargin ~ add ~
1901 currentpoint ~ exch ~ pop ~
1902 pdf.outerbox ~ pdf.rect.ht ~ sub ~ pdf.linkmargin ~ sub
1903 ]
1904 /ANN ~ pdf.pdfmark
1905 }
1906 if
1907 }
1908 def
1909 /pdf.bordertracking.continue
1910 {
1911 /pdf.link.dict ~ pdf.globaldict
1912 /pdf.brokenlink.dict ~ get ~ def
1913 /pdf.outerbox ~ pdf.globaldict
1914 /pdf.brokenlink.rect ~ get ~ def
1915 /pdf.baselineskip ~ pdf.globaldict
1916 /pdf.brokenlink.skip ~ get ~ def ~
1917 pdf.globaldict ~ dup ~ dup
1918 /pdf.brokenlink.dict ~ undef
1919 /pdf.brokenlink.skip ~ undef
1920 /pdf.brokenlink.rect ~ undef ~
1921 currentpoint
1922 /pdf.originy ~ exch ~ def
1923 /pdf.originx ~ exch ~ def
1924 /a ~ where
1925 {
1926 /a
1927 {
1928 moveto ~
1929 SDict ~
1930 begin ~
1931 currentpoint ~ pdf.originy ~ ne ~ exch ~
1932 pdf.originx ~ ne ~ or
1933 {
1934 pdf.save.linkll
1935 /pdf.lly ~
1936 pdf.lly ~ pdf.outerbox ~ 1 ~ get ~ sub ~ def ~
1937 pdf.bordertracking.begin
1938 }
1939 if ~
1940 end
1941 }
1942 put
1943 }
1944 if
1945 /x ~ where

```

```

1946     {
1947         /x
1948         {
1949             0 ~ exch ~ rmoveto ~
1950             SDict~
1951             begin ~
1952             currentpoint ~
1953             pdf.originy ~ ne ~ exch ~ pdf.originx ~ ne ~ or
1954             {
1955                 pdf.save.linkll
1956                 /pdf.lly ~
1957                 pdf.lly ~ pdf.outerbox ~ 1 ~ get ~ sub ~ def ~
1958                 pdf.bordertracking.begin
1959             }
1960             if ~
1961             end
1962         }
1963         put
1964     }
1965     if
1966 }
1967     def
1968 }

```

(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.

```

1969 \_kernel_backend_postscript_header:n
1970 {
1971     /pdf.breaklink
1972     {
1973         pop ~
1974         counttomark ~ 2 ~ mod ~ 0 ~ eq
1975         {
1976             counttomark /pdf.count ~ exch ~ def
1977             {
1978                 pdf.count ~ 0 ~ eq { exit } if ~
1979                 counttomark ~ 2 ~ roll ~
1980                 1 ~ index ~ /Rect ~ eq
1981                 {
1982                     dup ~ 4 ~ array ~ copy ~
1983                     dup ~ dup ~
1984                     1 ~ get ~
1985                     pdf.outerbox ~ pdf.rect.ht ~
1986                     pdf.linkmargin ~ 2 ~ mul ~ add ~ sub ~
1987                     3 ~ exch ~ put ~
1988                     dup ~
1989                     pdf.outerbox ~ 2 ~ get ~
1990                     pdf.linkmargin ~ add ~
1991                     2 ~ exch ~ put ~

```

```

1992     dup ~ dup ~
1993     3 ~ get ~
1994     pdf.outerbox ~ pdf.rect.ht ~
1995     pdf.linkmargin ~ 2 ~ mul ~ add ~ add ~
1996     1 ~ exch ~ put
1997     /pdf.currentrect ~ exch ~ def ~
1998     pdf.breaklink.write
1999     {
2000         pdf.currentrect ~
2001         dup ~
2002         pdf.outerbox ~ 0 ~ get ~
2003         pdf.linkmargin ~ sub ~
2004         0 ~ exch ~ put ~
2005         dup ~
2006         pdf.outerbox ~ 2 ~ get ~
2007         pdf.linkmargin ~ add ~
2008         2 ~ exch ~ put ~
2009         dup ~ dup ~
2010         1 ~ get ~
2011         pdf.baselineskip ~ add ~
2012         1 ~ exch ~ put ~
2013         dup ~ dup ~
2014         3 ~ get ~
2015         pdf.baselineskip ~ add ~
2016         3 ~ exch ~ put ~
2017         /pdf.currentrect ~ exch ~ def ~
2018         pdf.breaklink.write
2019     }
2020     1 ~ index ~ 3 ~ get ~
2021     pdf.linkmargin ~ 2 ~ mul ~ add ~
2022     pdf.outerbox ~ pdf.rect.ht ~ add ~
2023     2 ~ index ~ 1 ~ get ~ sub ~
2024     pdf.baselineskip ~ div ~ round ~ cvi ~ 1 ~ sub ~
2025     exch ~
2026     repeat ~
2027     pdf.currentrect ~
2028     dup ~
2029     pdf.outerbox ~ 0 ~ get ~
2030     pdf.linkmargin ~ sub ~
2031     0 ~ 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     dup ~ 2 ~ index ~ 2 ~ get ~ 2 ~ exch ~ put
2041     /pdf.currentrect ~ exch ~ def ~
2042     pdf.breaklink.write ~
2043     SDict /pdf.pdfmark.good ~ false ~ put ~
2044     exit
2045 }

```

```

2046         { pdf.count ~ 2 ~ sub /pdf.count ~ exch ~ def }
2047     ifelse
2048 }
2049 loop
2050 }
2051 if
2052 /ANN
2053 }
2054 def
2055 /pdf.breaklink.write
2056 {
2057     counttomark ~ 1 ~ sub ~
2058     index /_objdef ~ eq
2059     {
2060         counttomark ~ -2 ~ roll ~
2061         dup ~ wcheck ~
2062         {
2063             readonly ~
2064             counttomark ~ 2 ~ roll
2065         }
2066         { pop ~ pop }
2067     ifelse
2068 }
2069 if ~
2070 counttomark ~ 1 ~ add ~ copy ~
2071 pop ~ pdf.currentrect
2072 /ANN ~ pdfmark
2073 }
2074 def
2075 }

```

(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.

```

2076 \_kernel_backend_postscript_header:n
2077 {
2078     /pdf.pdfmark
2079     {
2080         SDict /pdf.pdfmark.good ~ true ~ put ~
2081         dup /ANN ~ eq
2082         {
2083             pdf.pdfmark.store ~
2084             pdf.pdfmark.dict ~
2085             begin ~
2086             Subtype /Link ~ eq ~
2087             currentdict /Rect ~ known ~ and ~
2088             SDict /pdf.outerbox ~ known ~ and ~
2089             SDict /pdf.baselineskip ~ known ~ and ~
2090             {
2091                 Rect ~ 3 ~ get ~

```

```

2092         pdf.linkmargin ~ 2 ~ mul ~ add ~
2093         pdf.outerbox ~ pdf.rect.ht ~ add ~
2094         Rect ~ 1 ~ get ~ sub ~
2095         pdf.baselineskip ~ div ~ round ~ cvi ~ 0 ~ gt
2096         { pdf.breaklink }
2097     if
2098     }
2099     if ~
2100     end ~
2101     SDict /pdf.outerbox ~ undef ~
2102     SDict /pdf.baselineskip ~ undef ~
2103     currentdict /pdf.pdfmark.dict ~ undef ~
2104 }
2105 if ~
2106 pdf.pdfmark.good
2107 { pdfmark }
2108 { cleartomark }
2109 ifelse
2110 }
2111 def
2112 /pdf.pdfmark.store
2113 {
2114     /pdf.pdfmark.dict ~ 65534 ~ dict ~ def ~
2115     counttomark ~ 1 ~ add ~ copy ~
2116     pop
2117     {
2118         dup ~ mark ~ eq
2119         {
2120             pop ~
2121             exit
2122         }
2123         {
2124             pdf.pdfmark.dict ~
2125             begin ~ def ~ end
2126         }
2127         ifelse
2128     }
2129     loop
2130 }
2131 def
2132 }

```

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

\l__pdf_backend_content_box The content of an annotation.

