

File I

Implementation

1 l3backend-basics Implementation

```
1 <*initex | package>
```

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

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

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

The order of the backend code here is such that we get somewhat logical outcomes in terms of code sharing whilst keeping things readable. (Trying to mix all of the code by concept is almost unmanageable.) The key parts which are shared are

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

The one shared function for all backends is access to the basic `\special` primitive: it has slightly odd expansion behaviour so a wrapper is provided.

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

(End definition for `__kernel_backend_literal:e`.)

1.1 dvips backend

29 `<*dvips>`

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

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

(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 $\backslash cs_new_protected:Npn _kernel_backend_scope_begin:$
118 $\{ _kernel_backend_literal_svg:n \{ <g> \} \}$
119 $\backslash cs_new_protected:Npn _kernel_backend_scope_end:$
120 $\{ _kernel_backend_literal_svg:n \{ </g> \} \}$

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

(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 $\backslash cs_new_protected:Npn _box_backend_clip:N \#1$
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 $\text{T}_{\text{E}}\text{X}$ box and keep the reference point the same!

```

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

```

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

```

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

```

```

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

```

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

`__box_backend_rotate:Nn` Rotation has a dedicated operation which includes a centre-of-rotation optional pair. That can be picked up from the backend syntax, so there is no need to worry about the transformation matrix.

```

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

```

(End definition for `__box_backend_rotate:Nn`.)

`__box_backend_scale:Nnn` In contrast to rotation, we have to account for the current position in this case. That is done using a couple of translations in addition to the scaling (which is therefore done backward with a flip).

```

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

```

(End definition for `_box_backend_scale:Nnn`.)

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

3 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>
```

`_color_backend_pickup:N` Allow for L^AT_EX 2_ε color. Here, the possible input values are limited: dvips-style colors can mainly be taken as-is with the exception spot ones (here we need a model and a tint).

```
375 <*package>
376 \cs_new_protected:Npn \_color_backend_pickup:N #1 { }
377 \AtBeginDocument
378 {
379   \@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` 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
```

```
\_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
```

```

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.

```

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

```

\__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_fill:n
\__draw_backend_color_fill:x
\__draw_backend_color_stroke:n
\__draw_backend_color_stroke:x
239 \__draw_backend_color_fill_cmyk:nnnn #1#2#3#4
240 {
241     \__draw_backend_color_fill:x
242     {
243         \fp_eval:n {#1} ~ \fp_eval:n {#2} ~

```

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.

```

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     { /color.fc ~ { #1 } ~ def }
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 </divisvgm>
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 \l__graphics_backend_dir_str \sys_if_shell:T
\__graphics_backend_getbb_eps:nm \l__graphics_backend_name_str
\__graphics_backend_include_eps:n \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:nn
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>`

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

```

1452 <*initex>
1453 \use:n
1454 </initex>
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.

```

1475 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1476 {
1477   \__kernel_backend_literal:x
1478 {
1479     PSfile = #1 \c_space_tl
1480     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1481     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
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_XeTeXpdfpicfile: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_eps:n` Simply use the generic function.

```
1616 <*initex>
1617 \use:n
1618 </initex>
1619 <*package>
1620 \AtBeginDocument
1621 </package>
1622 { \cs_new_eq:NN \_graphics_backend_getbb_eps:n \graphics_read_bb:n }

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

`_graphics_backend_getbb_png:n` These can be included by extracting the bounding box data.

```
\_graphics_backend_getbb_jpg:n
1623 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1624 {
1625   \int_zero:N \l_graphics_page_int
1626   \tl_clear:N \l_graphics_pagebox_tl
1627   \graphics_extract_bb:n {#1}
1628 }
1629 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n

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

`_graphics_backend_getbb_pdf:n` Same as for dvipdfmx: use the generic function

```
1630 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1631 {
1632   \tl_clear:N \l_graphics_decodearray_tl
1633   \bool_set_false:N \l_graphics_interpolate_bool
1634   \graphics_extract_bb:n {#1}
1635 }

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

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

```
\_graphics_backend_include_pdf:n
\_graphics_backend_include:nn
1636 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1637 { \_graphics_backend_include:nn { PSfile } {#1} }
1638 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1639 { \_graphics_backend_include:nn { pdffile } {#1} }
1640 \cs_new_protected:Npn \_graphics_backend_include:nn #1#2
1641 {
1642   \__kernel_backend_literal:x
1643   {
1644     #1 = #2 \c_space_tl
1645     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1646     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1647     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1648     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1649   }
1650 }

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

```

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

```

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

```

1651 \cs_new_protected:Npn \__graphics_backend_include_png:n #1
1652 {
1653     \__kernel_backend_literal:x
1654     {
1655         dvisvgm:img~
1656         \dim_to_decimal:n { \l_graphics_ury_dim } ~
1657         \dim_to_decimal:n { \l_graphics_ury_dim } ~
1658         \__graphics_backend_include_bitmap_quote:w #1 " #1 " \q_stop
1659     }
1660 }
1661 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_png:n
1662 \cs_new:Npn \__graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \q_stop
1663 { " #2 " }

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

1664 </dvisvgm>
1665 </initex | package>

```

6 l3backend-pdf Implementation

```

1666 <*initex | package>
1667 <@@=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

```

```

1668 \box_new:N \l__pdf_internal_box

(End definition for \l__pdf_internal_boxf.)

```

6.2 dvips backend

```

1669 <*dvips>

```

Used often enough it should be a separate function.

```

\__pdf_backend_pdfmark:n
\__pdf_backend_pdfmark:x

```

```

1670 \cs_new_protected:Npn \__pdf_backend_pdfmark:n #1
1671 { \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }
1672 \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
1673 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
1674 { \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
1675 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
1676 { \__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

For tracking objects to allow finalisation.

```

\g__pdf_backend_object_int
\g__pdf_backend_object_prop
1677 \int_new:N \g__pdf_backend_object_int
1678 \prop_new:N \g__pdf_backend_object_prop

(End definition for \g__pdf_backend_object_int and \g__pdf_backend_object_prop.)

```

Tracking objects is similar to dvipdfmx.

```

\__pdf_backend_object_new:nn
\__pdf_backend_object_ref:n
1679 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
1680 {
1681   \int_gincr:N \g__pdf_backend_object_int
1682   \int_const:cn
1683     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
1684     { \g__pdf_backend_object_int }
1685   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
1686 }
1687 \cs_new:Npn \__pdf_backend_object_ref:n #1
1688 { { pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } } }

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

```

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

```

\__pdf_backend_object_write:nn
\__pdf_backend_object_write:nx
\__pdf_backend_object_write_array:nn
\__pdf_backend_object_write_dict:nn
\__pdf_backend_object_write_stream:nn
\__pdf_backend_object_write_stream:nnn
1689 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
1690 {
1691   \__pdf_backend_pdfmark:x
1692   {
1693     /objdef ~ \__pdf_backend_object_ref:n {#1}
1694     /type
1695     \str_case_e:nn
1696       { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
1697       {
1698         { array } { /array }
1699         { dict } { /dict }
1700         { fstream } { /stream }
1701         { stream } { /stream }
1702       }
1703     /OBJ
1704   }
1705   \use:c
1706     { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
1707     { \__pdf_backend_object_ref:n {#1} } {#2}
1708 }
1709 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
1710 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2

```

```

1711 {
1712   \__pdf_backend_pdfmark:x
1713   { #1 [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
1714 }
1715 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
1716 {
1717   \__pdf_backend_pdfmark:x
1718   { #1 << \exp_not:n {#2} >> /PUT }
1719 }
1720 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
1721 {
1722   \exp_args:Nx
1723   \__pdf_backend_object_write_stream:nnn {#1} #2
1724 }
1725 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
1726 {
1727   \__kernel_backend_postscript:n
1728   {
1729     [nobreak]
1730     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
1731     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
1732   }
1733 }

```

(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
1734 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
1735 {
1736   \int_gincr:N \g__pdf_backend_object_int
1737   \__pdf_backend_pdfmark:x
1738   {
1739     /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
1740     /type
1741     \str_case:nn
1742     {#1}
1743     {
1744       { array } { /array }
1745       { dict } { /dict }
1746       { fstream } { /stream }
1747       { stream } { /stream }
1748     }
1749     /OBJ
1750   }
1751   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
1752   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
1753 }
1754 \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.

```

1755 \cs_new:Npn \__pdf_backend_object_last:
1756 { { 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.

```
1757 \__kernel_backend_postscript_header:n
1758 {
1759     true ~ setglobal ~
1760     /pdf.globaldict ~ 4 ~ dict ~ def ~
1761     false ~ setglobal
1762 }
```

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

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

```
1763 \__kernel_backend_postscript_header:n
1764 {
1765     /pdf.cvs { 65534 ~ string ~ cvs } def
1766     /pdf.dvi.pt { 72.27 ~ mul ~ Resolution ~ div } def
1767     /pdf.pt.dvi { 72.27 ~ div ~ Resolution ~ mul } def
1768     /pdf.rect.ht { dup ~ 1 ~ get ~ neg ~ exch ~ 3 ~ get ~ add } def
1769 }
```

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

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

```
1770 \__kernel_backend_postscript_header:n
1771 {
1772     /pdf.linkmargin { 1 ~ pdf.pt.dvi } def
1773     /pdf.linkdp.pad { 0 } def
1774     /pdf.linkht.pad { 0 } def
1775 }
```

(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.

