

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}{2020-02-03}{ }
6 {L3 backend support: dvipdfmx}
7 </dvipdfmx>
8 <*dvips>
9 {l3backend-dvips.def}{2020-02-03}{ }
10 {L3 backend support: dvips}
11 </dvips>
12 <*dvisvgm>
13 {l3backend-dvisvgm.def}{2020-02-03}{ }
14 {L3 backend support: dvisvgm}
15 </dvisvgm>
16 <*pdfmode>
17 {l3backend-pdfmode.def}{2020-02-03}{ }
18 {L3 backend support: PDF mode}
19 </pdfmode>
20 <*xdvipdfmx>
21 {l3backend-xdvipdfmx.def}{2020-02-03}{ }
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`
`_kernel_backend_literal_postscript:x` 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`
`_kernel_backend_postscript:x` 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`.)

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_if_exist:NTF \AtBeginDvi
37   { \exp_not:N \AtBeginDvi }
38   { \use:n }
39   { \_kernel_backend_literal:n { header = l3backend-dvips.pro } }
```

`_kernel_backend_align_begin:`
`_kernel_backend_align_end:` In `dvips` there is no built-in saving of the current position, and so some additional PostScript is required to set up the transformation matrix and also to restore it afterwards. Notice the use of the stack to save the current position “up front” and to move back to it at the end of the process. Notice that the `[begin]`/`[end]` pair here mean that we can use a run of PostScript statements in separate lines: not *required* but does make the code and output more clear.

```
40 \cs_new_protected:Npn \_kernel_backend_align_begin:
41   {
42     \_kernel_backend_literal:n { ps::[begin] }
43     \_kernel_backend_literal_postscript:n { currentpoint }
44     \_kernel_backend_literal_postscript:n { currentpoint~translate }
45   }
46 \cs_new_protected:Npn \_kernel_backend_align_end:
47   {
48     \_kernel_backend_literal_postscript:n { neg~exch~neg~exch~translate }
49     \_kernel_backend_literal:n { ps::[end] }
50   }
```

(End definition for `_kernel_backend_align_begin:` and `_kernel_backend_align_end:.`)

`_kernel_backend_scope_begin:`
`_kernel_backend_scope_end:` Saving/restoring scope for general operations needs to be done with `dvips` positioning (try without to see this!). Thus we need the `ps:` version of the special here. As only the graphics state is ever altered within this pairing, we use the lower-cost `g`-versions.

```
51 \cs_new_protected:Npn \_kernel_backend_scope_begin:
52   { \_kernel_backend_literal:n { ps:gsave } }
53 \cs_new_protected:Npn \_kernel_backend_scope_end:
54   { \_kernel_backend_literal:n { ps:grestore } }
```

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

55 `</dvips>`

1.2 pdfmode backend

56 `<*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` 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).

`_kernel_backend_literal_pdf:x`

```
57 \cs_new_protected:Npx \_kernel_backend_literal_pdf:n #1
58 {
59   \cs_if_exist:NTF \tex_pdfextension:D
60   { \tex_pdfextension:D literal }
61   { \tex_pdfliteral:D }
62   { \exp_not:N \exp_not:n {#1} }
63 }
64 \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.

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

(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:`

```
73 \cs_new_protected:Npx \_kernel_backend_scope_begin:
74 {
75   \cs_if_exist:NTF \tex_pdfextension:D
76   { \tex_pdfextension:D save \scan_stop: }
77   { \tex_pdfsave:D }
78 }
79 \cs_new_protected:Npx \_kernel_backend_scope_end:
80 {
81   \cs_if_exist:NTF \tex_pdfextension:D
82   { \tex_pdfextension:D restore \scan_stop: }
83   { \tex_pdfrestore:D }
84 }
```

(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.

```

85 \cs_new_protected:Npx \_kernel_backend_matrix:n #1
86 {
87   \cs_if_exist:NTF \tex_pdfextension:D
88     { \tex_pdfextension:D setmatrix }
89     { \tex_pdfsetmatrix:D }
90     { \exp_not:N \exp_not:n {#1} }
91 }
92 \cs_generate_variant:Nn \_kernel_backend_matrix:n { x }

```

(End definition for `_kernel_backend_matrix:n`.)

```

93 </pdfmode>

```

1.3 dvipdfmx backend

```

94 <*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.

```

95 \cs_new_protected:Npn \_kernel_backend_literal_pdf:n #1
96 { \_kernel_backend_literal:n { pdf:literal~ #1 } }
97 \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!

```

98 \cs_new_protected:Npn \_kernel_backend_literal_page:n #1
99 { \_kernel_backend_literal:n { pdf:literal~direct~ #1 } }

```

(End definition for `_kernel_backend_literal_page:n`.)

`_kernel_backend_scope_begin:` Scoping is done using the backend-specific specials.

```

\_kernel_backend_scope_end:
100 \cs_new_protected:Npn \_kernel_backend_scope_begin:
101 { \_kernel_backend_literal:n { x:gsave } }
102 \cs_new_protected:Npn \_kernel_backend_scope_end:
103 { \_kernel_backend_literal:n { x:grestore } }

```

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

```

104 </dvipdfmx | xdvipdfmx>

```

1.4 dvisvgm backend

105 $\langle *dvisvgm \rangle$

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

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.