```

2133 \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.

```

2134 \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.
2135 \int_new:N \g__pdf_backend_annotation_int
(End definition for \g__pdf_backend_annotation_int.)

\_pdf_backend_annotation:nnnn Annotations are objects, but we track them separately. Notably, they are not in the
\_pdf_backend_annotation_aux:nnnn 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 2ε 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.

2136 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2137 {
2138   \__pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4}
2139   \int_gincr:N \g__pdf_backend_object_int
2140   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2141   \__pdf_backend_pdfmark:x
2142   {
2143
2144     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2145     pdf.rect ~
2146     #4 ~
2147     /ANN
2148   }
2149 }
2150 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2151 {
2152   \box_move_down:nn {#3}
2153   { \hbox:n { \__pdf_postscript:n { pdf.save.ll } } }
2154   \hbox:n {#4}
2155   \box_move_up:nn {#2}
2156   {
2157     \hbox:n
2158     {
2159       \tex_kern:D \dim_eval:n {#1} \scan_stop:
2160       \__pdf_postscript:n { pdf.save.ur }
2161     }
2162   }
2163   \int_gincr:N \g__pdf_backend_object_int
2164   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2165   \__pdf_backend_pdfmark:x
2166   {
2167     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2168     pdf.rect
2169     /ANN
2170   }
2171 }

(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.

2172 \cs_new:Npn \__pdf_backend_annotation_last:
2173 { { 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.

2174 `\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.

2175 `\tl_new:N \g_pdf_backend_link_dict_tl`

(End definition for `\g_pdf_backend_link_dict_tl:`)

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

2176 `\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.

2177 `\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.

2178 `\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.

2179 `\tl_new:N \l_pdf_breaklink_pdfmark_tl`

2180 `\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.

2181 `\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.

2182 `\cs_new_eq:NN _pdf_breaklink_usebox:N \box_use:N`

(End definition for `_pdf_breaklink_usebox:N:`)

`_pdf_backend_link_begin_goto:nw`

`_pdf_backend_link_begin_user:nw`

`_pdf_backend_link:nw`

`_pdf_backend_link_aux:nw`

`_pdf_backend_link_end:`

`_pdf_backend_link_end_aux:`

`_pdf_backend_link_minima:`

`_pdf_backend_link_outerbox:n`

`_pdf_backend_link_sf_save:`

`_pdf_backend_link_sf_restore:`

`pdf.linkdp.pad`

`pdf.linkht.pad`

`pdf.llx`

`pdf.lly`

`pdf.ury`

`pdf.link.dict`

`pdf.outerbox`

`pdf.baselineskip`

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

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

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

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

```

2183 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2184 { \__pdf_backend_link_begin:nw { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2185 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
2186 { \__pdf_backend_link_begin:nw {#1#2} }
2187 \cs_new_protected:Npn \__pdf_backend_link_begin:nw #1
2188 {
2189   \bool_if:NF \g__pdf_backend_link_bool
2190     { \__pdf_backend_link_begin_aux:nw {#1} }
2191 }
2192 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2193 {
2194   \bool_gset_true:N \g__pdf_backend_link_bool
2195   \__pdf_postscript:n
2196     { /pdf.link.dict ( #1 ) def }
2197   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2198   \__pdf_backend_link_sf_save:
2199   \mode_if_math:TF
2200     { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2201     { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2202   \hbox_set:Nw \l__pdf_backend_content_box
2203   \__pdf_backend_link_sf_restore:
2204   \bool_if:NT \g__pdf_backend_link_math_bool
2205     { \c_math_toggle_token }
2206 }
2207 \cs_new_protected:Npn \__pdf_backend_link_end:
2208 {
2209   \bool_if:NT \g__pdf_backend_link_bool
2210     { \__pdf_backend_link_end_aux: }
2211 }
2212 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2213 {
2214   \bool_if:NT \g__pdf_backend_link_math_bool
2215     { \c_math_toggle_token }
2216   \__pdf_backend_link_sf_save:
2217   \hbox_set_end:
2218   \__pdf_backend_link_minima:
2219   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2220   \exp_args:Nx \__pdf_backend_link_outerbox:n
2221   {
2222     \*initex
2223     \l_galley_total_left_margin_dim
2224     \*initex
2225     \*package
2226     \int_if_odd:nTF { \value { page } }
2227       { \oddsidemargin }
2228       { \evensidemargin }
2229     \*package
2230   }
2231   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2232   { \hbox:n { \__pdf_postscript:n { pdf.save.linkll } } }
2233   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2234   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box

```



```

2235 \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2236 \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2237 {
2238   \hbox:n
2239   { \__pdf_postscript:n { pdf.save.linkur } }
2240 }
2241 \int_gincr:N \g__pdf_backend_object_int
2242 \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2243 \__pdf_postscript:x
2244 {
2245   mark
2246   /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2247   \g__pdf_backend_link_dict_tl \c_space_tl
2248   pdf.rect
2249   /ANN ~ \l__pdf_breaklink_pdfmark_tl
2250 }
2251 \__pdf_backend_link_sf_restore:
2252 \bool_gset_false:N \g__pdf_backend_link_bool
2253 }
2254 \cs_new_protected:Npn \__pdf_backend_link_minima:
2255 {
2256   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2257   \__pdf_postscript:x
2258   {
2259     /pdf.linkdp.pad ~
2260     \dim_to_decimal:n
2261     {
2262       \dim_max:nn
2263       {
2264         \box_dp:N \l__pdf_backend_model_box
2265         - \box_dp:N \l__pdf_backend_content_box
2266       }
2267       { Opt }
2268     } ~
2269     pdf.pt.dvi ~ def
2270     /pdf.linkht.pad ~
2271     \dim_to_decimal:n
2272     {
2273       \dim_max:nn
2274       {
2275         \box_ht:N \l__pdf_backend_model_box
2276         - \box_ht:N \l__pdf_backend_content_box
2277       }
2278       { Opt }
2279     } ~
2280     pdf.pt.dvi ~ def
2281   }
2282 }
2283 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2284 {
2285   \__pdf_postscript:x
2286   {
2287     /pdf.outerbox
2288     [

```

```

2289         \dim_to_decimal:n {#1} ~
2290         \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2291     \<initex>
2292         \dim_to_decimal:n { #1 + \l_galley_text_width_dim } ~
2293     \</initex>
2294     \<package>
2295         \dim_to_decimal:n { #1 + \textwidth } ~
2296     \</package>
2297         \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2298     ]
2299     [ exch { pdf.pt.dvi } forall ] def
2300 /pdf.baselineskip ~
2301     \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2302     { pdf.pt.dvi ~ def }
2303     { pop ~ pop }
2304     ifelse
2305 }
2306 }
2307 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2308 {
2309     \int_gset:Nn \g__pdf_backend_link_sf_int
2310     {
2311         \mode_if_horizontal:TF
2312         { \tex_spacefactor:D }
2313         { 0 }
2314     }
2315 }
2316 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2317 {
2318     \mode_if_horizontal:T
2319     {
2320         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2321         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2322     }
2323 }

```

(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.

```

2324 \<package>
2325 \use_none:n
2326 {
2327     \cs_if_exist:NT \@makecol@hook
2328     {
2329         \tl_put_right:Nn \@makecol@hook
2330         {
2331             \box_if_empty:NF \@cclv
2332             {
2333                 \vbox_set:Nn \@cclv
2334                 {
2335                     \__pdf_postscript:n

```

```

2336         {
2337             pdf.globaldict /pdf.brokenlink.rect ~ known
2338             { pdf.bordertracking.continue }
2339             if
2340         }
2341         \vbox_unpack_drop:N \@cclv
2342         \__pdf_postscript:n
2343         { pdf.bordertracking.endpage }
2344     }
2345 }
2346 }
2347 \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2348 \cs_set_eq:NN \__pdf_breaklink_postscript:n \__pdf_postscript:n
2349 \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2350 }
2351 }
2352 </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.

```

2353 \cs_new:Npn \__pdf_backend_link_last:
2354 { { 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.

```

2355 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2356 {
2357     \__pdf_postscript:x
2358     {
2359         /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2360     }
2361 }

```

(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 anchor point actually is: worked out in PostScript. For the rectangle version, we have a bit more PostScript: we need two points.