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

```

1789         currentpoint
1790         /pdf.ury ~ exch ~ def
1791         /pdf.urx ~ exch ~ def
1792     }
1793     def
1794 /pdf.save.linkll
1795 {
1796     currentpoint ~
1797     pdf.linkmargin ~ add ~
1798     pdf.linkdp.pad ~ add
1799     /pdf.lly ~ exch ~ def ~
1800     pdf.linkmargin ~ sub
1801     /pdf.llx ~ exch ~ def
1802 }
1803 def
1804 /pdf.save.linkur
1805 {
1806     currentpoint ~
1807     pdf.linkmargin ~ sub ~
1808     pdf.linkht.pad ~ sub
1809     /pdf.ury ~ exch ~ def ~
1810     pdf.linkmargin ~ add
1811     /pdf.urx ~ exch ~ def
1812 }
1813 def
1814 }

```

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

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

```

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

```

```

1835     matrix ~ invertmatrix ~
1836     matrix ~ concatmatrix ~
1837     cvx ~ exec
1838     /pdf.dev.y ~ exch ~ def
1839     /pdf.dev.x ~ exch ~ def
1840     /pdf.tmpd ~ exch ~ def
1841     /pdf.tmpc ~ exch ~ def
1842     /pdf.tmpb ~ exch ~ def
1843     /pdf.tmpa ~ exch ~ def ~
1844     pdf.dest.x ~ pdf.tmpa ~ mul ~
1845     pdf.dest.y ~ pdf.tmpc ~ mul ~ add ~
1846     pdf.dev.x ~ add ~
1847     pdf.dest.x ~ pdf.tmpb ~ mul ~
1848     pdf.dest.y ~ pdf.tmpd ~ mul ~ add ~
1849     pdf.dev.y ~ add
1850   }
1851   def
1852 }

```

(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	1853 _kernel_backend_postscript_header:n
pdf.brokenlink.skip	1854 {
pdf.brokenlink.dict	1855 /pdf.bordertracking ~ false ~ def
pdf.bordertracking.endpage	1856 /pdf.bordertracking.begin
pdf.bordertracking.continue	1857 {
pdf.originx	1858 SDict ~ /pdf.bordertracking ~ true ~ put ~
pdf.originy	1859 SDict ~ /pdf.leftboundary ~ undef ~
	1860 SDict ~ /pdf.rightboundary ~ undef ~
	1861 /a ~ where
	1862 {
	1863 /a
	1864 {
	1865 currentpoint ~ pop ~
	1866 SDict /pdf.rightboundary ~ known ~ dup
	1867 {
	1868 SDict /pdf.rightboundary ~ get ~ 2 ~ index ~ lt
	1869 { not }
	1870 if
	1871 }
	1872 if
	1873 { pop }
	1874 { SDict ~ exch /pdf.rightboundary ~ exch ~ put }
	1875 ifelse ~
	1876 moveto ~
	1877 currentpoint ~ pop ~
	1878 SDict /pdf.leftboundary ~ known ~ dup
	1879 {
	1880 SDict /pdf.leftboundary ~ get ~ 2 ~ index ~ gt
	1881 { not }


```

1882         if
1883     }
1884     if
1885     { pop }
1886     { SDict ~ exch /pdf.leftboundary ~ exch ~ put }
1887     ifelse
1888 }
1889 put
1890 }
1891 if
1892 }
1893 def
1894 /pdf.bordertracking.end
1895 {
1896     /a ~ where { /a { moveto } put } if
1897     /x ~ where { /x { 0 ~ exch ~ rmoveto } put } if ~
1898     SDict /pdf.leftboundary ~ known
1899     { pdf.outerbox ~ 0 ~ pdf.leftboundary ~ put }
1900     if ~
1901     SDict /pdf.rightboundary ~ known
1902     { pdf.outerbox ~ 2 ~ pdf.rightboundary ~ put }
1903     if ~
1904     SDict /pdf.bordertracking ~ false ~ put
1905 }
1906 def
1907 /pdf.bordertracking.endpage
1908 {
1909     pdf.bordertracking
1910     {
1911         pdf.bordertracking.end ~
1912         true ~ setglobal ~
1913         pdf.globaldict
1914         /pdf.brokenlink.rect [ pdf.outerbox ~ aload ~ pop ] put ~
1915         pdf.globaldict
1916         /pdf.brokenlink.skip ~ pdf.baselineskip ~ put ~
1917         pdf.globaldict
1918         /pdf.brokenlink.dict ~
1919         pdf.link.dict ~ pdf.cvs ~ put ~
1920         false ~ setglobal ~
1921         mark ~ pdf.link.dict ~ cvx ~ exec ~ /Rect
1922         [
1923             pdf.llx ~
1924             pdf.lly ~
1925             pdf.outerbox ~ 2 ~ get ~ pdf.linkmargin ~ add ~
1926             currentpoint ~ exch ~ pop ~
1927             pdf.outerbox ~ pdf.rect.ht ~ sub ~ pdf.linkmargin ~ sub
1928         ]
1929         /ANN ~ pdf.pdfmark
1930     }
1931     if
1932 }
1933 def
1934 /pdf.bordertracking.continue
1935 {

```

```

1936 /pdf.link.dict ~ pdf.globaldict
1937 /pdf.brokenlink.dict ~ get ~ def
1938 /pdf.outerbox ~ pdf.globaldict
1939 /pdf.brokenlink.rect ~ get ~ def
1940 /pdf.baselineskip ~ pdf.globaldict
1941 /pdf.brokenlink.skip ~ get ~ def ~
1942 pdf.globaldict ~ dup ~ dup
1943 /pdf.brokenlink.dict ~ undef
1944 /pdf.brokenlink.skip ~ undef
1945 /pdf.brokenlink.rect ~ undef ~
1946 currentpoint
1947 /pdf.originy ~ exch ~ def
1948 /pdf.originx ~ exch ~ def
1949 /a ~ where
1950 {
1951     /a
1952     {
1953         moveto ~
1954         SDict ~
1955         begin ~
1956         currentpoint ~ pdf.originy ~ ne ~ exch ~
1957         pdf.originx ~ ne ~ or
1958         {
1959             pdf.save.linkll
1960             /pdf.lly ~
1961             pdf.lly ~ pdf.outerbox ~ 1 ~ get ~ sub ~ def ~
1962             pdf.bordertracking.begin
1963         }
1964         if ~
1965         end
1966     }
1967     put
1968 }
1969 if
1970 /x ~ where
1971 {
1972     /x
1973     {
1974         0 ~ exch ~ rmoveto ~
1975         SDict~
1976         begin ~
1977         currentpoint ~
1978         pdf.originy ~ ne ~ exch ~ pdf.originx ~ ne ~ or
1979         {
1980             pdf.save.linkll
1981             /pdf.lly ~
1982             pdf.lly ~ pdf.outerbox ~ 1 ~ get ~ sub ~ def ~
1983             pdf.bordertracking.begin
1984         }
1985         if ~
1986         end
1987     }
1988     put
1989 }

```

```

1990         if
1991     }
1992     def
1993 }

```

(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.

```

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

```

```

2036         pdf.baselineskip ~ add ~
2037         1 ~ exch ~ put ~
2038     dup ~ dup ~
2039     3 ~ get ~
2040     pdf.baselineskip ~ add ~
2041     3 ~ exch ~ put ~
2042     /pdf.currentrect ~ exch ~ def ~
2043     pdf.breaklink.write
2044 }
2045 1 ~ index ~ 3 ~ get ~
2046 pdf.linkmargin ~ 2 ~ mul ~ add ~
2047 pdf.outerbox ~ pdf.rect.ht ~ add ~
2048 2 ~ index ~ 1 ~ get ~ sub ~
2049 pdf.baselineskip ~ div ~ round ~ cvi ~ 1 ~ sub ~
2050 exch ~
2051 repeat ~
2052 pdf.currentrect ~
2053 dup ~
2054 pdf.outerbox ~ 0 ~ get ~
2055 pdf.linkmargin ~ sub ~
2056 0 ~ exch ~ put ~
2057 dup ~ dup ~
2058 1 ~ get ~
2059 pdf.baselineskip ~ add ~
2060 1 ~ exch ~ put ~
2061 dup ~ dup ~
2062 3 ~ get ~
2063 pdf.baselineskip ~ add ~
2064 3 ~ exch ~ put ~
2065 dup ~ 2 ~ index ~ 2 ~ get ~ 2 ~ exch ~ put
2066 /pdf.currentrect ~ exch ~ def ~
2067 pdf.breaklink.write ~
2068 SDict /pdf.pdfmark.good ~ false ~ put ~
2069 exit
2070 }
2071 { pdf.count ~ 2 ~ sub /pdf.count ~ exch ~ def }
2072 ifelse
2073 }
2074 loop
2075 }
2076 if
2077 /ANN
2078 }
2079 def
2080 /pdf.breaklink.write
2081 {
2082     counttomark ~ 1 ~ sub ~
2083     index /_objdef ~ eq
2084     {
2085         counttomark ~ -2 ~ roll ~
2086         dup ~ wcheck ~
2087         {
2088             readonly ~
2089             counttomark ~ 2 ~ roll