106 $\backslash cs_new_protected:Npn \backslash_kernel_backend_literal_svg:n \#1$
 107 $\{ \backslash_kernel_backend_literal:n \{ dvisvgm:raw~ \#1 \{ ?nl \} \} \}$
 108 $\backslash cs_generate_variant:Nn \backslash_kernel_backend_literal_svg:n \{ x \}$

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

$\backslash_kernel_backend_scope_begin:$
 $\backslash_kernel_backend_scope_end:$

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

109 $\backslash cs_new_protected:Npn \backslash_kernel_backend_scope_begin:$
 110 $\{ \backslash_kernel_backend_literal_svg:n \{ <g> \} \}$
 111 $\backslash cs_new_protected:Npn \backslash_kernel_backend_scope_end:$
 112 $\{ \backslash_kernel_backend_literal_svg:n \{ </g> \} \}$

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

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

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

113 $\backslash cs_new_protected:Npn \backslash_kernel_backend_scope_begin:n \#1$
 114 $\{ \backslash_kernel_backend_literal_svg:n \{ <g~ \#1 > \} \}$
 115 $\backslash cs_generate_variant:Nn \backslash_kernel_backend_scope_begin:n \{ x \}$

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

116 $\langle /dvisvgm \rangle$

117 $\langle /initex | package \rangle$

2 l3backend-box Implementation

118 $\langle *initex | package \rangle$

119 $\langle @@=box \rangle$

2.1 dvips backend

120 $\langle *dvips \rangle$

$\backslash_box_backend_clip:N$

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.

121 $\backslash cs_new_protected:Npn \backslash_box_backend_clip:N \#1$
 122 $\{$
 123 $\backslash_kernel_backend_scope_begin:$
 124 $\backslash_kernel_backend_align_begin:$
 125 $\backslash_kernel_backend_literal_postscript:n \{ matrix-currentmatrix \}$
 126 $\backslash_kernel_backend_literal_postscript:n$

```

127     { Resolution~72~div~VResolution~72~div~scale }
128   \__kernel_backend_literal_postscript:n { DVImag-dup~scale }
129   \__kernel_backend_literal_postscript:x
130   {
131     0 ~
132     \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
133     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
134     \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
135     rectclip
136   }
137   \__kernel_backend_literal_postscript:n { setmatrix }
138   \__kernel_backend_align_end:
139   \hbox_overlap_right:n { \box_use:N #1 }
140   \__kernel_backend_scope_end:
141   \skip_horizontal:n { \box_wd:N #1 }
142 }

```

(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.

```

143 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
144 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
145 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
146 {
147   \__kernel_backend_scope_begin:
148   \__kernel_backend_align_begin:
149   \__kernel_backend_literal_postscript:x
150   {
151     \fp_compare:nNnTF {#2} = \c_zero_fp
152     { 0 }
153     { \fp_eval:n { round ( -(#2) , 5 ) } } ~
154     rotate
155   }
156   \__kernel_backend_align_end:
157   \box_use:N #1
158   \__kernel_backend_scope_end:
159 }

```

(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.

```

160 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
161 {
162   \__kernel_backend_scope_begin:
163   \__kernel_backend_align_begin:
164   \__kernel_backend_literal_postscript:x
165   {
166     \fp_eval:n { round ( #2 , 5 ) } ~
167     \fp_eval:n { round ( #3 , 5 ) } ~
168     scale
169   }
170   \__kernel_backend_align_end:

```

```

171 \hbox_overlap_right:n { \box_use:N #1 }
172 \__kernel_backend_scope_end:
173 }

```

(End definition for __box_backend_scale:Nnn.)

```

174 </dvips>

```

2.2 pdfmode backend

```

175 <*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.

```

176 \cs_new_protected:Npn \__box_backend_clip:N #1
177 {
178   \__kernel_backend_scope_begin:
179   \__kernel_backend_literal_pdf:x
180   {
181     0~
182     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
183     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
184     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
185     re~W~n
186   }
187   \hbox_overlap_right:n { \box_use:N #1 }
188   \__kernel_backend_scope_end:
189   \skip_horizontal:n { \box_wd:N #1 }
190 }

```

(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.