```

2362 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2363 {
2364     \__pdf_postscript:n { pdf.dest.anchor }
2365     \__pdf_backend_pdfmark:x
2366     {
2367         /View
2368         [
2369             \str_case:nnF {#2}
2370             {
2371                 { xyz } { /XYZ ~ pdf.dest.point ~ null }
2372                 { fit } { /Fit }
2373                 { fitb } { /FitB }
2374                 { fitbh } { /FitBH ~ pdf.dest.y }
2375                 { fitbv } { /FitBV ~ pdf.dest.x }
2376                 { fith } { /FitH ~ pdf.dest.y }

```

```

2377         { fitv } { /FitV ~ pdf.dest.x }
2378     }
2379     {
2380         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2381     }
2382 ]
2383 /Dest ( \exp_not:n {#1} ) cvn
2384 /DEST
2385 }
2386 }
2387 \cs_new_protected:Npn \__pdf_backend_destination_rectangle:nn #1#2
2388 {
2389     \group_begin:
2390     \hbox_set:Nn \l__pdf_internal_box {#2}
2391     \box_move_down:nn
2392     { \box_dp:N \l__pdf_internal_box }
2393     { \hbox:n { \__pdf_postscript:n { pdf.save.ll } } }
2394     \box_use:N \l__pdf_internal_box
2395     \box_move_up:nn
2396     { \box_ht:N \l__pdf_internal_box }
2397     { \hbox:n { \__pdf_postscript:n { pdf.save.ur } } }
2398     \__pdf_backend_pdfmark:n
2399     {
2400         /View
2401         [
2402             /FitR ~
2403             pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2404             pdf.urx ~ pdf.ury ~ pdf.dest2device
2405         ]
2406         /Dest ( #1 ) cvn
2407         /DEST
2408     }
2409     \group_end:
2410 }

```

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

6.2.4 Structure

__pdf_backend_compresslevel:n
__pdf_backend_compress_objects:n

These are all no-ops.

```

2411 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2412 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1 { }

```

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

__pdf_backend_version_major_gset:n
__pdf_backend_version_minor_gset:n

Data not available!

```

2413 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2414 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1 { }

```

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

__pdf_backend_version_major:
__pdf_backend_version_minor:

Data not available!

```

2415 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2416 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

```

(End definition for __pdf_backend_version_major: and __pdf_backend_version_minor:.)

6.2.5 Marked content

Simple wrappers.

```

\__pdf_backend_bdc:nn
\__pdf_backend_emc:
2417 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2418 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2419 \cs_new_protected:Npn \__pdf_backend_emc:
2420 { \__pdf_backend_pdfmark:n { /EMC } }

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

2421 </dvips>

```

6.3 pdfmode backend

```
2422 <*pdfmode>
```

6.3.1 Annotations

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

```

2423 \cs_new_protected:Npx \__pdf_backend_annotation:nnnn #1#2#3#4
2424 {
2425   \cs_if_exist:NTF \tex_pdfextension:D
2426   { \tex_pdfextension:D annot ~ }
2427   { \tex_pdfannot:D }
2428   width ~ \exp_not:N \dim_eval:n {#1} ~
2429   height ~ \exp_not:N \dim_eval:n {#2} ~
2430   depth ~ \exp_not:N \dim_eval:n {#3} ~
2431   {#4}
2432 }

```

(End definition for __pdf_backend_annotation:nnnn.)

A tiny amount of extra data gets added here.

```

\__pdf_backend_annotation_last:
2433 \cs_new:Npx \__pdf_backend_annotation_last:
2434 {
2435   \exp_not:N \int_value:w
2436   \cs_if_exist:NTF \tex_pdffeedback:D
2437   { \exp_not:N \tex_pdffeedback:D lastannot ~ }
2438   { \exp_not:N \tex_pdflastannot:D }
2439   \c_space_tl 0 ~ R
2440 }

```

(End definition for __pdf_backend_annotation_last:.)

Links are all created using the same internals.

```

\__pdf_backend_link_begin_goto:nnw
\__pdf_backend_link_begin_user:nnw
\__pdf_backend_link_begin:nnnw
\__pdf_backend_link_end:
2441 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2442 { \__pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2443 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
2444 { \__pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2445 \cs_new_protected:Npx \__pdf_backend_link_begin:nnnw #1#2#3
2446 {
2447   \cs_if_exist:NTF \tex_pdfextension:D
2448   { \tex_pdfextension:D startlink ~ }
2449   { \tex_pdfstartlink:D }
2450   attr {#1}
2451   #2 {#3}

```

```

2452 }
2453 \cs_new_protected:Npx \__pdf_backend_link_end:
2454 {
2455   \cs_if_exist:NTF \tex_pdfextension:D
2456   { \tex_pdfextension:D endlink \scan_stop: }
2457   { \tex_pdfendlink:D }
2458 }

```

(End definition for __pdf_backend_link_begin_goto:nw and others.)

__pdf_backend_link_last: Formatted for direct use.

```

2459 \cs_new:Npx \__pdf_backend_link_last:
2460 {
2461   \exp_not:N \int_value:w
2462   \cs_if_exist:NTF \tex_pdffeedback:D
2463   { \exp_not:N \tex_pdffeedback:D lastlink ~ }
2464   { \exp_not:N \tex_pdflastlink:D }
2465   \c_space_tl 0 ~ R
2466 }

```

(End definition for __pdf_backend_link_last:.)

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

```

2467 \cs_new_protected:Npx \__pdf_backend_link_margin:n #1
2468 {
2469   \cs_if_exist:NTF \tex_pdfvariable:D
2470   { \exp_not:N \tex_pdfvariable:D linkmargin }
2471   { \exp_not:N \tex_pdflinkmargin:D }
2472   \exp_not:N \dim_eval:n {#1} \scan_stop:
2473 }

```

(End definition for __pdf_backend_link_margin:n.)

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

```

2474 \cs_new_protected:Npx \__pdf_backend_destination:nn #1#2
2475 {
2476   \cs_if_exist:NTF \tex_pdfextension:D
2477   { \exp_not:N \tex_pdfextension:D dest ~ }
2478   { \exp_not:N \tex_pdfdest:D }
2479   name {#1}
2480   \exp_not:N \str_case:nnF {#2}
2481   {
2482     { xyz } { xyz }
2483     { fit } { fit }
2484     { fitb } { fitb }
2485     { fitbh } { fitbh }
2486     { fitbv } { fitbv }
2487     { fith } { fith }
2488     { fitv } { fitv }
2489   }
2490   { xyz ~ zoom \exp_not:N \fp_eval:n { #2 * 10 } }
2491   \scan_stop:
2492 }

```

```

2493 \cs_new_protected:Npx \__pdf_backend_destination_rectangle:nn #1#2
2494 {
2495   \group_begin:
2496     \hbox_set:Nn \l__pdf_internal_box {#2}
2497     \cs_if_exist:NTF \tex_pdfextension:D
2498     { \exp_not:N \tex_pdfextension:D dest ~ }
2499     { \exp_not:N \tex_pdfdest:D }
2500     name {#1}
2501     fitr ~
2502     width \exp_not:N \box_wd:N \l__pdf_internal_box
2503     height \exp_not:N \box_ht:N \l__pdf_internal_box
2504     depth \exp_not:N \box_dp:N \l__pdf_internal_box
2505     \box_use:N \l__pdf_internal_box
2506   \group_end:
2507 }

```

(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
2508 \cs_new_protected:Npx \__pdf_backend_catalog_gput:nn #1#2
2509 {
2510   \cs_if_exist:NTF \tex_pdfextension:D
2511   { \tex_pdfextension:D catalog }
2512   { \tex_pdfcatalog:D }
2513   { / #1 ~ #2 }
2514 }
2515 \cs_new_protected:Npx \__pdf_backend_info_gput:nn #1#2
2516 {
2517   \cs_if_exist:NTF \tex_pdfextension:D
2518   { \tex_pdfextension:D info }
2519   { \tex_pdfinfo:D }
2520   { / #1 ~ #2 }
2521 }

```

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

6.3.3 Objects

\g__pdf_backend_object_prop For tracking objects to allow finalisation.

```

2522 \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
2523 \group_begin:
2524   \cs_set_protected:Npn \__pdf_tmp:w #1#2
2525   {
2526     \cs_new_protected:Npx \__pdf_backend_object_new:nn ##1##2
2527     {
2528       #1 reserveobjnum ~
2529       \int_const:cn
2530       { c__pdf_backend_object_ \exp_not:N \tl_to_str:n {##1} _int }

```

```

2531         {#2}
2532         \prop_gput:Nnn \exp_not:N \g__pdf_backend_object_prop {##1} {##2}
2533     }
2534 }
2535 \cs_if_exist:NTF \tex_pdfextension:D
2536 {
2537     \__pdf_tmp:w
2538     { \tex_pdfextension:D obj ~ }
2539     { \exp_not:N \tex_pdffeedback:D lastobj }
2540 }
2541 { \__pdf_tmp:w { \tex_pdfobj:D } { \tex_pdflastobj:D } }
2542 \group_end:
2543 \cs_new:Npn \__pdf_backend_object_ref:n #1
2544 { \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

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

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