```

```

2090         }
2091         { pop ~ pop }
2092     ifelse
2093 }
2094 if ~
2095 counttomark ~ 1 ~ add ~ copy ~
2096 pop ~ pdf.currentrect
2097 /ANN ~ pdfmark
2098 }
2099 def
2100 }

```

(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.

```

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

```

```

2136     def
2137 /pdf.pdfmark.store
2138 {
2139     /pdf.pdfmark.dict ~ 65534 ~ dict ~ def ~
2140     counttomark ~ 1 ~ add ~ copy ~
2141     pop
2142     {
2143         dup ~ mark ~ eq
2144         {
2145             pop ~
2146             exit
2147         }
2148         {
2149             pdf.pdfmark.dict ~
2150             begin ~ def ~ end
2151         }
2152         ifelse
2153     }
2154     loop
2155 }
2156 def
2157 }

```

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

`\l__pdf_backend_content_box` The content of an annotation.
2158 `\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.
2159 `\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.
2160 `\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 $\text{\LaTeX} 2_{\epsilon}$ picture of zero size). Once the data is collected, use it to set up the annotation
pdf.urx border. There is a split into two parts here to allow an easy way of applying the Adobe
pdf.ury Reader fix.

```

2161 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2162 {
2163     \__pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4}
2164     \int_gincr:N \g__pdf_backend_object_int
2165     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2166     \__pdf_backend_pdfmark:x
2167     {
2168

```

```

2169         /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2170 pdf.rect ~
2171 #4 ~
2172 /ANN
2173     }
2174 }
2175 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2176 {
2177     \box_move_down:nn {#3}
2178     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2179     \hbox:n {#4}
2180     \box_move_up:nn {#2}
2181     {
2182         \hbox:n
2183         {
2184             \tex_kern:D \dim_eval:n {#1} \scan_stop:
2185             \__kernel_backend_postscript:n { pdf.save.ur }
2186         }
2187     }
2188     \int_gincr:N \g__pdf_backend_object_int
2189     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2190     \__pdf_backend_pdfmark:x
2191     {
2192         /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2193         pdf.rect
2194         /ANN
2195     }
2196 }

```

(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.

```

2197 \cs_new:Npn \__pdf_backend_annotation_last:
2198 { { 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.

```

2199 \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.

```

2200 \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.

```

2201 \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.

```

2202 \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.

2203 `\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.

2204 `\tl_new:N \l__pdf_breaklink_pdfmark_tl`

2205 `\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.

2206 `\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.

2207 `\cs_new_eq:NN __pdf_breaklink_usebox:N \box_use:N`

(End definition for `__pdf_breaklink_usebox:N`.)

`_pdf_backend_link_begin_goto:nnw`

`_pdf_backend_link_begin_user:nnw`

`__pdf_backend_link:nw`

`__pdf_backend_link_aux:nw`

`__pdf_backend_link_end:`

`_pdf_backend_link_end_aux:`

`__pdf_backend_link_minima:`

`_pdf_backend_link_outerbox:n`

`_pdf_backend_link_sf_save:`

`_pdf_backend_link_sf_restore:`

`pdf.linkdp.pad`

`pdf.linkht.pad`

`pdf.llx`

`pdf.lly`

`pdf.ury`

`pdf.link.dict`

`pdf.outerbox`

`pdf.baselineskip`

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

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.

2208 `\cs_new_protected:Npn __pdf_backend_link_begin_goto:nnw #1#2`

2209 `{ __pdf_backend_link_begin:nw { #1 /Subtype /Link /A << /S /GoTo /D (#2) >> } }`

2210 `\cs_new_protected:Npn __pdf_backend_link_begin_user:nnw #1#2`

2211 `{ __pdf_backend_link_begin:nw {#1#2} }`

2212 `\cs_new_protected:Npn __pdf_backend_link_begin:nw #1`

2213 `{`

2214 `\bool_if:NF \g__pdf_backend_link_bool`

2215 `{ __pdf_backend_link_begin_aux:nw {#1} }`

2216 `}`

2217 `\cs_new_protected:Npn __pdf_backend_link_begin_aux:nw #1`

2218 `{`

2219 `\bool_gset_true:N \g__pdf_backend_link_bool`

2220 `_kernel_backend_postscript:n`

2221 `{ /pdf.link.dict (#1) def }`

2222 `\tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}`

2223 `_pdf_backend_link_sf_save:`

2224 `\mode_if_math:TF`


```

2225     { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2226     { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2227 \hbox_set:Nw \l__pdf_backend_content_box
2228   \__pdf_backend_link_sf_restore:
2229   \bool_if:NT \g__pdf_backend_link_math_bool
2230     { \c_math_toggle_token }
2231 }
2232 \cs_new_protected:Npn \__pdf_backend_link_end:
2233 {
2234   \bool_if:NT \g__pdf_backend_link_bool
2235     { \__pdf_backend_link_end_aux: }
2236 }
2237 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2238 {
2239   \bool_if:NT \g__pdf_backend_link_math_bool
2240     { \c_math_toggle_token }
2241   \__pdf_backend_link_sf_save:
2242   \hbox_set_end:
2243   \__pdf_backend_link_minima:
2244   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2245   \exp_args:Nx \__pdf_backend_link_outerbox:n
2246   {
2247     <*initex>
2248       \l_galley_total_left_margin_dim
2249     </initex>
2250     <*package>
2251       \int_if_odd:nTF { \value { page } }
2252         { \oddsidemargin }
2253         { \evensidemargin }
2254     </package>
2255   }
2256   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2257   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2258   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2259   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2260   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2261   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2262   {
2263     \hbox:n
2264       { \__kernel_backend_postscript:n { pdf.save.linkur } }
2265   }
2266   \int_gincr:N \g__pdf_backend_object_int
2267   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2268   \__kernel_backend_postscript:x
2269   {
2270     mark
2271     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2272     \g__pdf_backend_link_dict_tl \c_space_tl
2273     pdf.rect
2274     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2275   }
2276   \__pdf_backend_link_sf_restore:
2277   \bool_gset_false:N \g__pdf_backend_link_bool
2278 }

```

```

2279 \cs_new_protected:Npn \__pdf_backend_link_minima:
2280 {
2281   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2282   \__kernel_backend_postscript:x
2283   {
2284     /pdf.linkdp.pad ~
2285     \dim_to_decimal:n
2286     {
2287       \dim_max:nn
2288       {
2289         \box_dp:N \l__pdf_backend_model_box
2290         - \box_dp:N \l__pdf_backend_content_box
2291       }
2292       { Opt }
2293     } ~
2294     pdf.pt.dvi ~ def
2295     /pdf.linkht.pad ~
2296     \dim_to_decimal:n
2297     {
2298       \dim_max:nn
2299       {
2300         \box_ht:N \l__pdf_backend_model_box
2301         - \box_ht:N \l__pdf_backend_content_box
2302       }
2303       { Opt }
2304     } ~
2305     pdf.pt.dvi ~ def
2306   }
2307 }
2308 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2309 {
2310   \__kernel_backend_postscript:x
2311   {
2312     /pdf.outerbox
2313     [
2314       \dim_to_decimal:n {#1} ~
2315       \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2316       ⟨*initex⟩
2317       \dim_to_decimal:n { #1 + \l_galley_text_width_dim } ~
2318       ⟨/initex⟩
2319       ⟨*package⟩
2320       \dim_to_decimal:n { #1 + \textwidth } ~
2321       ⟨/package⟩
2322       \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2323     ]
2324     [ exch { pdf.pt.dvi } forall ] def
2325     /pdf.baselineskip ~
2326     \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2327     { pdf.pt.dvi ~ def }
2328     { pop ~ pop }
2329     ifelse
2330   }
2331 }
2332 \cs_new_protected:Npn \__pdf_backend_link_sf_save:

```

```

2333 {
2334   \int_gset:Nn \g__pdf_backend_link_sf_int
2335   {
2336     \mode_if_horizontal:TF
2337     { \tex_spacefactor:D }
2338     { 0 }
2339   }
2340 }
2341 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2342 {
2343   \mode_if_horizontal:T
2344   {
2345     \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2346     { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2347   }
2348 }

```

(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.

```

2349 (*package)
2350 \use_none:n
2351 {
2352   \cs_if_exist:NT \@makecol@hook
2353   {
2354     \tl_put_right:Nn \@makecol@hook
2355     {
2356       \box_if_empty:NF \@cclv
2357       {
2358         \vbox_set:Nn \@cclv
2359         {
2360           \__kernel_backend_postscript:n
2361           {
2362             pdf.globaldict /pdf.brokenlink.rect ~ known
2363             { pdf.bordertracking.continue }
2364             if
2365           }
2366           \vbox_unpack_drop:N \@cclv
2367           \__kernel_backend_postscript:n
2368           { pdf.bordertracking.endpage }
2369         }
2370       }
2371     }
2372     \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2373     \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2374     \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2375   }
2376 }
2377 </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.

```
2378 \cs_new:Npn \__pdf_backend_link_last:
2379 { { 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.