```

191 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
192 { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
193 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
194 {
195   \__kernel_backend_scope_begin:
196   \box_set_wd:Nn #1 { Opt }
197   \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
198   \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
199     { \fp_zero:N \l__box_backend_cos_fp }
200   \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
201   \__kernel_backend_matrix:x
202   {
203     \fp_use:N \l__box_backend_cos_fp \c_space_tl

```

```

204 \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp
205 { 0~0 }
206 {
207   \fp_use:N \l__box_backend_sin_fp
208   \c_space_tl
209   \fp_eval:n { -\l__box_backend_sin_fp }
210 }
211 \c_space_tl
212 \fp_use:N \l__box_backend_cos_fp
213 }
214 \box_use:N #1
215 \__kernel_backend_scope_end:
216 }
217 \fp_new:N \l__box_backend_cos_fp
218 \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.

```

219 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
220 {
221   \__kernel_backend_scope_begin:
222   \__kernel_backend_matrix:x
223   {
224     \fp_eval:n { round ( #2 , 5 ) } ~
225     0~0~
226     \fp_eval:n { round ( #3 , 5 ) }
227   }
228   \hbox_overlap_right:n { \box_use:N #1 }
229   \__kernel_backend_scope_end:
230 }

```

(End definition for __box_backend_scale:Nnn.)

231 </pdfmode>

2.3 dvipdfmx backend

232 <*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.

```

233 \cs_new_protected:Npn \__box_backend_clip:N #1
234 {
235   \__kernel_backend_scope_begin:
236   \__kernel_backend_literal_pdf:x
237   {
238     0~
239     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
240     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
241     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
242     re~W~n
243   }
244   \hbox_overlap_right:n { \box_use:N #1 }
245   \__kernel_backend_scope_end:

```



```

246     \skip_horizontal:n { \box_wd:N #1 }
247 }

```

(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.

```

248 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
249 { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
250 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
251 {
252   \__kernel_backend_scope_begin:
253   \__kernel_backend_literal:x
254   {
255     x:rotate~
256     \fp_compare:nNnTF {#2} = \c_zero_fp
257     { 0 }
258     { \fp_eval:n { round ( #2 , 5 ) } }
259   }
260   \box_use:N #1
261   \__kernel_backend_scope_end:
262 }

```

(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.

```

263 \cs_new_protected:Npn \_box_backend_scale:Nnn #1#2#3
264 {
265   \__kernel_backend_scope_begin:
266   \__kernel_backend_literal:x
267   {
268     x:scale~
269     \fp_eval:n { round ( #2 , 5 ) } ~
270     \fp_eval:n { round ( #3 , 5 ) }
271   }
272   \hbox_overlap_right:n { \box_use:N #1 }
273   \__kernel_backend_scope_end:
274 }

```

(End definition for `_box_backend_scale:Nnn`.)

```

275 </dvipdfmx | xdvipdfmx>

```

2.4 dvisvgm backend

```

276 <*dvisvgm>

```

`_box_backend_clip:N`
`\g__box_clip_path_int`

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

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

```

277 \cs_new_protected:Npn \__box_backend_clip:N #1
278 {
279   \int_gincr:N \g__box_clip_path_int
280   \__kernel_backend_literal_svg:x
281   { < clipPath~id = " l3cp \int_use:N \g__box_clip_path_int " > }
282   \__kernel_backend_literal_svg:x
283   {
284     <
285     path ~ d =
286     "
287       M ~ 0 ~
288       \dim_to_decimal:n { -\box_dp:N #1 } ~
289       L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
290       \dim_to_decimal:n { -\box_dp:N #1 } ~
291       L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
292       \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
293       L ~ 0 ~
294       \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
295       Z
296     "
297   />
298   }
299   \__kernel_backend_literal_svg:n
300   { < /clipPath > }

```

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

```

301   \__kernel_backend_scope_begin:n
302   {
303     transform =
304     "
305       translate ( { ?x } , { ?y } ) ~
306       scale ( 1 , -1 )
307     "
308   }
309   \__kernel_backend_scope_begin:x
310   {
311     clip-path =
312     "url ( \c_hash_str l3cp \int_use:N \g__box_clip_path_int ) "
313   }
314   \__kernel_backend_scope_begin:n
315   {
316     transform =
317     "
318       scale ( -1 , 1 ) ~
319       translate ( { ?x } , { ?y } ) ~
320       scale ( -1 , -1 )

```

```

321         "
322     }
323     \box_use:N #1
324     \__kernel_backend_scope_end:
325     \__kernel_backend_scope_end:
326     \__kernel_backend_scope_end:
327 %     \skip_horizontal:n { \box_wd:N #1 }
328 }
329 \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.

```

330 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
331 {
332     \__kernel_backend_scope_begin:x
333     {
334         transform =
335         "
336             rotate
337             ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
338         "
339     }
340     \box_use:N #1
341     \__kernel_backend_scope_end:
342 }

```

(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).

```

343 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
344 {
345     \__kernel_backend_scope_begin:x
346     {
347         transform =
348         "
349             translate ( { ?x } , { ?y } ) ~
350             scale
351             (
352                 \fp_eval:n { round ( -#2 , 5 ) } ,
353                 \fp_eval:n { round ( -#3 , 5 ) }
354             ) ~
355             translate ( { ?x } , { ?y } ) ~
356             scale ( -1 )
357         "
358     }
359     \hbox_overlap_right:n { \box_use:N #1 }
360     \__kernel_backend_scope_end:
361 }

```

(End definition for `_box_backend_scale:Nnn`.)