```

\_pdf_backend_object_write:nn 2545 \group_begin:
\_pdf_backend_object_write:nx 2546 \cs_set_protected:Npn \__pdf_tmp:w #1
\_pdf_exp_not_i:nn 2547 {
\_pdf_exp_not_ii:nn 2548     \cs_new_protected:Npn \__pdf_backend_object_write:nn ##1##2
2549     {
2550         \tex_immediate:D #1 useobjnum ~
2551         \int_use:c
2552         { c__pdf_backend_object_ \tl_to_str:n {##1} _int }
2553         \str_case_e:nn
2554         { \prop_item:Nn \g__pdf_backend_object_prop {##1} }
2555         {
2556             { array } { { [ ~ \exp_not:n {##2} ~ ] } }
2557             { dict } { { << ~ \exp_not:n {##2} ~ >> } }
2558             { fstream }
2559             {
2560                 stream ~ attr ~ { \__pdf_exp_not_i:nn ##2 } ~
2561                 file ~ { \__pdf_exp_not_ii:nn ##2 }
2562             }
2563             { stream }
2564             {
2565                 stream ~ attr ~ { \__pdf_exp_not_i:nn ##2 } ~
2566                 { \__pdf_exp_not_ii:nn ##2 }
2567             }
2568         }
2569     }
2570 }
2571 \cs_if_exist:NTF \tex_pdfextension:D
2572 { \__pdf_tmp:w { \tex_pdfextension:D obj ~ } }
2573 { \__pdf_tmp:w { \tex_pdfobj:D } }
2574 \group_end:
2575 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2576 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2577 \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`
`_pdf_backend_object_now:nx`

Much like writing, but direct creation.

```

2578 \group_begin:
2579   \cs_set_protected:Npn \_pdf_tmp:w #1
2580   {
2581     \cs_new_protected:Npn \_pdf_backend_object_now:nn ##1##2
2582     {
2583       \tex_immediate:D #1
2584       \str_case:nn
2585         {##1}
2586         {
2587           { array } { { [ ~ \exp_not:n {##2} ~ ] } }
2588           { dict } { { << ~ \exp_not:n {##2} ~ >> } }
2589           { fstream }
2590           {
2591             stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2592             file ~ { \_pdf_exp_not_ii:nn ##2 }
2593           }
2594           { stream }
2595           {
2596             stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2597             { \_pdf_exp_not_ii:nn ##2 }
2598           }
2599         }
2600     }
2601   }
2602   \cs_if_exist:NTF \tex_pdfextension:D
2603   { \_pdf_tmp:w { \tex_pdfextension:D obj ~ } }
2604   { \_pdf_tmp:w { \tex_pdfobj:D } }
2605 \group_end:
2606 \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.

```

2607 \cs_new:Npx \_pdf_backend_object_last:
2608 {
2609   \exp_not:N \int_value:w
2610   \cs_if_exist:NTF \tex_pdffeedback:D
2611   { \exp_not:N \tex_pdffeedback:D lastobj ~ }
2612   { \exp_not:N \tex_pdflastobj:D }
2613   \c_space_tl 0 ~ R
2614 }

```

(End definition for `_pdf_backend_object_last:.`)

6.3.4 Structure

`_pdf_backend_compresslevel:n`
`_pdf_backend_compress_objects:n`
`_pdf_backend_objcompresslevel:n`

Simply pass data to the engine.

```

2615 \cs_new_protected:Npx \_pdf_backend_compresslevel:n #1
2616 {
2617   \exp_not:N \tex_global:D
2618   \cs_if_exist:NTF \tex_pdfcompresslevel:D
2619   { \tex_pdfcompresslevel:D }
2620   { \tex_pdfvariable:D compresslevel }

```

```

2621     \exp_not:N \int_value:w \exp_not:N \int_eval:n {#1} \scan_stop:
2622   }
2623   \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2624   {
2625     \bool_if:NTF {#1}
2626     { \__pdf_backend_objcompresslevel:n { 2 } }
2627     { \__pdf_backend_objcompresslevel:n { 0 } }
2628   }
2629   \cs_new_protected:Npx \__pdf_backend_objcompresslevel:n #1
2630   {
2631     \exp_not:N \tex_global:D
2632     \cs_if_exist:NTF \tex_pdfobjcompresslevel:D
2633     { \tex_pdfobjcompresslevel:D }
2634     { \tex_pdfvariable:D objcompresslevel }
2635     #1 \scan_stop:
2636   }

```

(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
 __pdf_backend_version_minor_gset:n one ...

```

2637   \cs_new_protected:Npx \__pdf_backend_version_major_gset:n #1
2638   {
2639     \cs_if_exist:NTF \tex_pdfvariable:D
2640     {
2641       \int_compare:nNnT \tex_luatexversion:D > { 106 }
2642       {
2643         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2644         \exp_not:N \int_eval:n {#1} \scan_stop:
2645       }
2646     }
2647     {
2648       \cs_if_exist:NT \tex_pdfmajorversion:D
2649       {
2650         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2651         \exp_not:N \int_eval:n {#1} \scan_stop:
2652       }
2653     }
2654   }
2655   \cs_new_protected:Npx \__pdf_backend_version_minor_gset:n #1
2656   {
2657     \exp_not:N \tex_global:D
2658     \cs_if_exist:NTF \tex_pdfminorversion:D
2659     { \exp_not:N \tex_pdfminorversion:D }
2660     { \tex_pdfvariable:D minorversion }
2661     \exp_not:N \int_eval:n {#1} \scan_stop:
2662   }

```

(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: 2663   \cs_new:Npx \__pdf_backend_version_major:
2664   {
2665     \cs_if_exist:NTF \tex_pdfvariable:D

```

```

2666     {
2667         \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2668         { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2669         { 1 }
2670     }
2671     {
2672         \cs_if_exist:NTF \tex_pdfmajorversion:D
2673         { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2674         { 1 }
2675     }
2676 }
2677 \cs_new:Npx \__pdf_backend_version_minor:
2678 {
2679     \exp_not:N \tex_the:D
2680     \cs_if_exist:NTF \tex_pdfminorversion:D
2681     { \exp_not:N \tex_pdfminorversion:D }
2682     { \tex_pdfvariable:D minorversion }
2683 }

```

(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>.

```

2684 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2685     { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2686 \cs_new_protected:Npn \__pdf_backend_emc:
2687     { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2688 </pdfmode>

```

6.4 dvipdfmx backend

```

2689 <*dvipdfmx | xdvipdfmx>

```

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

```

2690 \__pdf_backend:x
2691 \cs_new_protected:Npx \__pdf_backend:n #1
2692     { \__kernel_backend_literal:n { pdf: #1 } }
2693 \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

```

```

2693 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2694     { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2695 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2696     { \__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` 2697 `\int_new:N \g__pdf_backend_object_int`
2698 `\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` 2699 `\cs_new_protected:Npn __pdf_backend_object_new:nn #1#2`
2700 `{`
2701 `\int_gincr:N \g__pdf_backend_object_int`
2702 `\int_const:cn`
2703 `{ c__pdf_backend_object_ \tl_to_str:n {#1} _int }`
2704 `{ \g__pdf_backend_object_int }`
2705 `\prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}`
2706 `}`
2707 `\cs_new:Npn __pdf_backend_object_ref:n #1`
2708 `{ @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` 2709 `\cs_new_protected:Npn __pdf_backend_object_write:nn #1#2`
2710 `{`
2711 `\exp_args:Nx __pdf_backend_object_write:nnn`
2712 `{ \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}`
2713 `}`
2714 `\cs_generate_variant:Nn __pdf_backend_object_write:nn { nx }`
2715 `\cs_new_protected:Npn __pdf_backend_object_write:nnn #1#2#3`
2716 `{`
2717 `\use:c { __pdf_backend_object_write_ #1 :nn }`
2718 `{ __pdf_backend_object_ref:n {#2} } {#3}`
2719 `}`
2720 `\cs_new_protected:Npn __pdf_backend_object_write_array:nn #1#2`
2721 `{`
2722 `__pdf_backend:x`
2723 `{ obj ~ #1 ~ [~ \exp_not:n {#2} ~] }`
2724 `}`
2725 `\cs_new_protected:Npn __pdf_backend_object_write_dict:nn #1#2`
2726 `{`
2727 `__pdf_backend:x`
2728 `{ obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }`
2729 `}`
2730 `\cs_new_protected:Npn __pdf_backend_object_write_fstream:nn #1#2`
2731 `{ __pdf_backend_object_write_stream:nnnn { f } {#1} #2 }`
2732 `\cs_new_protected:Npn __pdf_backend_object_write_stream:nn #1#2`
2733 `{ __pdf_backend_object_write_stream:nnnn { } {#1} #2 }`
2734 `\cs_new_protected:Npn __pdf_backend_object_write_stream:nnnn #1#2#3#4`
2735 `{`
2736 `__pdf_backend:x`
2737 `{`
2738 `#1 stream ~ #2 ~`
2739 `(\exp_not:n {#4}) ~ << \exp_not:n {#3} >>`
2740 `}`

```
2741 }
```

(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 2742 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2743 {
2744   \int_gincr:N \g__pdf_backend_object_int
2745   \exp_args:Nnx \use:c { \_pdf_backend_object_write_ #1 :nn }
2746   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2747   {#2}
2748 }
2749 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }
```

(End definition for `_pdf_backend_object_now:nn`.)