```
2380 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2381 {
2382   \__kernel_backend_postscript:x
2383   {
2384     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2385   }
2386 }
```

(End definition for `__pdf_backend_link_margin:n.`)

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

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

```

2421     { \box_ht:N \l__pdf_internal_box }
2422     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } } }
2423 \__pdf_backend_pdfmark:n
2424 {
2425     /View
2426     [
2427         /FitR ~
2428         pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2429         pdf.urx ~ pdf.ury ~ pdf.dest2device
2430     ]
2431     /Dest ( #1 ) cvn
2432     /DEST
2433 }
2434 \group_end:
2435 }

```

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

6.2.4 Structure

__pdf_backend_compresslevel:n These are all no-ops.

```

2436 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2437 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1 { }

```

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

__pdf_backend_version_major_gset:n Data not available!

```

2438 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2439 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1 { }

```

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

__pdf_backend_version_major: Data not available!

```

2440 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2441 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

```

(End definition for __pdf_backend_version_major: and __pdf_backend_version_minor:.)

6.2.5 Marked content

__pdf_backend_bdc:nn Simple wrappers.

```

2442 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2443 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2444 \cs_new_protected:Npn \__pdf_backend_emc:
2445 { \__pdf_backend_pdfmark:n { /EMC } }

```

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

```

2446 </dvips>

```

6.3 pdfmode backend

2447 `*pdfmode\`

6.3.1 Annotations

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

```
2448 \cs_new_protected:Npx \_pdf_backend_annotation:nnnn #1#2#3#4
2449 {
2450   \cs_if_exist:NTF \tex_pdfextension:D
2451     { \tex_pdfextension:D annot ~ }
2452     { \tex_pdfannot:D }
2453   width ~ \exp_not:N \dim_eval:n {#1} ~
2454   height ~ \exp_not:N \dim_eval:n {#2} ~
2455   depth ~ \exp_not:N \dim_eval:n {#3} ~
2456   {#4}
2457 }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

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

```
2458 \cs_new:Npx \_pdf_backend_annotation_last:
2459 {
2460   \exp_not:N \int_value:w
2461   \cs_if_exist:NTF \tex_pdffeedback:D
2462     { \exp_not:N \tex_pdffeedback:D lastannot ~ }
2463     { \exp_not:N \tex_pdflastannot:D }
2464   \c_space_tl 0 ~ R
2465 }
```

(End definition for `_pdf_backend_annotation_last:`.)

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

```
\_pdf_backend_link_begin_user:nnw
\_pdf_backend_link_begin:nnnw
\_pdf_backend_link_end:
2466 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2467 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2468 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2469 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2470 \cs_new_protected:Npx \_pdf_backend_link_begin:nnnw #1#2#3
2471 {
2472   \cs_if_exist:NTF \tex_pdfextension:D
2473     { \tex_pdfextension:D startlink ~ }
2474     { \tex_pdfstartlink:D }
2475   attr {#1}
2476   #2 {#3}
2477 }
2478 \cs_new_protected:Npx \_pdf_backend_link_end:
2479 {
2480   \cs_if_exist:NTF \tex_pdfextension:D
2481     { \tex_pdfextension:D endlink \scan_stop: }
2482     { \tex_pdfendlink:D }
2483 }
```

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

`__pdf_backend_link_last:` Formatted for direct use.

```

2484 \cs_new:Npx \__pdf_backend_link_last:
2485 {
2486   \exp_not:N \int_value:w
2487   \cs_if_exist:NTF \tex_pdffeedback:D
2488     { \exp_not:N \tex_pdffeedback:D lastlink ~ }
2489     { \exp_not:N \tex_pdflastlink:D }
2490   \c_space_tl 0 ~ R
2491 }

```

(End definition for `__pdf_backend_link_last:`.)

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

```

2492 \cs_new_protected:Npx \__pdf_backend_link_margin:n #1
2493 {
2494   \cs_if_exist:NTF \tex_pdfvariable:D
2495     { \exp_not:N \tex_pdfvariable:D linkmargin }
2496     { \exp_not:N \tex_pdflinkmargin:D }
2497     \exp_not:N \dim_eval:n {#1} \scan_stop:
2498 }

```

(End definition for `__pdf_backend_link_margin:n`.)

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

`__pdf_backend_destination_rectangle:nn`

```

2499 \cs_new_protected:Npx \__pdf_backend_destination:nn #1#2
2500 {
2501   \cs_if_exist:NTF \tex_pdfextension:D
2502     { \exp_not:N \tex_pdfextension:D dest ~ }
2503     { \exp_not:N \tex_pdfdest:D }
2504     name {#1}
2505     \exp_not:N \str_case:nnF {#2}
2506     {
2507       { xyz } { xyz }
2508       { fit } { fit }
2509       { fitb } { fitb }
2510       { fitbh } { fitbh }
2511       { fitbv } { fitbv }
2512       { fith } { fith }
2513       { fitv } { fitv }
2514     }
2515     { xyz ~ zoom \exp_not:N \fp_eval:n { #2 * 10 } }
2516     \scan_stop:
2517 }
2518 \cs_new_protected:Npx \__pdf_backend_destination_rectangle:nn #1#2
2519 {
2520   \group_begin:
2521     \hbox_set:Nn \l__pdf_internal_box {#2}
2522     \cs_if_exist:NTF \tex_pdfextension:D
2523       { \exp_not:N \tex_pdfextension:D dest ~ }
2524       { \exp_not:N \tex_pdfdest:D }
2525     name {#1}
2526     fitr ~

```

```

2527         width \exp_not:N \box_wd:N \l__pdf_internal_box
2528         height \exp_not:N \box_ht:N \l__pdf_internal_box
2529         depth \exp_not:N \box_dp:N \l__pdf_internal_box
2530         \box_use:N \l__pdf_internal_box
2531     \group_end:
2532 }

```

(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
2533 \cs_new_protected:Npx \__pdf_backend_catalog_gput:nn #1#2
2534 {
2535     \cs_if_exist:NTF \tex_pdfextension:D
2536     { \tex_pdfextension:D catalog }
2537     { \tex_pdfcatalog:D }
2538     { / #1 ~ #2 }
2539 }
2540 \cs_new_protected:Npx \__pdf_backend_info_gput:nn #1#2
2541 {
2542     \cs_if_exist:NTF \tex_pdfextension:D
2543     { \tex_pdfextension:D info }
2544     { \tex_pdfinfo:D }
2545     { / #1 ~ #2 }
2546 }

```

(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.

```

2547 \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
2548 \group_begin:
2549 \cs_set_protected:Npn \__pdf_tmp:w #1#2
2550 {
2551     \cs_new_protected:Npx \__pdf_backend_object_new:nn ##1##2
2552     {
2553         #1 reserveobjnum ~
2554         \int_const:cn
2555         { c__pdf_backend_object_ \exp_not:N \tl_to_str:n {##1} _int }
2556         {#2}
2557         \prop_gput:Nnn \exp_not:N \g__pdf_backend_object_prop {##1} {##2}
2558     }
2559 }
2560 \cs_if_exist:NTF \tex_pdfextension:D
2561 {
2562     \__pdf_tmp:w
2563     { \tex_pdfextension:D obj ~ }
2564     { \exp_not:N \tex_pdffeedback:D lastobj }

```



```

2565     }
2566     { \_pdf_tmp:w { \tex_pdfobj:D } { \tex_pdflastobj:D } }
2567 \group_end:
2568 \cs_new:Npn \_pdf_backend_object_ref:n #1
2569 { \int_use:c { c\_pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

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

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

```

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

```

(End definition for _pdf_backend_object_write:nn, _pdf_exp_not_i:nn, and _pdf_exp_not_ii:nn.)

_pdf_backend_object_now:nn Much like writing, but direct creation.

```

\_pdf_backend_object_now:nx 2603 \group_begin:
2604 \cs_set_protected:Npn \_pdf_tmp:w #1
2605 {
2606     \cs_new_protected:Npn \_pdf_backend_object_now:nn ##1##2
2607     {
2608         \tex_immediate:D #1
2609         \str_case:nn
2610         {##1}

```

```

2611         {
2612             { array } { { [ ~ \exp_not:n {##2} ~ ] } }
2613             { dict } { { << ~ \exp_not:n {##2} ~ >> } }
2614             { fstream }
2615             {
2616                 stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2617                 file ~ { \_pdf_exp_not_ii:nn ##2 }
2618             }
2619             { stream }
2620             {
2621                 stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2622                 { \_pdf_exp_not_ii:nn ##2 }
2623             }
2624         }
2625     }
2626 }
2627 \cs_if_exist:NTF \tex_pdfextension:D
2628 { \_pdf_tmp:w { \tex_pdfextension:D obj ~ } }
2629 { \_pdf_tmp:w { \tex_pdfobj:D } }
2630 \group_end:
2631 \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.

```

2632 \cs_new:Npx \_pdf_backend_object_last:
2633 {
2634     \exp_not:N \int_value:w
2635     \cs_if_exist:NTF \tex_pdffeedback:D
2636     { \exp_not:N \tex_pdffeedback:D lastobj ~ }
2637     { \exp_not:N \tex_pdflastobj:D }
2638     \c_space_tl 0 ~ R
2639 }

```

(End definition for _pdf_backend_object_last:.)

6.3.4 Structure

_pdf_backend_compresslevel:n Simply pass data to the engine.