```
362 </dvisvgm>
363 </initex | package>
```

3 I3backend-color Implementation

```
364 <*initex | package>
365 <@@=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

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

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

```
367 <*package>
368 \cs_new_protected:Npn \_color_backend_pickup:N #1 { }
369 \AtBeginDocument
370 {
371   \cs_if_exist:cT { ver@color.sty }
372   {
373     \cs_set_protected:Npn \_color_backend_pickup:N #1
374     {
375       \exp_args:NV \tl_if_head_is_space:nTF \current@color
376       {
377         \tl_set:Nx #1
378         {
379           spot ~
380           \exp_after:wN \use:n \current@color \c_space_tl 1
381         }
382       }
383       {
384         \exp_last_unbraced:Nx \_color_backend_pickup:w
385         { \current@color } \q_stop #1
386       }
387     }
388     \cs_new_protected:Npn \_color_backend_pickup:w #1 ~ #2 \q_stop #3
389     { \tl_set:Nn #3 { #1 ~ #2 } }
390   }
391 }
392 </package>
```

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

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

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

```
393 \cs_new_protected:Npn \_color_backend_cmyk:nnnn #1#2#3#4
394 {
```

```

395     \_color_backend_select:x
396     {
397         cmyk~
398         \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
399         \fp_eval:n {#3} ~ \fp_eval:n {#4}
400     }
401 }
402 \cs_new_protected:Npn \_color_backend_gray:n #1
403 { \_color_backend_select:x { gray~ \fp_eval:n {#1} } }
404 \cs_new_protected:Npn \_color_backend_rgb:nnn #1#2#3
405 {
406     \_color_backend_select:x
407     { rgb~ \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} }
408 }
409 \cs_new_protected:Npn \_color_backend_spot:nn #1#2
410 { \_color_backend_select:n { #1 } }
411 \cs_new_protected:Npn \_color_backend_select:n #1
412 {
413     \_kernel_backend_literal:n { color-push~ #1 }
414     <*dvips>
415     \_kernel_backend_postscript:n { /color.fc~{ }~def }
416     </dvips>
417     \group_insert_after:N \_color_backend_reset:
418 }
419 \cs_generate_variant:Nn \_color_backend_select:n { x }
420 \cs_new_protected:Npn \_color_backend_reset:
421 { \_kernel_backend_literal:n { color~pop } }

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

422 </divisgm | dvipdfmx | dvips | xdvipdfmx>

```

3.2 pdfmode

```

423 <*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 color

```

424 <*package>
425 \cs_new_protected:Npn \_color_backend_pickup:N #1 { }
426 \AtBeginDocument
427 {
428     \cs_if_exist:cT { ver@color.sty }
429     {
430         \cs_set_protected:Npn \_color_backend_pickup:N #1
431         {
432             \exp_last_unbraced:Nx \_color_backend_pickup:w
433             { \current@color } ~ 0 ~ 0 ~ 0 \q_stop #1
434         }
435         \cs_new_protected:Npn \_color_backend_pickup:w
436         #1 ~ #2 ~ #3 ~ #4 ~ #5 ~ #6 \q_stop #7
437         {

```

```

438     \str_if_eq:nnTF {#2} { g }
439     { \tl_set:Nn #7 { gray ~ #1 } }
440     {
441       \str_if_eq:nnTF {#4} { rg }
442       { \tl_set:Nn #7 { rgb ~ #1 ~ #2 ~ #3 } }
443       {
444         \str_if_eq:nnTF {#5} { k }
445         { \tl_set:Nn #7 { cmyk ~ #1 ~ #2 ~ #3 ~ #4 } }
446         {
447           \str_if_eq:nnTF {#2} { cs }
448           {
449             \tl_set:Nx #7 { spot ~ \use_none:n #1 ~ #5 }
450           }
451           {
452             \tl_set:Nn #7 { gray ~ 0 }
453           }
454         }
455       }
456     }
457   }
458 }
459 }
460 </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.

```

461 \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:
462 \cs_new_protected:Npn \__color_backend_cmyk:nnnn #1#2#3#4
463 {
464   \use:x
465   {
466     \__color_backend_cmyk_aux:nnnn
467     { \fp_eval:n {#1} }
468     { \fp_eval:n {#2} }
469     { \fp_eval:n {#3} }
470     { \fp_eval:n {#4} }
471   }
472 }
473 \cs_new_protected:Npn \__color_backend_cmyk_aux:nnnn #1#2#3#4
474 {
475   \__color_backend_select:n
476   { #1 ~ #2 ~ #3 ~ #4 ~ k ~ #1 ~ #2 ~ #3 ~ #4 ~ K }
477 }
478 \cs_new_protected:Npn \__color_backend_gray:n #1
479 { \exp_args:Nx \__color_backend_gray_aux:n { \fp_eval:n {#1} } }
480 \cs_new_protected:Npn \__color_backend_gray_aux:n #1
481 { \__color_backend_select:n { #1 ~ g ~ #1 ~ G } }
482 \cs_new_protected:Npn \__color_backend_rgb:nnn #1#2#3