`_pdf_backend_object_last:`

```
2750 \cs_new:Npn \_pdf_backend_object_last:
2751 { @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.

```
2752 \bool_new:N \g__pdf_landscape_bool
2753 <*package>
2754 \AtBeginDocument
2755 {
2756   \cs_if_exist:NT \landscape
2757   {
2758     \tl_put_right:Nn \landscape
2759     { \bool_gset_true:N \g__pdf_landscape_bool }
2760     \tl_put_left:Nn \endlandscape
2761     { \bool_gset_false:N \g__pdf_landscape_bool }
2762   }
2763 }
2764 </package>
```

(End definition for `\g__pdf_landscape_bool`.)

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

```
2765 \int_new:N \g__pdf_backend_annotation_int
```

(End definition for `\g__pdf_backend_annotation_int`.)

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

```
\_pdf_backend_annotation_aux:nmnn 2766 \cs_new_protected:Npn \_pdf_backend_annotation:nmnn #1#2#3#4
2767 {
2768   \bool_if:NTF \g__pdf_landscape_bool
2769   {
2770     \box_move_up:nn {#2}
2771     {
```

```

2772         \vbox:n
2773         {
2774             \_pdf_backend_annotation_aux:nnnn
2775             { #2 + #3 } {#1} { Opt } {#4}
2776         }
2777     }
2778 }
2779 { \_pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4} }
2780 }
2781 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2782 {
2783     \int_gincr:N \g__pdf_backend_object_int
2784     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2785     \_pdf_backend:x
2786     {
2787         ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
2788         width ~ \dim_eval:n {#1} ~
2789         height ~ \dim_eval:n {#2} ~
2790         depth ~ \dim_eval:n {#3} ~
2791         << #4 >>
2792     }
2793 }

```

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

_pdf_backend_annotation_last:

```

2794 \cs_new:Npn \_pdf_backend_annotation_last:
2795 { @pdf.obj \int_use:N \g__pdf_backend_annotation_int }

```

(End definition for _pdf_backend_annotation_last:.)

_pdf_backend_link_begin_goto:nnw

_pdf_backend_link_begin_user:nnw

_pdf_backend_link_begin:n

_pdf_backend_link_end:

All created using the same internals.

```

2796 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2797 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2798 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2799 { \_pdf_backend_link_begin:n {#1#2} }
2800 \cs_new_protected:Npn \_pdf_backend_link_begin:n #1
2801 {
2802     \_pdf_backend:n
2803     {
2804         bann
2805         <<
2806         /Type /Annot
2807         #1
2808         >>
2809     }
2810 }
2811 \cs_new_protected:Npn \_pdf_backend_link_end:
2812 { \_pdf_backend:n { eann } }

```

(End definition for _pdf_backend_link_begin_goto:nnw and others.)

_pdf_backend_link_last: Data not available.

```

2813 \cs_new:Npn \_pdf_backend_link_last: { }

```

(End definition for `_pdf_backend_link_last:`.)

`_pdf_backend_link_margin:n` Pass to dvipdfmx.

```
2814 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2815 { \_kernel_backend_literal:x { dvipdfmx:config~g~ \dim_eval:n {#1} } }
```

(End definition for `_pdf_backend_link_margin:n`.)

`_pdf_backend_destination:nn`
`_pdf_backend_destination_rectangle:nn`

Here, we need to turn the zoom into a scale. The method for `FitR` is from Alexander Grahn: the idea is to avoid needing to do any calculations in \TeX by using the backend data for `@xpos` and `@ypos`.

```
2816 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2817 {
2818   \_pdf_backend:x
2819   {
2820     dest ~ ( \exp_not:n {#1} )
2821     [
2822       @thispage
2823       \str_case:nnF {#2}
2824       {
2825         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2826         { fit } { /Fit }
2827         { fitb } { /FitB }
2828         { fitbh } { /FitBH }
2829         { fitbv } { /FitBV ~ @xpos }
2830         { fith } { /FitH ~ @ypos }
2831         { fitv } { /FitV ~ @xpos }
2832       }
2833       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2834     ]
2835   }
2836 }
2837 \cs_new_protected:Npn \_pdf_backend_destination_rectangle:nn #1#2
2838 {
2839   \group_begin:
2840   \hbox_set:Nn \l__pdf_internal_box {#2}
2841   \box_move_down:nn { \box_dp:N \l__pdf_internal_box }
2842   {
2843     \hbox:n
2844     {
2845       \_pdf_backend:n { obj ~ @pdf_ #1 _llx ~ @xpos }
2846       \_pdf_backend:n { obj ~ @pdf_ #1 _lly ~ @ypos }
2847     }
2848   }
2849   \box_use:N \l__pdf_internal_box
2850   \box_move_up:nn { \box_ht:N \l__pdf_internal_box }
2851   {
2852     \hbox:n
2853     {
2854       \_pdf_backend:n
2855       {
2856         dest ~ (#1)
2857         [
2858           @thispage
```

```

2859             /FitR ~
2860             @pdf_ #1 _llx ~ @pdf_ #1 _lly ~
2861             @xpos ~ @ypos
2862         ]
2863     }
2864 }
2865 }
2866 \group_end:
2867 }

```

(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

```

2868 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2869 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2870 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2871 {
2872     \bool_if:nF {#1}
2873     { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2874 }

```

(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
2875 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2876 {
2877     \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2878     \__kernel_backend_literal:x { pdf:majorversion~ \_pdf_backend_version_major: }
2879 }
2880 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2881 {
2882     \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2883     \__kernel_backend_literal:x { pdf:minorversion~ \_pdf_backend_version_minor: }
2884 }

```

(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:
2885 \cs_new:Npn \_pdf_backend_version_major: { 1 }
2886 \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:

```

2887 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2888 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2889 \cs_new_protected:Npn \_pdf_backend_emc:
2890 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2891 </dvipdfmx | xdvipdfmx>

```


6.5 dvisvgm backend

2892 $\langle *dvisvgm \rangle$

6.5.1 Catalogue entries

No-op.

2893 $\backslash cs_new_protected:Npn __pdf_backend_catalog_gput:nn \#1\#2 \{ \}$
2894 $\backslash cs_new_protected:Npn __pdf_backend_info_gput:nn \#1\#2 \{ \}$

(End definition for $\backslash_pdf_backend_catalog_gput:nn$ and $\backslash_pdf_backend_info_gput:nn$.)

6.5.2 Objects

All no-ops here.

2895 $\backslash cs_new_protected:Npn __pdf_backend_object_new:nn \#1\#2 \{ \}$
2896 $\backslash cs_new:Npn __pdf_backend_object_ref:n \#1 \{ \}$
2897 $\backslash cs_new_protected:Npn __pdf_backend_object_write:nn \#1\#2 \{ \}$
2898 $\backslash cs_new_protected:Npn __pdf_backend_object_write:nx \#1\#2 \{ \}$
2899 $\backslash cs_new_protected:Npn __pdf_backend_object_now:nn \#1\#2 \{ \}$
2900 $\backslash cs_new_protected:Npn __pdf_backend_object_now:nx \#1\#2 \{ \}$
2901 $\backslash cs_new:Npn __pdf_backend_object_last: \{ \}$

(End definition for $\backslash_pdf_backend_object_new:nn$ and others.)

6.5.3 Structure

These are all no-ops.

2902 $\backslash cs_new_protected:Npn __pdf_backend_compresslevel:n \#1 \{ \}$
2903 $\backslash cs_new_protected:Npn __pdf_backend_compress_objects:n \#1 \{ \}$

(End definition for $\backslash_pdf_backend_compresslevel:n$ and $\backslash_pdf_backend_compress_objects:n$.)

Data not available!

2904 $\backslash cs_new_protected:Npn __pdf_backend_version_major_gset:n \#1 \{ \}$
2905 $\backslash cs_new_protected:Npn __pdf_backend_version_minor_gset:n \#1 \{ \}$

(End definition for $\backslash_pdf_backend_version_major_gset:n$ and $\backslash_pdf_backend_version_minor_gset:n$.)