```

\_pdf_backend_compresslevel:n 2640 \cs_new_protected:Npx \_pdf_backend_compresslevel:n #1
\_pdf_backend_compress_objects:n 2641 {
\_pdf_backend_objcompresslevel:n 2642     \exp_not:N \tex_global:D
2643     \cs_if_exist:NTF \tex_pdfcompresslevel:D
2644     { \tex_pdfcompresslevel:D }
2645     { \tex_pdfvariable:D compresslevel }
2646     \exp_not:N \int_value:w \exp_not:N \int_eval:n {#1} \scan_stop:
2647 }
2648 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2649 {
2650     \bool_if:nTF {#1}
2651     { \_pdf_backend_objcompresslevel:n { 2 } }
2652     { \_pdf_backend_objcompresslevel:n { 0 } }
2653 }
2654 \cs_new_protected:Npx \_pdf_backend_objcompresslevel:n #1

```

```

2655 {
2656   \exp_not:N \tex_global:D
2657   \cs_if_exist:NTF \tex_pdfobjcompresslevel:D
2658     { \tex_pdfobjcompresslevel:D }
2659     { \tex_pdfvariable:D objcompresslevel }
2660     #1 \scan_stop:
2661 }

```

(End definition for `__pdf_backend_compresslevel:n`, `__pdf_backend_compress_objects:n`, and `__pdf_backend_objcompresslevel:n`.)

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

`__pdf_backend_version_minor_gset:n`

```

2662 \cs_new_protected:Npx \__pdf_backend_version_major_gset:n #1
2663 {
2664   \cs_if_exist:NTF \tex_pdfvariable:D
2665     {
2666       \int_compare:nNnT \tex_luatexversion:D > { 106 }
2667       {
2668         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2669         \exp_not:N \int_eval:n {#1} \scan_stop:
2670       }
2671     }
2672     {
2673       \cs_if_exist:NT \tex_pdfmajorversion:D
2674       {
2675         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2676         \exp_not:N \int_eval:n {#1} \scan_stop:
2677       }
2678     }
2679 }
2680 \cs_new_protected:Npx \__pdf_backend_version_minor_gset:n #1
2681 {
2682   \exp_not:N \tex_global:D
2683   \cs_if_exist:NTF \tex_pdfminorversion:D
2684     { \exp_not:N \tex_pdfminorversion:D }
2685     { \tex_pdfvariable:D minorversion }
2686     \exp_not:N \int_eval:n {#1} \scan_stop:
2687 }

```

(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:`

```

2688 \cs_new:Npx \__pdf_backend_version_major:
2689 {
2690   \cs_if_exist:NTF \tex_pdfvariable:D
2691     {
2692       \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2693       { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2694       { 1 }
2695     }
2696     {
2697       \cs_if_exist:NTF \tex_pdfmajorversion:D
2698       { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2699       { 1 }

```

```

2700     }
2701   }
2702   \cs_new:Npx \__pdf_backend_version_minor:
2703   {
2704     \exp_not:N \tex_the:D
2705     \cs_if_exist:NTF \tex_pdfminorversion:D
2706     { \exp_not:N \tex_pdfminorversion:D }
2707     { \tex_pdfvariable:D minorversion }
2708   }

```

(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>.

```

2709 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2710 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2711 \cs_new_protected:Npn \__pdf_backend_emc:
2712 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2713 \</pdfmode>

```

6.4 dvipdfmx backend

```

2714 \*dvipdfmx | xdvipdfmx>

```

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

```

2715 \cs_new_protected:Npx \__pdf_backend:n #1
2716 { \__kernel_backend_literal:n { pdf: #1 } }
2717 \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

```

```

2718 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2719 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2720 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2721 { \__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.

```

2722 \int_new:N \g__pdf_backend_object_int
2723 \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
2724 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2
2725 {
2726   \int_gincr:N \g__pdf_backend_object_int
2727   \int_const:cn
2728   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2729   { \g__pdf_backend_object_int }
2730   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2731 }
2732 \cs_new:Npn \_pdf_backend_object_ref:n #1
2733 { @pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } }
```

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

```
\_pdf_backend_object_write:nn
\_pdf_backend_object_write:nx
\_pdf_backend_object_write:nnn
\_pdf_backend_object_write_array:nn
\_pdf_backend_object_write_dict:nn
\_pdf_backend_object_write_fstream:nn
\_pdf_backend_object_write_stream:nn
\_pdf_backend_object_write_stream:nnnn
2734 \cs_new_protected:Npn \_pdf_backend_object_write:nn #1#2
2735 {
2736   \exp_args:Nx \_pdf_backend_object_write:nnn
2737   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2738 }
2739 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
2740 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
2741 {
2742   \use:c { __pdf_backend_object_write_ #1 :nn }
2743   { __pdf_backend_object_ref:n {#2} } {#3}
2744 }
2745 \cs_new_protected:Npn \_pdf_backend_object_write_array:nn #1#2
2746 {
2747   \_pdf_backend:x
2748   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2749 }
2750 \cs_new_protected:Npn \_pdf_backend_object_write_dict:nn #1#2
2751 {
2752   \_pdf_backend:x
2753   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2754 }
2755 \cs_new_protected:Npn \_pdf_backend_object_write_fstream:nn #1#2
2756 { \_pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2757 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nn #1#2
2758 { \_pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2759 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nnnn #1#2#3#4
2760 {
2761   \_pdf_backend:x
2762   {
2763     #1 stream ~ #2 ~
2764     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2765   }
2766 }
```

(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
2767 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2768 {
```

```

2769 \int_gincr:N \g__pdf_backend_object_int
2770 \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2771 { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2772 {#2}
2773 }
2774 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }
(End definition for \__pdf_backend_object_now:nn.)

```

__pdf_backend_object_last:

```

2775 \cs_new:Npn \__pdf_backend_object_last:
2776 { @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.

```

2777 \bool_new:N \g__pdf_landscape_bool
2778 <*package>
2779 \AtBeginDocument
2780 {
2781   \cs_if_exist:NT \landscape
2782   {
2783     \tl_put_right:Nn \landscape
2784       { \bool_gset_true:N \g__pdf_landscape_bool }
2785     \tl_put_left:Nn \endlandscape
2786       { \bool_gset_false:N \g__pdf_landscape_bool }
2787   }
2788 }
2789 </package>
(End definition for \g__pdf_landscape_bool.)

```

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

```

2790 \int_new:N \g__pdf_backend_annotation_int
(End definition for \g__pdf_backend_annotation_int.)

```

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

```

\__pdf_backend_annotation_aux:nnnn
2791 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2792 {
2793   \bool_if:NTF \g__pdf_landscape_bool
2794   {
2795     \box_move_up:nn {#2}
2796     {
2797       \vbox:n
2798       {
2799         \__pdf_backend_annotation_aux:nnnn
2800         { #2 + #3 } {#1} { 0pt } {#4}
2801       }
2802     }
2803   }

```

```

2804     { \_pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4} }
2805   }
2806 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2807   {
2808     \int_gincr:N \g__pdf_backend_object_int
2809     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2810     \_pdf_backend:x
2811     {
2812       ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
2813       width ~ \dim_eval:n {#1} ~
2814       height ~ \dim_eval:n {#2} ~
2815       depth ~ \dim_eval:n {#3} ~
2816       <</Type/Annot #4 >>
2817     }
2818   }

```

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

_pdf_backend_annotation_last:

```

2819 \cs_new:Npn \_pdf_backend_annotation_last:
2820 { @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.

```

2821 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2822 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2823 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2824 { \_pdf_backend_link_begin:n {#1#2} }
2825 \cs_new_protected:Npn \_pdf_backend_link_begin:n #1
2826 {
2827   \_pdf_backend:n
2828   {
2829     bann
2830     <<
2831     /Type /Annot
2832     #1
2833     >>
2834   }
2835 }
2836 \cs_new_protected:Npn \_pdf_backend_link_end:
2837 { \_pdf_backend:n { eann } }

```

(End definition for _pdf_backend_link_begin_goto:nnw and others.)

_pdf_backend_link_last: Data not available.

```

2838 \cs_new:Npn \_pdf_backend_link_last: { }

```

(End definition for _pdf_backend_link_last:.)

_pdf_backend_link_margin:n

Pass to dvipdfmx.