```

```

483 {
484   \use:x
485   {
486     \_color_backend_rgb_aux:nnn
487     { \fp_eval:n {#1} }
488     { \fp_eval:n {#2} }
489     { \fp_eval:n {#3} }
490   }
491 }
492 \cs_new_protected:Npn \_color_backend_rgb_aux:nnn #1#2#3
493 { \_color_backend_select:n { #1 ~ #2 ~ #3 ~ rg ~ #1 ~ #2 ~ #3 ~ RG } }
494 \cs_new_protected:Npn \_color_backend_spot:nn #1#2
495 { \_color_backend_select:n { /#1 ~ cs ~ /#1 ~ CS ~ #2 ~ sc ~ #2 ~ SC } }
496 \cs_new_protected:Npx \_color_backend_select:n #1
497 {
498   \cs_if_exist:NTF \tex_pdfextension:D
499   { \tex_pdfextension:D colorstack }
500   { \tex_pdfcolorstack:D }
501   \exp_not:N \l__kernel_color_stack_int push {#1}
502   \group_insert_after:N \exp_not:N \_color_backend_reset:
503 }
504 \cs_generate_variant:Nn \_color_backend_select:n { x }
505 \cs_new_protected:Npx \_color_backend_reset:
506 {
507   \cs_if_exist:NTF \tex_pdfextension:D
508   { \tex_pdfextension:D colorstack }
509   { \tex_pdfcolorstack:D }
510   \exp_not:N \l__kernel_color_stack_int pop \scan_stop:
511 }

```

(End definition for _color_backend_cmyk:nnnn and others.)

```

512 </pdfmode>
513 </initex | package>

```

4 I3backend-draw Implementation

```

514 <*initex | package>
515 <@@=draw>

```

4.1 dvips backend

```

516 <*dvips>

```

```

\_draw_backend_literal:n The same as literal PostScript: same arguments about positioning apply her.
\_draw_backend_literal:x 517 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
518 \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.

```

519 \cs_new_protected:Npn \__draw_backend_begin:
520 {
521   \__kernel_backend_literal:n { ps::[begin] }
522   \__draw_backend_literal:n { @beginspecial }
523   \__draw_backend_literal:n { SDict ~ begin ~ /color.fc ~ { } ~ def ~ end }
524 }
525 \cs_new_protected:Npn \__draw_backend_end:
526 {
527   \__draw_backend_literal:n { @endspecial }
528   \__kernel_backend_literal:n { ps::[end] }
529 }

```

(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.

```

530 \cs_new_protected:Npn \__draw_backend_scope_begin:
531 { \__draw_backend_literal:n { save } }
532 \cs_new_protected:Npn \__draw_backend_scope_end:
533 { \__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.

```

534 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
535 {
536   \__draw_backend_literal:x
537   {
538     \dim_to_decimal_in_bp:n {#1} ~
539     \dim_to_decimal_in_bp:n {#2} ~ moveto
540   }
541 }
542 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
543 {
544   \__draw_backend_literal:x
545   {
546     \dim_to_decimal_in_bp:n {#1} ~
547     \dim_to_decimal_in_bp:n {#2} ~ lineto
548   }
549 }
550 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
551 {
552   \__draw_backend_literal:x
553   {
554     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~

```



```

555         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
556         moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
557     }
558 }
559 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
560 {
561     \__draw_backend_literal:x
562     {
563         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
564         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
565         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
566         curveto
567     }
568 }

```

(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
559 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
560 { \bool_gset_true:N \g__draw_draw_eor_bool }
561 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
562 { \bool_gset_false:N \g__draw_draw_eor_bool }
563 \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_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
574 \cs_new_protected:Npn \__draw_backend_closepath:
575 { \__draw_backend_literal:n { closepath } }
576 \cs_new_protected:Npn \__draw_backend_stroke:
577 {
578     \__draw_backend_literal:n { stroke }
579     \bool_if:NT \g__draw_draw_clip_bool
580     {
581         \__draw_backend_literal:x
582         {
583             \bool_if:NT \g__draw_draw_eor_bool { eo }
584             clip
585         }
586     }
587     \__draw_backend_literal:n { newpath }
588     \bool_gset_false:N \g__draw_draw_clip_bool
589 }
590 \cs_new_protected:Npn \__draw_backend_closestroke:
591 {
592     \__draw_backend_closepath:
593     \__draw_backend_stroke:
594 }

```