Data not available!

2906 $\backslash cs_new:Npn __pdf_backend_version_major: \{ -1 \}$
2907 $\backslash cs_new:Npn __pdf_backend_version_minor: \{ -1 \}$

(End definition for $\backslash_pdf_backend_version_major:$ and $\backslash_pdf_backend_version_minor:.$)

More no-ops.

2908 $\backslash cs_new_protected:Npn __pdf_backend_bdc:nn \#1\#2 \{ \}$
2909 $\backslash cs_new_protected:Npn __pdf_backend_emc: \{ \}$

(End definition for $\backslash_pdf_backend_bdc:nn$ and $\backslash_pdf_backend_emc:.$)

2910 $\langle /dvisvgm \rangle$

2911 $\langle /initex | package \rangle$

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, 435, 1321, 1456, 1620, 2754
 - \AtBeginDvi 42, 43
- B**
- \begin 1370, 1375
 - bool commands:
 - \bool_gset_false:N 582, 598, 624, 646, 662, 814, 1140, 1176, 2201, 2252, 2761
 - \bool_gset_true:N 580, 649, 812, 1155, 2194, 2200, 2759
 - \bool_if:NTF 589, 593, 611, 615, 619, 632, 637, 641, 653, 657, 825, 830, 835, 1114, 1159, 1346, 1387, 1503, 1545, 2189, 2204, 2209, 2214, 2768
 - \bool_if:nTF 2625, 2872
 - \bool_lazy_or:nnTF 1379, 1538
 - \bool_new:N 583, 650, 815, 1156, 2177, 2178, 2752
 - \bool_set_false:N ... 1356, 1470, 1563
 - box commands:
 - \box_dp:N 140, 142, 190, 192, 247, 249, 296, 298, 300, 302, 2231, 2264, 2265, 2290, 2392, 2504, 2841
 - \box_ht:N 142, 192, 249, 300, 302, 1399, 1600, 2236, 2275, 2276, 2297, 2396, 2503, 2850
 - \box_if_empty:NTF 2331
 - \box_move_down:nn 2152, 2231, 2391, 2841
 - \box_move_up:nn 2155, 2236, 2395, 2770, 2850
 - \box_new:N 1643, 2133, 2134
 - \box_set_dp:Nn 1307
 - \box_set_ht:Nn 1306
 - \box_set_wd:Nn 204, 1305
 - \box_use:N . 147, 165, 179, 195, 222, 236, 252, 268, 280, 331, 348, 367, 765, 1022, 1308, 2182, 2394, 2505, 2849
 - \box_wd:N .. 141, 149, 191, 197, 248, 254, 297, 299, 335, 1398, 1599, 2502
 - 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 670, 845
 - \clist_map_function:nn 1183
 - color internal commands:
 - __color_backend_cmyk:nnnn . 402, 472
 - __color_backend_cmyk_aux:nnnn . 472
 - __color_backend_gray:n 402, 472
 - __color_backend_gray_aux:n ... 472
 - __color_backend_pickup:N .. 375, 433
 - __color_backend_pickup:w 13, 375, 433
 - __color_backend_reset: 402, 472
 - __color_backend_rgb:nnn ... 402, 472
 - __color_backend_rgb_aux:nnn .. 472
 - __color_backend_select:n .. 402, 472
 - __color_backend_spot:nn ... 402, 472
 - color.fc 402, 529
 - cs commands:
 - \cs_generate_variant:Nn 28, 32, 35, 72, 100, 105, 116, 123, 428, 514, 528, 733, 739, 775, 923, 1031, 1062, 1517, 1574, 1590, 1647, 1684, 1729, 2575, 2606, 2692, 2714, 2749
 - \cs_gset:Npx 2877, 2882
 - \cs_if_exist:NTF 42, 67, 75, 83, 89, 95, 508, 517, 906, 914, 2327, 2425, 2436, 2447, 2455, 2462, 2469, 2476, 2497, 2510, 2517, 2535, 2571, 2602, 2610, 2618, 2632, 2639, 2648, 2658, 2665, 2672, 2680, 2756
 - \cs_new:Npn 675, 850, 1187, 1603, 1612, 1637, 1662, 1730, 2172, 2353, 2415, 2416, 2543, 2576, 2577, 2707, 2750, 2794, 2813, 2885, 2886, 2896, 2901, 2906, 2907
 - \cs_new:Npx 2433, 2459, 2607, 2663, 2677
 - \cs_new_eq:NN 25, 527, 774, 780, 781, 921, 1030,

1323, 1352, 1409, 1410, 1458, 1466,
1488, 1559, 1623, 1624, 1636, 2182

D

dim commands:

\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, 402, 411, 413, 418, 420, 429,
434, 444, 472, 483, 488, 490, 492,
502, 504, 529, 535, 540, 542, 544,
552, 560, 569, 579, 581, 584, 586,
600, 605, 626, 648, 651, 664, 677,
682, 684, 686, 688, 690, 692, 694,
696, 705, 714, 716, 718, 723, 728,
734, 740, 752, 776, 778, 782, 787,
792, 802, 811, 813, 816, 818, 820,
822, 827, 832, 837, 839, 852, 857,
859, 861, 863, 865, 867, 869, 871,
880, 889, 891, 893, 898, 924, 939,
964, 976, 988, 1000, 1007, 1032,
1037, 1039, 1047, 1057, 1065, 1070,
1075, 1086, 1096, 1106, 1108, 1110,
1112, 1143, 1145, 1150, 1152, 1154,
1157, 1178, 1189, 1202, 1204, 1206,
1208, 1210, 1212, 1214, 1216, 1218,
1228, 1237, 1245, 1247, 1249, 1259,
1274, 1279, 1294, 1324, 1338, 1353,
1365, 1376, 1404, 1416, 1429, 1439,
1460, 1467, 1475, 1486, 1490, 1493,
1508, 1518, 1553, 1560, 1566, 1572,
1575, 1582, 1591, 1596, 1604, 1626,
1645, 1648, 1650, 1654, 1664, 1685,
1690, 1695, 1700, 1709, 2136, 2150,
2181, 2183, 2185, 2187, 2192, 2207,
2212, 2254, 2283, 2307, 2316, 2355,
2362, 2387, 2411, 2412, 2413, 2414,
2417, 2419, 2441, 2443, 2548, 2581,
2623, 2684, 2686, 2693, 2695, 2699,
2709, 2715, 2720, 2725, 2730, 2732,
2734, 2742, 2766, 2781, 2796, 2798,
2800, 2811, 2814, 2816, 2837, 2868,
2870, 2875, 2880, 2887, 2889, 2893,
2894, 2895, 2897, 2898, 2899, 2900,
2902, 2903, 2904, 2905, 2908, 2909

\cs_new_protected:Npx
..... 36, 65, 73, 81, 87,
93, 506, 515, 904, 912, 2423, 2445,
2453, 2467, 2474, 2493, 2508, 2515,
2526, 2615, 2629, 2637, 2655, 2690

\cs_set_eq:NN 2348, 2349

\cs_set_protected:Npn
..... 381, 439, 2524, 2546, 2579

\dim_eval:n 2159, 2428, 2429,
2430, 2472, 2788, 2789, 2790, 2815
\dim_max:nn 2262, 2273
\dim_set:Nn ... 1398, 1399, 1599, 1600
\dim_to_decimal:n 296, 297, 298, 299,
300, 302, 1068, 1073, 1079, 1080,
1081, 1082, 1091, 1092, 1093, 1184,
1203, 1631, 1632, 2260, 2271, 2289,
2290, 2292, 2295, 2297, 2301, 2359
\dim_to_decimal_in_bp:n
..... 140, 141, 142, 190,
191, 192, 247, 248, 249, 548, 549,
556, 557, 564, 565, 573, 574, 575,
672, 676, 680, 785, 790, 796, 797,
798, 806, 807, 847, 851, 855, 1188,
1329, 1330, 1331, 1332, 1480, 1481,
1482, 1483, 1532, 1533, 1534, 1535

draw internal commands:

__draw_align_currentpoint... .. 21
__draw_backend_add_to_path:n ...
..... 1065, 1111
__draw_backend_begin: 529, 776, 1032
__draw_backend_box_use:Nnnnn ...
..... 16, 752, 1007, 1294
__draw_backend_cap_but:
..... 664, 839, 1178
__draw_backend_cap_rectangle: ..
..... 664, 839, 1178
__draw_backend_cap_round:
..... 664, 839, 1178
__draw_backend_clip: 584, 816, 1110
__draw_backend_closepath:
..... 584, 816, 1110
__draw_backend_closestroke: ...
..... 584, 816, 1110
__draw_backend_cm:nnnn 740,
760, 761, 762, 924, 1011, 1279, 1297
__draw_backend_cm_aux:nnnn ... 924
__draw_backend_cm_decompose:nnnnN
..... 934, 963
__draw_backend_cm_decompose_-
auxi:nnnnN 963
__draw_backend_cm_decompose_-
auxii:nnnnN 963
__draw_backend_cm_decompose_-
auxiii:nnnnN 963
__draw_backend_color_fill:n .. 696
__draw_backend_color_fill:nnn 1218
__draw_backend_color_fill_-
cmyk:nnnn 696, 871, 1218
__draw_backend_color_fill_-
gray:n 696, 871, 1218

_draw_backend_color_fill_- rgb:nnn	696, 871, 1218
_draw_backend_color_gray_aux:n	1241, 1245
_draw_backend_color_reset: . .	871
_draw_backend_color_select:n .	871
_draw_backend_color_stroke:n .	696
_draw_backend_color_stroke_- cmyk:nnnn	696, 871, 1218
_draw_backend_color_stroke_- gray:n	696, 871, 1218
_draw_backend_color_stroke_- rgb:nnn	696, 871, 1218
_draw_backend_curveto:nnnnnn . .	544, 782, 1065
_draw_backend_dash:n	664, 839, 1178
_draw_backend_dash_aux:nn . .	1178
_draw_backend_dash_pattern:nn .	664, 839, 1178
_draw_backend_discardpath: . . .	584, 816, 1110
_draw_backend_end:	529, 776, 1032
_draw_backend_evenodd_rule: . . .	579, 811, 1106
_draw_backend_fill:	584, 816, 1110
_draw_backend_fillstroke:	584, 816, 1110
_draw_backend_join_bevel:	664, 839, 1178
_draw_backend_join_miter:	664, 839, 1178
_draw_backend_join_round:	664, 839, 1178
_draw_backend_lineto:nn	544, 782, 1065
_draw_backend_linewidth:n	664, 839, 1178
_draw_backend_literal:n	527, 532, 533, 537, 541, 543, 546, 554, 562, 571, 585, 588, 591, 597, 607, 608, 609, 614, 617, 623, 628, 629, 630, 635, 636, 639, 645, 655, 661, 666, 679, 683, 685, 687, 689, 691, 693, 695, 742, 754, 755, 756, 757, 758, 759, 763, 764, 766, 767, 768, 769, 770, 774, 784, 789, 794, 804, 817, 819, 821, 824, 829, 834, 838, 841, 854, 858, 860, 862, 864, 866, 868, 870, 1030, 1051, 1059, 1117, 1136, 1162
_draw_backend_miterlimit:n . . .	664, 839, 1178
_draw_backend_moveto:nn	544, 782, 1065
_draw_backend_nonzero_rule: . . .	579, 811, 1106
_draw_backend_path:n	1110
_draw_backend_rectangle:nnnn . .	544, 782, 1065
_draw_backend_scope:n	1035, 1039, 1107, 1109, 1129, 1169, 1191, 1203, 1205, 1207, 1209, 1211, 1213, 1215, 1217, 1261, 1281
_draw_backend_scope_begin: . . .	540, 777, 780, 1034, 1039
_draw_backend_scope_end:	540, 779, 780, 1038, 1039
_draw_backend_select:n	1230, 1248, 1276
_draw_backend_stroke:	584, 816, 1110
\g_draw_clip_path_int	1116, 1119, 1132, 1161, 1164, 1172
_draw_color_reset:	737
\g_draw_draw_clip_bool	584, 1110
\g_draw_draw_eor_bool	579, 593, 611, 619, 632, 641, 657, 811, 825, 830, 835
\g_draw_draw_path_int	1110
\g_draw_draw_path_tl	1065, 1121, 1137, 1139, 1166, 1175
\g_draw_draw_scope_int	1039
\l_draw_draw_scope_int	1039
\g_draw_path_int	1125, 1142
E	
\endlandscape	2760
\evensidemargin	2228
exp commands:	
\exp_after:wN	388, 1610
\exp_args:Nf	669, 844
\exp_args:NNf	152, 200, 257
\exp_args:Nnx	1726, 2745
\exp_args:NV	383
\exp_args:Nx	489, 1422, 1443, 1697, 2220, 2711
\exp_last_unbraced:Nn	392, 441
\exp_not:N	43, 70, 79, 98, 511, 512, 520, 909, 910, 917, 2428, 2429, 2430, 2435, 2437, 2438, 2461, 2463, 2464, 2470, 2471, 2472, 2477, 2478, 2480, 2490, 2498, 2499, 2502, 2503, 2504, 2530, 2532, 2539, 2609, 2611, 2612, 2617, 2621, 2631, 2643, 2644, 2650, 2651, 2657, 2659, 2661, 2668, 2673, 2679, 2681
\exp_not:n	27, 70, 79, 98, 1688, 1693, 2383, 2556, 2557, 2576, 2577, 2587, 2588, 2723, 2728, 2739, 2820

F

file commands:

\file_compare_timestamp:nNnTF . 1431
 \file_parse_full_name:nNNN 1418, 1441

fp commands:

\fp_compare:nNnTF 159, 206, 212, 264, 944, 957, 1002
 \fp_eval:n 152, 161, 174,
 175, 200, 217, 232, 234, 257, 266,
 277, 278, 345, 360, 361, 407, 408,
 412, 416, 477, 478, 479, 480, 489,
 497, 498, 499, 683, 700, 701, 710,
 711, 715, 717, 721, 726, 745, 746,
 858, 875, 876, 884, 885, 890, 892,
 896, 901, 929, 930, 946, 951, 952,
 959, 969, 970, 971, 972, 981, 982,
 983, 984, 993, 994, 995, 996, 1017,
 1018, 1205, 1223, 1224, 1225, 1233,
 1234, 1242, 1248, 1254, 1255, 1256,
 1277, 1287, 1288, 2380, 2490, 2833
 \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, 944, 957

G

galley commands:

\l_galley_text_width_dim 2292
 \l_galley_total_left_margin_dim 2223

graphics commands:

\graphics_bb_restore:nTF . 1367, 1593
 \graphics_bb_save:n 1402, 1601
 \l_graphics_decodearray_tl 1344,
 1345, 1355, 1381, 1385, 1386, 1469,
 1501, 1502, 1540, 1543, 1544, 1562
 \graphics_extract_bb:n
 1464, 1471, 1623, 1624
 \l_graphics_interpolate_bool
 1346, 1356, 1380,
 1387, 1470, 1503, 1539, 1545, 1563
 \l_graphics_llx_dim . 1329, 1480, 1532
 \l_graphics_lly_dim . 1330, 1481, 1533
 \l_graphics_name_tl 1436
 \l_graphics_page_int
 1340, 1360, 1361,
 1391, 1392, 1462, 1499, 1500, 1526,
 1527, 1555, 1568, 1569, 1608, 1609
 \l_graphics_pagebox_tl . . 41, 1341,
 1359, 1393, 1394, 1463, 1497, 1498,
 1528, 1530, 1556, 1577, 1578, 1610
 \graphics_read_bb:n 1323, 1458
 \l_graphics_urx_dim
 1331, 1398, 1482, 1534, 1599

\l_graphics_ury_dim 1332,
 1399, 1483, 1535, 1600, 1631, 1632

graphics internal commands:

\l_graphics_backend_dir_str . 1411
 \l_graphics_backend_ext_str . 1411
 __graphics_backend_getbb_auxi:n
 1338
 __graphics_backend_getbb_-
 auxi:nN 1553
 __graphics_backend_getbb_-
 auxii:n 1338
 __graphics_backend_getbb_-
 auxii:nnN 1553
 __graphics_backend_getbb_-
 auxiii:nnNn 1553
 __graphics_backend_getbb_-
 auxiv:nnNnn 1553
 __graphics_backend_getbb_-
 auxv:nNnn 1553
 __graphics_backend_getbb_-
 auxvi:nNnn 1594, 1596
 __graphics_backend_getbb_eps:n .
 1317, 1411, 1452
 __graphics_backend_getbb_eps:nm
 1411
 __graphics_backend_getbb_eps:nn
 1422, 1429
 __graphics_backend_getbb_jpg:n .
 1338, 1452, 1553, 1616
 __graphics_backend_getbb_-
 pagebox:w 1553, 1610
 __graphics_backend_getbb_pdf:n .
 1338, 1437, 1452, 1553
 __graphics_backend_getbb_png:n .
 1338, 1452, 1553, 1616
 __graphics_backend_include_-
 auxi:nn 1475
 __graphics_backend_include_-
 auxii:nnn 1475
 __graphics_backend_include_-
 auxiii:nnn 1475
 __graphics_backend_include_-
 bitmap_quote:w 1604, 1626
 __graphics_backend_include_-
 eps:n 1324, 1411, 1475
 __graphics_backend_include_-
 jpg:n 1404, 1475, 1626
 __graphics_backend_include_-
 pdf:n 1404, 1443, 1475, 1604
 __graphics_backend_include_pdf_-
 quote:w 1607, 1612
 __graphics_backend_include_-
 png:n 1404, 1475, 1626
 \l_graphics_backend_name_str . 1411