```

2839 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2840 { \_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`.

```

2841 \cs_new_protected:Npn \\_pdf_backend_destination:nn #1#2
2842 {
2843   \\_pdf_backend:x
2844   {
2845     dest ~ ( \\exp_not:n {#1} )
2846     [
2847       @thispage
2848       \\str_case:nnF {#2}
2849       {
2850         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2851         { fit } { /Fit }
2852         { fitb } { /FitB }
2853         { fitbh } { /FitBH }
2854         { fitbv } { /FitBV ~ @xpos }
2855         { fith } { /FitH ~ @ypos }
2856         { fitv } { /FitV ~ @xpos }
2857       }
2858       { /XYZ ~ @xpos ~ @ypos ~ \\fp_eval:n { (#2) / 100 } }
2859     ]
2860   }
2861 }
2862 \cs_new_protected:Npn \\_pdf_backend_destination_rectangle:nn #1#2
2863 {
2864   \\group_begin:
2865   \\hbox_set:Nn \\l__pdf_internal_box {#2}
2866   \\box_move_down:nn { \\box_dp:N \\l__pdf_internal_box }
2867   {
2868     \\hbox:n
2869     {
2870       \\_pdf_backend:n { obj ~ @pdf_ #1 _llx ~ @xpos }
2871       \\_pdf_backend:n { obj ~ @pdf_ #1 _lly ~ @ypos }
2872     }
2873   }
2874   \\box_use:N \\l__pdf_internal_box
2875   \\box_move_up:nn { \\box_ht:N \\l__pdf_internal_box }
2876   {
2877     \\hbox:n
2878     {
2879       \\_pdf_backend:n
2880       {
2881         dest ~ (#1)
2882         [
2883           @thispage
2884           /FitR ~
2885           @pdf_ #1 _llx ~ @pdf_ #1 _lly ~
2886           @xpos ~ @ypos
2887         ]
2888       }
2889     }
2890   }
2891   \\group_end:

```



```
2892 }
(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
2893 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2894 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2895 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2896 {
2897   \bool_if:nF {#1}
2898   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2899 }
(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
2900 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2901 {
2902   \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2903   \__kernel_backend_literal:x { pdf:majorversion~ \_pdf_backend_version_major: }
2904 }
2905 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2906 {
2907   \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2908   \__kernel_backend_literal:x { pdf:minorversion~ \_pdf_backend_version_minor: }
2909 }
(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:
2910 \cs_new:Npn \_pdf_backend_version_major: { 1 }
2911 \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:
2912 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2913 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2914 \cs_new_protected:Npn \_pdf_backend_emc:
2915 { \__kernel_backend_literal_page:n { EMC } }
(End definition for \_pdf_backend_bdc:nn and \_pdf_backend_emc:.)
2916 </dvipdfmx | xdvipdfmx>
```

6.5 dvisvgm backend

2917 $\langle *dvisvgm \rangle$

6.5.1 Catalogue entries

$\backslash_pdf_backend_catalog_gput:nn$
 $\backslash_pdf_backend_info_gput:nn$

No-op.

2918 $\backslash cs_new_protected:Npn \backslash_pdf_backend_catalog_gput:nn \#1\#2 \{ \}$
2919 $\backslash cs_new_protected:Npn \backslash_pdf_backend_info_gput:nn \#1\#2 \{ \}$

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

6.5.2 Objects

All no-ops here.

$\backslash_pdf_backend_object_new:nn$
 $\backslash_pdf_backend_object_ref:n$
 $\backslash_pdf_backend_object_write:nn$
 $\backslash_pdf_backend_object_write:nx$
 $\backslash_pdf_backend_object_now:nn$
 $\backslash_pdf_backend_object_now:nx$
 $\backslash_pdf_backend_object_last:$

2920 $\backslash cs_new_protected:Npn \backslash_pdf_backend_object_new:nn \#1\#2 \{ \}$
2921 $\backslash cs_new:Npn \backslash_pdf_backend_object_ref:n \#1 \{ \}$
2922 $\backslash cs_new_protected:Npn \backslash_pdf_backend_object_write:nn \#1\#2 \{ \}$
2923 $\backslash cs_new_protected:Npn \backslash_pdf_backend_object_write:nx \#1\#2 \{ \}$
2924 $\backslash cs_new_protected:Npn \backslash_pdf_backend_object_now:nn \#1\#2 \{ \}$
2925 $\backslash cs_new_protected:Npn \backslash_pdf_backend_object_now:nx \#1\#2 \{ \}$
2926 $\backslash cs_new:Npn \backslash_pdf_backend_object_last: \{ \}$

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

6.5.3 Structure

These are all no-ops.

$\backslash_pdf_backend_compresslevel:n$
 $\backslash_pdf_backend_compress_objects:n$

2927 $\backslash cs_new_protected:Npn \backslash_pdf_backend_compresslevel:n \#1 \{ \}$
2928 $\backslash cs_new_protected:Npn \backslash_pdf_backend_compress_objects:n \#1 \{ \}$

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

Data not available!

$\backslash_pdf_backend_version_major_gset:n$
 $\backslash_pdf_backend_version_minor_gset:n$

2929 $\backslash cs_new_protected:Npn \backslash_pdf_backend_version_major_gset:n \#1 \{ \}$
2930 $\backslash cs_new_protected:Npn \backslash_pdf_backend_version_minor_gset:n \#1 \{ \}$

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

Data not available!

$\backslash_pdf_backend_version_major:$
 $\backslash_pdf_backend_version_minor:$

2931 $\backslash cs_new:Npn \backslash_pdf_backend_version_major: \{ -1 \}$
2932 $\backslash cs_new:Npn \backslash_pdf_backend_version_minor: \{ -1 \}$

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

More no-ops.

$\backslash_pdf_backend_bdc:nn$
 $\backslash_pdf_backend_emc:$

2933 $\backslash cs_new_protected:Npn \backslash_pdf_backend_bdc:nn \#1\#2 \{ \}$
2934 $\backslash cs_new_protected:Npn \backslash_pdf_backend_emc: \{ \}$

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

2935 $\langle /dvisvgm \rangle$

2936 $\langle /initex | package \rangle$

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, 2779
 - `\AtBeginDvi` 42, 43
- B**
- `\begin` 1370, 1375
 - bool commands:
 - `\bool_gset_false:N` 582, 598, 624, 646, 662, 814, 1140, 1176, 2226, 2277, 2786
 - `\bool_gset_true:N` 580, 649, 812, 1155, 2219, 2225, 2784
 - `\bool_if:NTF` 589, 593, 611, 615, 619, 632, 637, 641, 653, 657, 825, 830, 835, 1114, 1159, 1346, 1387, 1503, 1545, 2214, 2229, 2234, 2239, 2793
 - `\bool_if:nTF` 2650, 2897
 - `\bool_lazy_or:nnTF` 1379, 1538
 - `\bool_new:N` 583, 650, 815, 1156, 2202, 2203, 2777
 - `\bool_set_false:N` 1356, 1470, 1563, 1633
 - box commands:
 - `\box_dp:N` 140, 142, 190, 192, 247, 249, 296, 298, 300, 302, 2256, 2289, 2290, 2315, 2417, 2529, 2866
 - `\box_ht:N` 142, 192, 249, 300, 302, 1399, 1600, 2261, 2300, 2301, 2322, 2421, 2528, 2875
 - `\box_if_empty:N` 2356
 - `\box_move_down:nn` 2177, 2256, 2416, 2866
 - `\box_move_up:nn` 2180, 2261, 2420, 2795, 2875
 - `\box_new:N` 1668, 2158, 2159
 - `\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, 2207, 2419, 2530, 2874
 - `\box_wd:N` .. 141, 149, 191, 197, 248, 254, 297, 299, 335, 1398, 1599, 2527
 - 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, 1672, 1709, 1754, 2600, 2631, 2717, 2739, 2774
 - `\cs_gset:Npx` 2902, 2907
 - `\cs_if_exist:N` 42, 67, 75, 83, 89, 95, 508, 517, 906, 914, 2352, 2450, 2461, 2472, 2480, 2487, 2494, 2501, 2522, 2535, 2542, 2560, 2596, 2627, 2635, 2643, 2657, 2664, 2673, 2683, 2690, 2697, 2705, 2781
 - `\cs_new:Npn` 675, 850, 1187, 1603, 1612, 1662, 1687, 1755, 2197, 2378, 2440, 2441, 2568, 2601, 2602, 2732, 2775, 2819, 2838, 2910, 2911, 2921, 2926, 2931, 2932
 - `\cs_new:Npx` 2458, 2484, 2632, 2688, 2702
 - `\cs_new_eq:NN` 25, 527, 774, 780, 781, 921, 1030,

1323, 1352, 1409, 1410, 1458, 1466,
1488, 1559, 1622, 1629, 1661, 2207

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,
 1623, 1630, 1636, 1638, 1640, 1651,
 1670, 1673, 1675, 1679, 1689, 1710,
 1715, 1720, 1725, 1734, 2161, 2175,
 2206, 2208, 2210, 2212, 2217, 2232,
 2237, 2279, 2308, 2332, 2341, 2380,
 2387, 2412, 2436, 2437, 2438, 2439,
 2442, 2444, 2466, 2468, 2573, 2606,
 2648, 2709, 2711, 2718, 2720, 2724,
 2734, 2740, 2745, 2750, 2755, 2757,
 2759, 2767, 2791, 2806, 2821, 2823,
 2825, 2836, 2839, 2841, 2862, 2893,
 2895, 2900, 2905, 2912, 2914, 2918,
 2919, 2920, 2922, 2923, 2924, 2925,
 2927, 2928, 2929, 2930, 2933, 2934

\cs_new_protected:Npx
 36, 65, 73, 81, 87,
 93, 506, 515, 904, 912, 2448, 2470,
 2478, 2492, 2499, 2518, 2533, 2540,
 2551, 2640, 2654, 2662, 2680, 2715