```

595 \cs_new_protected:Npn \__draw_backend_fill:
596 {
597   \__draw_backend_literal:n { gsave }
598   \__draw_backend_literal:n { color.fc }
599   \__draw_backend_literal:x
600   {
601     \bool_if:NT \g__draw_draw_eor_bool { eo }
602     fill
603   }
604   \__draw_backend_literal:n { grestore }
605   \bool_if:NT \g__draw_draw_clip_bool
606   {
607     \__draw_backend_literal:x
608     {
609       \bool_if:NT \g__draw_draw_eor_bool { eo }
610       clip
611     }
612   }
613   \__draw_backend_literal:n { newpath }
614   \bool_gset_false:N \g__draw_draw_clip_bool
615 }
616 \cs_new_protected:Npn \__draw_backend_fillstroke:
617 {
618   \__draw_backend_literal:n { gsave }
619   \__draw_backend_literal:n { color.fc }
620   \__draw_backend_literal:x
621   {
622     \bool_if:NT \g__draw_draw_eor_bool { eo }
623     fill
624   }
625   \__draw_backend_literal:n { grestore }
626   \__draw_backend_literal:n { stroke }
627   \bool_if:NT \g__draw_draw_clip_bool
628   {
629     \__draw_backend_literal:x
630     {
631       \bool_if:NT \g__draw_draw_eor_bool { eo }
632       clip
633     }
634   }
635   \__draw_backend_literal:n { newpath }
636   \bool_gset_false:N \g__draw_draw_clip_bool
637 }
638 \cs_new_protected:Npn \__draw_backend_clip:
639 { \bool_gset_true:N \g__draw_draw_clip_bool }
640 \bool_new:N \g__draw_draw_clip_bool
641 \cs_new_protected:Npn \__draw_backend_discardpath:
642 {
643   \bool_if:NT \g__draw_draw_clip_bool
644   {
645     \__draw_backend_literal:x
646     {
647       \bool_if:NT \g__draw_draw_eor_bool { eo }
648       clip

```

```

649     }
650   }
651   \__draw_backend_literal:n { newpath }
652   \bool_gset_false:N \g__draw_draw_clip_bool
653 }

```

(End definition for __draw_backend_closepath: and others.)

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

```

654 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
655 {
656   \__draw_backend_literal:x
657   {
658     [
659       \exp_args:Nf \use:n
660       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
661     ] ~
662     \dim_to_decimal_in_bp:n {#2} ~ setdash
663   }
664 }
665 \cs_new:Npn \__draw_backend_dash:n #1
666 { ~ \dim_to_decimal_in_bp:n {#1} }
667 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
668 {
669   \__draw_backend_literal:x
670   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
671 }
672 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
673 { \__draw_backend_literal:x { \fp_eval:n {#1} ~ setmiterlimit } }
674 \cs_new_protected:Npn \__draw_backend_cap_but:
675 { \__draw_backend_literal:n { 0 ~ setlinecap } }
676 \cs_new_protected:Npn \__draw_backend_cap_round:
677 { \__draw_backend_literal:n { 1 ~ setlinecap } }
678 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
679 { \__draw_backend_literal:n { 2 ~ setlinecap } }
680 \cs_new_protected:Npn \__draw_backend_join_miter:
681 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
682 \cs_new_protected:Npn \__draw_backend_join_round:
683 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
684 \cs_new_protected:Npn \__draw_backend_join_bevel:
685 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

(End definition for __draw_backend_dash_pattern:nn and others.)

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

```

686 \cs_new_protected:Npn \__draw_backend_color_fill_cmyk:nnnn #1#2#3#4
687 {
688   \__draw_backend_color_fill:x
689   {
690     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
691     \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
692     setcmykcolor

```

```

693     }
694   }
695   \cs_new_protected:Npn \__draw_backend_color_stroke:cm:nnnn #1#2#3#4
696   {
697     \__draw_backend_color_stroke:x
698     {
699       cmyk ~
700       \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
701       \fp_eval:n {#3} ~ \fp_eval:n {#4}
702     }
703   }
704   \cs_new_protected:Npn \__draw_backend_color_fill_gray:n #1
705   { \__draw_backend_color_fill:x { \fp_eval:n {#1} ~ setgray } }
706   \cs_new_protected:Npn \__draw_backend_color_stroke_gray:n #1
707   { \__draw_backend_color_stroke:x { gray ~ \fp_eval:n {#1} } }
708   \cs_new_protected:Npn \__draw_backend_color_fill_rgb:nnn #1#2#3
709   {
710     \__draw_backend_color_fill:x
711     { \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} ~ setrgbcolor }
712   }
713   \cs_new_protected:Npn \__draw_backend_color_stroke_rgb:nnn #1#2#3
714   {
715     \__draw_backend_color_stroke:x
716     { rgb ~ \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} }
717   }
718   \cs_new_protected:Npn \__draw_backend_color_fill:n #1
719   {
720     \__kernel_backend_postscript:n
721     { /color.fc ~ { #1 } ~ def }
722   }
723   \cs_generate_variant:Nn \__draw_backend_color_fill:n { x }
724   \cs_new_protected:Npn \__draw_backend_color_stroke:n #1
725   {
726     \__kernel_backend_literal:n { color~push~#1 }
727     \group_insert_after:N \__draw_color_reset:
728   }
729   \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.

```

730   \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
731   {
732     \__draw_backend_literal:n
733     {
734       [
735         \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
736         \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
737         0 ~ 0
738       ] ~
739     concat

```

```

740     }
741 }

```

(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).