<code>\l__graphics_graphics_attr_tl</code>	<code>\int_value:w</code>
. 1337 , 1342 ,	<code>\int_zero:N</code> 1340 , 1462 , 1555
1349 , 1357 , 1367 , 1400 , 1402 , 1407	
<code>\l__graphics_internal_box</code>	
. 1396 , 1398 , 1399 , 1598 , 1599 , 1600	
<code>\g__graphics_track_int</code>	
. 1474 , 1520 , 1521	
group commands:	
<code>\group_begin:</code> 1044 ,	
2389 , 2495 , 2523 , 2545 , 2578 , 2839	
<code>\group_end:</code> 1052 ,	
2409 , 2506 , 2542 , 2574 , 2605 , 2866	
<code>\group_insert_after:N</code>	
. 426 , 512 , 737 , 910	
	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:e</code> 25
	<code>__kernel_backend_literal:n</code>
 25 , 31 ,
	34 , 38 , 45 , 50 , 57 , 60 , 62 , 104 , 107 ,
	109 , 111 , 115 , 261 , 274 , 422 , 430 ,
	531 , 538 , 736 , 941 , 948 , 954 , 1014 ,
	1024 , 1326 , 1477 , 1512 , 1522 , 1628 ,
	2691 , 2815 , 2869 , 2873 , 2878 , 2883
	<code>__kernel_backend_literal_page:n</code>
 73 , 106 , 2685 , 2687 , 2888 , 2890
	<code>__kernel_backend_literal_pdf:n</code>
 65 , 103 , 187 , 244 , 774 , 921
	<code>__kernel_backend_literal_-</code>
	<code>postscript:n</code> 30 , 51 , 52 , 56 ,
	133 , 134 , 136 , 137 , 145 , 157 , 172 , 527
	<code>__kernel_backend_literal_svg:n</code>
 114 , 118 , 120 ,
	122 , 288 , 290 , 307 , 1030 , 1298 , 1309
	<code>__kernel_backend_matrix:n</code>
 93 , 209 , 230 , 927
	<code>__kernel_backend_postscript:n</code>
 33 , 424 , 730 , 1646
	<code>__kernel_backend_postscript_-</code>
	<code>header:n</code> 36 , 1732 , 1738 ,
	1745 , 1751 , 1790 , 1828 , 1969 , 2076
	<code>__kernel_backend_scope_begin:</code> 5 ,
	59 , 81 , 108 , 117 , 131 , 155 , 170 , 186 ,
	203 , 229 , 243 , 260 , 273 , 780 , 1009 , 1296
	<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 , 781 , 1026 , 1310
	<code>\l__kernel_color_stack_int</code>
 471 , 511 , 520 , 909 , 917
	L
	<code>\landscape</code> 2756 , 2758
	M
	math commands:
	<code>\c_math_toggle_token</code> 2205 , 2215
	mode commands:
	<code>\mode_if_horizontal:TF</code> 2311 , 2318

<code>\mode_if_math:TF</code>	2199	<code>__pdf_backend_link_end:</code>	2183, 2441, 2796
O		<code>__pdf_backend_link_end_aux:</code> ..	2183
<code>\oddsidemargin</code>	2227	<code>\g__pdf_backend_link_int</code>	2174, 2242, 2246, 2354
P		<code>__pdf_backend_link_last:</code>	2353, 2459, 2813
pdf internal commands:		<code>__pdf_backend_link_margin:n</code> ..	2355, 2467, 2814
<code>__pdf_backend:n</code>	2690, 2694, 2696, 2722, 2727, 2736, 2785, 2802, 2812, 2818, 2845, 2846, 2854	<code>\g__pdf_backend_link_math_bool</code> ..	2177, 2200, 2201, 2204, 2214
<code>__pdf_backend_annotation:nnnn</code> ..	2136, 2423, 2766	<code>__pdf_backend_link_minima:</code> ..	2183
<code>__pdf_backend_annotation-</code> <code>aux:nnnn</code>	2136, 2766	<code>__pdf_backend_link_outterbox:n</code>	2183
<code>\g__pdf_backend_annotation_int</code> ..	2135, 2140, 2164, 2173, 2765, 2784, 2795	<code>\g__pdf_backend_link_sf_int</code>	2176, 2309, 2320, 2321
<code>__pdf_backend_annotation_last:</code> ..	2172, 2433, 2794	<code>__pdf_backend_link_sf_restore:</code>	2183
<code>__pdf_backend_bdc:nn</code>	2417, 2684, 2887, 2908	<code>__pdf_backend_link_sf_save:</code> ..	2183
<code>__pdf_backend_catalog_gput:nn</code> ..	1648, 2508, 2693, 2893	<code>\l__pdf_backend_model_box</code> ..	2134, 2219, 2256, 2264, 2275, 2290, 2297
<code>__pdf_backend_compress_objects:n</code>	2411, 2615, 2868, 2902	<code>__pdf_backend_objcompresslevel:n</code>	2615
<code>__pdf_backend_compresslevel:n</code> ..	2411, 2615, 2868, 2902	<code>\g__pdf_backend_object_int</code>	1652, 1656, 1659, 1711, 1714, 1727, 1731, 2139, 2140, 2144, 2163, 2164, 2167, 2241, 2242, 2697, 2701, 2704, 2744, 2746, 2751, 2783, 2784, 2787
<code>\l__pdf_backend_content_box</code> 2133, 2202, 2231, 2234, 2236, 2265, 2276		<code>__pdf_backend_object_last:</code>	1730, 2607, 2750, 2895
<code>__pdf_backend_destination:nn</code>	2362, 2474, 2816	<code>__pdf_backend_object_new:nn</code> ..	1654, 2523, 2699, 2895
<code>__pdf_backend_destination-</code> <code>rectangle:nn</code>	2362, 2474, 2816	<code>__pdf_backend_object_now:nn</code> ..	1709, 2578, 2742, 2895
<code>__pdf_backend_emc:</code>	2417, 2684, 2887, 2908	<code>\g__pdf_backend_object_prop</code>	1652, 1660, 1671, 1681, 2522, 2532, 2554, 2697, 2705, 2712
<code>__pdf_backend_info_gput:nn</code>	1648, 2508, 2693, 2893	<code>__pdf_backend_object_ref:n</code> 1654, 1668, 1682, 2523, 2699, 2718, 2895	
<code>__pdf_backend_link:nw</code>	2183	<code>__pdf_backend_object_write:nn</code> ..	1664, 2545, 2709, 2895
<code>__pdf_backend_link_aux:nw</code>	2183	<code>__pdf_backend_object_write:nnn</code>	2709
<code>__pdf_backend_link_begin:n</code> ..	2796	<code>__pdf_backend_object_write-</code> <code>array:nn</code>	1664, 2709
<code>__pdf_backend_link_begin:nnnw</code>	2441	<code>__pdf_backend_object_write-</code> <code>dict:nn</code>	1664, 2709
<code>__pdf_backend_link_begin:nw</code>	2184, 2186, 2187	<code>__pdf_backend_object_write-</code> <code>fstream:nn</code>	2709
<code>__pdf_backend_link_begin_aux:nw</code>	2190, 2192	<code>__pdf_backend_object_write-</code> <code>stream:nn</code>	1664, 2709
<code>__pdf_backend_link_begin-</code> <code>goto:nnw</code>	2183, 2441, 2796	<code>__pdf_backend_object_write-</code> <code>stream:nnn</code>	1664
<code>__pdf_backend_link_begin-</code> <code>user:nnw</code>	2183, 2441, 2796	<code>__pdf_backend_object_write-</code> <code>stream:nnnn</code>	2709
<code>\g__pdf_backend_link_bool</code>	2178, 2189, 2194, 2209, 2252		
<code>\g__pdf_backend_link_dict_tl</code>	2175, 2197, 2247		