\cs_set_eq:NN 2373, 2374

\cs_set_protected:Npn
 381, 439, 2549, 2571, 2604

\dim_eval:n 2184, 2453, 2454,
 2455, 2497, 2813, 2814, 2815, 2840
 \dim_max:nn 2287, 2298
 \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, 1656, 1657, 2285, 2296, 2314,
 2315, 2317, 2320, 2322, 2326, 2384
 \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, 1645, 1646, 1647, 1648

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	2785
\evensidemargin	2253
exp commands:	
\exp_after:wN	388, 1610
\exp_args:Nf	669, 844
\exp_args:NNf	152, 200, 257
\exp_args:Nnx	1751, 2770
\exp_args:NV	383
\exp_args:Nx	489, 1422, 1443, 1722, 2245, 2736
\exp_last_unbraced:Nx	392, 441
\exp_not:N	43, 70, 79, 98, 511, 512, 520, 909, 910, 917, 2453, 2454, 2455, 2460, 2462, 2463, 2486, 2488, 2489, 2495, 2496, 2497, 2502, 2503, 2505, 2515, 2523, 2524, 2527, 2528, 2529, 2555, 2557, 2564, 2634, 2636, 2637, 2642, 2646, 2656, 2668, 2669, 2675, 2676, 2682, 2684, 2686, 2693, 2698, 2704, 2706
\exp_not:n	27, 70, 79, 98, 1713, 1718, 2408, 2581, 2582, 2601, 2602, 2612, 2613, 2748, 2753, 2764, 2845

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, 2405, 2515, 2858
 \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 2317
 \l_galley_total_left_margin_dim 2248

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, 1632
 \graphics_extract_bb:n
 1464, 1471, 1627, 1634
 \l_graphics_interpolate_bool ...
 1346, 1356, 1380, 1387,
 1470, 1503, 1539, 1545, 1563, 1633
 \l_graphics_llx_dim
 1329, 1480, 1532, 1645
 \l_graphics_lly_dim
 1330, 1481, 1533, 1646
 \l_graphics_name_tl 1436
 \l_graphics_page_int
 1340, 1360, 1361, 1391,
 1392, 1462, 1499, 1500, 1526, 1527,
 1555, 1568, 1569, 1608, 1609, 1625
 \l_graphics_pagebox_tl
 41, 1341, 1359,

1393, 1394, 1463, 1497, 1498, 1528,
 1530, 1556, 1577, 1578, 1610, 1626

\graphics_read_bb:n . 1323, 1458, 1622
 \l_graphics_urx_dim
 .. 1331, 1398, 1482, 1534, 1599, 1647
 \l_graphics_ury_dim .. 1332, 1399,
 1483, 1535, 1600, 1648, 1656, 1657

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, 1616
 __graphics_backend_getbb_eps:nm
 1411
 __graphics_backend_getbb_eps:nn
 1422, 1429
 __graphics_backend_getbb_jpg:n .
 1338, 1452, 1553, 1623
 __graphics_backend_getbb_-
 pagebox:w 1553, 1610
 __graphics_backend_getbb_pdf:n .
 1338, 1437, 1452, 1553, 1630
 __graphics_backend_getbb_png:n .
 1338, 1452, 1553, 1623
 __graphics_backend_include:nn 1636
 __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, 1651
 __graphics_backend_include_-
 eps:n 1324, 1411, 1475, 1636
 __graphics_backend_include_-
 jpg:n 1404, 1475, 1651

<code>__graphics_backend_include_-pdf:n</code> .. 1404 , 1443 , 1475 , 1604 , 1636	<code>\int_set_eq:NN</code> 1041 , 2346
<code>__graphics_backend_include_pdf_-quote:w</code> 1607 , 1612	<code>\int_use:N</code> 289 , 320 , 1119 , 1125 , 1132 , 1164 , 1172 , 1361 , 1392 , 1407 , 1500 , 1513 , 1525 , 1527 , 1609 , 1688 , 1739 , 1752 , 1756 , 2169 , 2192 , 2198 , 2271 , 2379 , 2569 , 2576 , 2733 , 2771 , 2776 , 2812 , 2820
<code>__graphics_backend_include_-png:n</code> 1404 , 1475 , 1651	<code>\int_value:w</code> .. 2460 , 2486 , 2634 , 2646
<code>\l__graphics_backend_name_str</code> . 1411	<code>\int_zero:N</code> ... 1340 , 1462 , 1555 , 1625
<code>\l__graphics_graphics_attr_tl</code> 1337 , 1342 , 1349 , 1357 , 1367 , 1400 , 1402 , 1407	
<code>\l__graphics_internal_box</code> 1396 , 1398 , 1399 , 1598 , 1599 , 1600	K
<code>\g__graphics_track_int</code> 1474 , 1520 , 1521	kernel internal commands:
group commands:	<code>__kernel_backend_align_begin:</code> 48 , 132 , 156 , 171
<code>\group_begin:</code> 1044 , 2414 , 2520 , 2548 , 2570 , 2603 , 2864	<code>__kernel_backend_align_end:</code> 48 , 146 , 164 , 178
<code>\group_end:</code> 1052 , 2434 , 2531 , 2567 , 2599 , 2630 , 2891	<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 , 1642 , 1653 , 2716 , 2840 , 2894 , 2898 , 2903 , 2908
<code>\group_insert_after:N</code> 426 , 512 , 737 , 910	<code>__kernel_backend_literal_page:n</code> 73 , 106 , 2710 , 2712 , 2913 , 2915
H	<code>__kernel_backend_literal_pdf:n</code> 65 , 103 , 187 , 244 , 774 , 921
hbox commands:	<code>__kernel_backend_literal_-postscript:n</code> ... 30 , 51 , 52 , 56 , 133 , 134 , 136 , 137 , 145 , 157 , 172 , 527
<code>\hbox:n</code> 2178 , 2179 , 2182 , 2257 , 2263 , 2418 , 2422 , 2868 , 2877	<code>__kernel_backend_literal_svg:n</code> 114 , 118 , 120 , 122 , 288 , 290 , 307 , 1030 , 1298 , 1309
<code>\hbox_overlap_right:n</code> 147 , 179 , 195 , 236 , 252 , 280 , 367 , 765 , 1022	<code>__kernel_backend_matrix:n</code> 93 , 209 , 230 , 927
<code>\hbox_set:Nn</code> 1396 , 1598 , 2244 , 2281 , 2415 , 2521 , 2865	<code>__kernel_backend_postscript:n</code> 33 , 424 , 730 , 1671 , 1727 , 2178 , 2185 , 2220 , 2257 , 2264 , 2268 , 2282 , 2310 , 2360 , 2367 , 2373 , 2382 , 2389 , 2418 , 2422
<code>\hbox_set:Nw</code> 2227	<code>__kernel_backend_postscript_-header:n</code> 36 , 1757 , 1763 , 1770 , 1776 , 1815 , 1853 , 1994 , 2101
<code>\hbox_set_end:</code> 2242	<code>__kernel_backend_scope_begin:</code> 5 , 59 , 81 , 108 , 117 , 131 , 155 , 170 , 186 , 203 , 229 , 243 , 260 , 273 , 780 , 1009 , 1296
<code>\hbox_unpack:N</code> 2374	<code>__kernel_backend_scope_begin:n</code> 121 , 309 , 317 , 322 , 340 , 353
I	<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
int commands:	<code>\l__kernel_color_stack_int</code> 471 , 511 , 520 , 909 , 917
<code>\int_compare:nNnTF</code> 1360 , 1391 , 1499 , 1526 , 1568 , 1608 , 2345 , 2666 , 2692	
<code>\int_const:Nn</code> 1400 , 1521 , 1682 , 2554 , 2727	
<code>\int_eval:n</code> 2646 , 2669 , 2676 , 2686 , 2894 , 2902 , 2907	
<code>\int_gincr:N</code> 287 , 1060 , 1116 , 1161 , 1520 , 1681 , 1736 , 2164 , 2188 , 2266 , 2726 , 2769 , 2808	
<code>\int_gset:Nn</code> 2334	
<code>\int_gset_eq:NN</code> 1053 , 2165 , 2189 , 2267 , 2809	
<code>\int_gzero:N</code> 1045	
<code>\int_if_exist:NTF</code> 1510	
<code>\int_if_odd:nTF</code> 2251	
<code>\int_new:N</code> 337 , 471 , 1063 , 1064 , 1142 , 1474 , 1677 , 2160 , 2199 , 2201 , 2722 , 2790	