```

742 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
743 {
744   \_draw_backend_literal:n { @endspecial }
745   \_draw_backend_literal:n { [end] }
746   \_draw_backend_literal:n { [begin] }
747   \_draw_backend_literal:n { save }
748   \_draw_backend_literal:n { currentpoint }
749   \_draw_backend_literal:n { currentpoint~translate }
750   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
751   \_draw_backend_cm:nnnn { #2 } { #3 } { #4 } { #5 }
752   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
753   \_draw_backend_literal:n { neg~exch~neg~exch~translate }
754   \_draw_backend_literal:n { [end] }
755   \hbox_overlap_right:n { \box_use:N #1 }
756   \_draw_backend_literal:n { [begin] }
757   \_draw_backend_literal:n { restore }
758   \_draw_backend_literal:n { [end] }
759   \_draw_backend_literal:n { [begin] }
760   \_draw_backend_literal:n { @beginspecial }
761 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```

762 </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.

```

763 <*dvipdfmx | pdfmode | xdvipdfmx>

```

4.2.1 Drawing

`_draw_backend_literal:n`
`_draw_backend_literal:x`

Pass data through using a dedicated interface.

```

764 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_pdf:n
765 \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:
766 \cs_new_protected:Npn \__draw_backend_begin:
767 { \__draw_backend_scope_begin: }
768 \cs_new_protected:Npn \__draw_backend_end:
769 { \__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:
770 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
771 \cs_new_eq:NN \__draw_backend_scope_end: \__kernel_backend_scope_end:
```

(End definition for `__draw_backend_scope_begin:` and `__draw_backend_scope_end:.`)

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

```
\__draw_backend_lineto:nn
\__draw_backend_curveto:nnnnnn
\__draw_backend_rectangle:nnnn
772 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
773 {
774   \__draw_backend_literal:x
775   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
776 }
777 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
778 {
779   \__draw_backend_literal:x
780   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
781 }
782 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
783 {
784   \__draw_backend_literal:x
785   {
786     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
787     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
788     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
789     c
790   }
791 }
792 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
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     re
799   }
800 }
```

(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
801 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
802 { \bool_gset_true:N \g__draw_draw_eor_bool }
803 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
804 { \bool_gset_false:N \g__draw_draw_eor_bool }
805 \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:`.)

<code>__draw_backend_closepath:</code>	Converting paths to output is again a case of mapping directly to PDF operations.
<code>__draw_backend_stroke:</code>	806 <code>\cs_new_protected:Npn __draw_backend_closepath:</code>
<code>__draw_backend_closestroke:</code>	807 <code>{ __draw_backend_literal:n { h } }</code>
<code>__draw_backend_fill:</code>	808 <code>\cs_new_protected:Npn __draw_backend_stroke:</code>
<code>__draw_backend_fillstroke:</code>	809 <code>{ __draw_backend_literal:n { S } }</code>
<code>__draw_backend_clip:</code>	810 <code>\cs_new_protected:Npn __draw_backend_closestroke:</code>
<code>__draw_backend_discardpath:</code>	811 <code>{ __draw_backend_literal:n { s } }</code>
	812 <code>\cs_new_protected:Npn __draw_backend_fill:</code>
	813 <code>{</code>
	814 <code>__draw_backend_literal:x</code>
	815 <code>{ f \bool_if:NT \g__draw_draw_eor_bool * }</code>
	816 <code>}</code>
	817 <code>\cs_new_protected:Npn __draw_backend_fillstroke:</code>
	818 <code>{</code>
	819 <code>__draw_backend_literal:x</code>
	820 <code>{ B \bool_if:NT \g__draw_draw_eor_bool * }</code>
	821 <code>}</code>
	822 <code>\cs_new_protected:Npn __draw_backend_clip:</code>
	823 <code>{</code>
	824 <code>__draw_backend_literal:x</code>
	825 <code>{ W \bool_if:NT \g__draw_draw_eor_bool * }</code>
	826 <code>}</code>
	827 <code>\cs_new_protected:Npn __draw_backend_discardpath:</code>
	828 <code>{ __draw_backend_literal:n { n } }</code>