L	
<code>\landscape</code>	2781, 2783
M	
math commands:	
<code>\c_math_toggle_token</code>	2230, 2240
mode commands:	
<code>\mode_if_horizontal:TF</code> ...	2336, 2343
<code>\mode_if_math:TF</code>	2224
O	
<code>\oddsidemargin</code>	2252
P	
pdf internal commands:	
<code>__pdf_backend:n</code>	2715, 2719, 2721, 2747, 2752, 2761, 2810, 2827, 2837, 2843, 2870, 2871, 2879
<code>__pdf_backend_annotation:nnnn</code> ..	2161, 2448, 2791
<code>__pdf_backend_annotation_-</code> <code>aux:nnnn</code>	2161, 2791
<code>\g__pdf_backend_annotation_int</code> ..	2160, 2165, 2189, 2198, 2790, 2809, 2820
<code>__pdf_backend_annotation_last:</code> ..	2197, 2458, 2819
<code>__pdf_backend_bdc:nn</code>	2442, 2709, 2912, 2933
<code>__pdf_backend_catalog_gput:nn</code> ..	1673, 2533, 2718, 2918
<code>__pdf_backend_compress_objects:n</code>	2436, 2640, 2893, 2927
<code>__pdf_backend_compresslevel:n</code> ..	2436, 2640, 2893, 2927
<code>\l__pdf_backend_content_box</code> ..	2158, 2227, 2256, 2259, 2261, 2290, 2301
<code>__pdf_backend_destination:nn</code> ...	2387, 2499, 2841
<code>__pdf_backend_destination_-</code> <code>rectangle:nn</code>	2387, 2499, 2841
<code>__pdf_backend_emc:</code>	2442, 2709, 2912, 2933
<code>__pdf_backend_info_gput:nn</code>	1673, 2533, 2718, 2918
<code>__pdf_backend_link:nw</code>	2208
<code>__pdf_backend_link_aux:nw</code> ...	2208
<code>__pdf_backend_link_begin:n</code> ..	2821
<code>__pdf_backend_link_begin:nnnw</code> ..	2466
<code>__pdf_backend_link_begin:nw</code> ...	2209, 2211, 2212
<code>__pdf_backend_link_begin_aux:nw</code>	2215, 2217
<code>__pdf_backend_link_begin_-</code> <code>goto:nnw</code>	2208, 2466, 2821
<code>__pdf_backend_link_begin_-</code> <code>user:nnw</code>	2208, 2466, 2821
<code>\g__pdf_backend_link_bool</code>	2203, 2214, 2219, 2234, 2277
<code>\g__pdf_backend_link_dict_tl</code> ...	2200, 2222, 2272
<code>__pdf_backend_link_end:</code>	2208, 2466, 2821
<code>__pdf_backend_link_end_aux:</code> ..	2208
<code>\g__pdf_backend_link_int</code>	2199, 2267, 2271, 2379
<code>__pdf_backend_link_last:</code>	2378, 2484, 2838
<code>__pdf_backend_link_margin:n</code> ...	2380, 2492, 2839
<code>\g__pdf_backend_link_math_bool</code> ..	2202, 2225, 2226, 2229, 2239
<code>__pdf_backend_link_minima:</code> ..	2208
<code>__pdf_backend_link_outerbox:n</code> ..	2208
<code>\g__pdf_backend_link_sf_int</code>	2201, 2334, 2345, 2346
<code>__pdf_backend_link_sf_restore:</code> ..	2208
<code>__pdf_backend_link_sf_save:</code> ..	2208
<code>\l__pdf_backend_model_box</code> ..	2159, 2244, 2281, 2289, 2300, 2315, 2322
<code>__pdf_backend_objcompresslevel:n</code>	2640
<code>\g__pdf_backend_object_int</code>	1677, 1681, 1684, 1736, 1739, 1752, 1756, 2164, 2165, 2169, 2188, 2189, 2192, 2266, 2267, 2722, 2726, 2729, 2769, 2771, 2776, 2808, 2809, 2812
<code>__pdf_backend_object_last:</code>	1755, 2632, 2775, 2920
<code>__pdf_backend_object_new:nn</code> ...	1679, 2548, 2724, 2920
<code>__pdf_backend_object_now:nn</code> ...	1734, 2603, 2767, 2920
<code>\g__pdf_backend_object_prop</code>	1677, 1685, 1696, 1706, 2547, 2557, 2579, 2722, 2730, 2737
<code>__pdf_backend_object_ref:n</code> ..	1679, 1693, 1707, 2548, 2724, 2743, 2920
<code>__pdf_backend_object_write:nn</code> ..	1689, 2570, 2734, 2920
<code>__pdf_backend_object_write:nnn</code> ..	2734
<code>__pdf_backend_object_write_-</code> <code>array:nn</code>	1689, 2734
<code>__pdf_backend_object_write_-</code> <code>dict:nn</code>	1689, 2734
<code>__pdf_backend_object_write_-</code> <code>fstream:nn</code>	2734
<code>__pdf_backend_object_write_-</code> <code>stream:nn</code>	1689, 2734

<code>\str_tail:N</code>	1424, 1445	<code>\tex_pdfstartlink:D</code>	2474
sys commands:		<code>\tex_pdfvariable:D</code>	
<code>\sys_if_shell:TF</code>	1411	2494, 2495, 2645, 2659,
<code>\sys_shell_now:n</code>	1433	2664, 2668, 2685, 2690, 2693, 2707	
T			
T _E X and L ^A T _E X 2 _ε commands:		<code>\tex_pdfximage:D</code>	1378
<code>\ccclv</code>	2356, 2358, 2366	<code>\tex_pdfximagebbox:D</code>	1372
<code>\@ifpackageloaded</code>	379, 437	<code>\tex_spacefactor:D</code>	2337, 2346
<code>\@makecol@hook</code>	2349	<code>\tex_special:D</code>	25
<code>\current@color</code>	13, 383, 388, 393, 442	<code>\tex_the:D</code>	1401, 2693, 2698, 2704
<code>\special</code>	1	<code>\tex_XeTeXpdffile:D</code>	1564, 1606
tex commands:		<code>\tex_XeTeXpicfile:D</code>	1557
<code>\tex_baselineskip:D</code>	2326	<code>\textwidth</code>	2320
<code>\tex_global:D</code>		tl commands:	
.	2642, 2656, 2668, 2675, 2682	<code>\c_space_tl</code>	
<code>\tex_immediate:D</code>	1378, 2575, 2608	211, 216, 219, 388, 1101, 1328,
<code>\tex_kern:D</code>	2184	1329, 1330, 1331, 1479, 1480, 1481,	
<code>\tex_luatexversion:D</code>	2666, 2692	1482, 1527, 1530, 1532, 1533, 1534,	
<code>\tex_pdfannot:D</code>	2452	1535, 1607, 1609, 1644, 1645, 1646,	
<code>\tex_pdfcatalog:D</code>	2537	1647, 2272, 2464, 2490, 2638, 2812	
<code>\tex_pdfcolorstack:D</code>	510, 519, 908, 916	<code>\tl_clear:N</code>	1341, 1349, 1355,
<code>\tex_pdfcompresslevel:D</code>	2643, 2644	1463, 1469, 1556, 1562, 1626, 1632	
<code>\tex_pdfdest:D</code>	2503, 2524	<code>\tl_gclear:N</code>	1139, 1175
<code>\tex_pdfendlink:D</code>	2482	<code>\tl_gset:Nn</code>	1098, 2222
<code>\tex_pdfextension:D</code>	67, 68, 75, 76,	<code>\tl_if_empty:N</code>	1101, 1344, 1385,
83, 84, 89, 90, 95, 96, 508, 509, 517,		1393, 1497, 1501, 1528, 1543, 1577	
518, 906, 907, 914, 915, 2450, 2451,		<code>\tl_if_empty:nTF</code>	1195
2472, 2473, 2480, 2481, 2501, 2502,		<code>\tl_if_empty_p:N</code>	1381, 1540
2522, 2523, 2535, 2536, 2542, 2543,		<code>\tl_if_head_is_space:nTF</code>	383
2560, 2563, 2596, 2597, 2627, 2628		<code>\tl_new:N</code>	1105, 1337, 2200, 2204
<code>\tex_pdffeedback:D</code>	2461,	<code>\tl_put_left:Nn</code>	2785
2462, 2487, 2488, 2564, 2635, 2636		<code>\tl_put_right:Nn</code>	2354, 2783
<code>\tex_pdfinfo:D</code>	2544	<code>\tl_set:Nn</code>	385, 397, 448, 451, 454,
<code>\tex_pdflastannot:D</code>	2463	458, 461, 1342, 1357, 1436, 2205, 2372	
<code>\tex_pdflastlink:D</code>	2489	<code>\tl_to_str:n</code>	1683,
<code>\tex_pdflastobj:D</code>	2566, 2637	1688, 2555, 2569, 2577, 2728, 2733	
<code>\tex_pdflastximage:D</code>	1397, 1401	U	
<code>\tex_pdflinkmargin:D</code>	2496	use commands:	
<code>\tex_pdfliteral:D</code>	69, 77	<code>\use:N</code>	1705, 1751, 2742, 2770
<code>\tex_pdfmajorversion:D</code>		<code>\use:n</code>	44, 388, 474, 494,
.	2673, 2675, 2697, 2698	669, 844, 966, 978, 990, 1180, 1220,	
<code>\tex_pdfminorversion:D</code>		1239, 1251, 1318, 1453, 1584, 1617	
.	2683, 2684, 2705, 2706	<code>\use_none:n</code>	458, 1195, 1197, 2350
<code>\tex_pdfobj:D</code>	2566, 2598, 2629	V	
<code>\tex_pdfobjcompresslevel:D</code>	2657, 2658	<code>\value</code>	2251
<code>\tex_pdfrefximage:D</code>	1397, 1406	vbox commands:	
<code>\tex_pdfrestore:D</code>	91	<code>\vbox:n</code>	2797
<code>\tex_pdfsave:D</code>	85	<code>\vbox_set:Nn</code>	2358
<code>\tex_pdfsetmatrix:D</code>	97	<code>\vbox_unpack_drop:N</code>	2366