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

<code>__draw_backend_dash_pattern:nn</code>	Converting paths to output is again a case of mapping directly to PDF operations.
<code>__draw_backend_dash:n</code>	829 <code>\cs_new_protected:Npn __draw_backend_dash_pattern:nn #1#2</code>
<code>__draw_backend_linewidth:n</code>	830 <code>{</code>
<code>__draw_backend_miterlimit:n</code>	831 <code>__draw_backend_literal:x</code>
<code>__draw_backend_cap_butt:</code>	832 <code>{</code>
<code>__draw_backend_cap_round:</code>	833 <code>[</code>
<code>__draw_backend_cap_rectangle:</code>	834 <code>\exp_args:Nf \use:n</code>
<code>__draw_backend_join_miter:</code>	835 <code>{ \clist_map_function:nN {#1} __draw_backend_dash:n }</code>
<code>__draw_backend_join_round:</code>	836 <code>] ~</code>
<code>__draw_backend_join_bevel:</code>	837 <code>\dim_to_decimal_in_bp:n {#2} ~ d</code>
	838 <code>}</code>
	839 <code>}</code>
	840 <code>\cs_new:Npn __draw_backend_dash:n #1</code>
	841 <code>{ ~ \dim_to_decimal_in_bp:n {#1} }</code>
	842 <code>\cs_new_protected:Npn __draw_backend_linewidth:n #1</code>
	843 <code>{</code>
	844 <code>__draw_backend_literal:x</code>
	845 <code>{ \dim_to_decimal_in_bp:n {#1} ~ w }</code>
	846 <code>}</code>
	847 <code>\cs_new_protected:Npn __draw_backend_miterlimit:n #1</code>
	848 <code>{ __draw_backend_literal:x { \fp_eval:n {#1} ~ M } }</code>
	849 <code>\cs_new_protected:Npn __draw_backend_cap_butt:</code>
	850 <code>{ __draw_backend_literal:n { 0 ~ J } }</code>
	851 <code>\cs_new_protected:Npn __draw_backend_cap_round:</code>

```

852 { \_draw_backend_literal:n { 1 ~ J } }
853 \cs_new_protected:Npn \_draw_backend_cap_rectangle:
854 { \_draw_backend_literal:n { 2 ~ J } }
855 \cs_new_protected:Npn \_draw_backend_join_miter:
856 { \_draw_backend_literal:n { 0 ~ j } }
857 \cs_new_protected:Npn \_draw_backend_join_round:
858 { \_draw_backend_literal:n { 1 ~ j } }
859 \cs_new_protected:Npn \_draw_backend_join_bevel:
860 { \_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:
861 \cs_new_protected:Npn \_draw_backend_color_fill_cmyk:nnnn #1#2#3#4
862 {
863   \_draw_backend_color_select:x
864   {
865     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
866     \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
867     k
868   }
869 }
870 \cs_new_protected:Npn \_draw_backend_color_stroke_cmyk:nnnn #1#2#3#4
871 {
872   \_draw_backend_color_select:x
873   {
874     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
875     \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
876     k
877   }
878 }
879 \cs_new_protected:Npn \_draw_backend_color_fill_gray:n #1
880 { \_draw_backend_color_select:x { \fp_eval:n {#1} ~ g } }
881 \cs_new_protected:Npn \_draw_backend_color_stroke_gray:n #1
882 { \_draw_backend_color_select:x { \fp_eval:n {#1} ~ G } }
883 \cs_new_protected:Npn \_draw_backend_color_fill_rgb:nnn #1#2#3
884 {
885   \_draw_backend_color_select:x
886   { \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} ~ rg }
887 }
888 \cs_new_protected:Npn \_draw_backend_color_stroke_rgb:nnn #1#2#3
889 {
890   \_draw_backend_color_select:x
891   { \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} ~ RG }
892 }
893 <*pdfmode>
894 \cs_new_protected:Npx \_draw_backend_color_select:n #1
895 {
896   \cs_if_exist:NTF \tex_pdfextension:D
897   { \tex_pdfextension:D colorstack }
898   { \tex_pdfcolorstack:D }
899   \exp_not:N \l__kernel_color_stack_int push {#1}
900   \group_insert_after:N \exp_not:N \_draw_backend_color_reset:

```



```

901 }
902 \cs_new_protected:Npx \__draw_backend_color_reset:
903 {
904   \cs_if_exist:NTF \tex_pdfextension:D
905   { \tex_pdfextension:D colorstack }
906   { \tex_pdfcolorstack:D }
907   \exp_not:N \l__kernel_color_stack_int pop \scan_stop:
908 }
909 \</pdfmode>
910 \<*dvipdfmx | xdvipdfmx>
911 \cs_new_eq:NN \__draw_backend_color_select:n \__kernel_backend_literal_pdf:n
912 \</dvipdfmx | xdvipdfmx>
913 \cs_generate_variant:Nn \__draw_backend_color_select:n { x }

```

(End definition for __draw_backend_color_fill_cmyk:nnnn and others.)

__draw_backend_cm:nnnn
__draw_backend_cm_aux:nnnn

Another split here between pdfmode and (x)dvipdfmx. In the former, we have a direct method to maintain alignment: the backend can use a matrix itself. For (x)dvipdfmx, we can to decompose the matrix into rotations and a scaling, then use those operations as they are handled by the backend. (There is backend support for matrix operations in (x)dvipdfmx, but as a matched pair so not suitable for the “stand alone” transformation set up here.)

```

914 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
915 {
916 \<*pdfmode>
917   \__kernel_backend_matrix:x
918   {
919     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
920     \fp_eval:n {#3} ~ \fp_eval:n {#4}
921   }
922 \</pdfmode>
923 \<*dvipdfmx | xdvipdfmx>
924   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
925   \__draw_backend_cm_aux:nnnn
926 \</dvipdfmx | xdvipdfmx>
927 }
928 \<*dvipdfmx | xdvipdfmx>
929 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
930 {
931   \__kernel_backend_literal:x
932   {
933     x:rotate~
934     \fp_compare:nNnTF {#1} = \c_zero_fp
935     { 0 }
936     { \fp_eval:n { round ( -#1 , 5 ) } }
937   }
938   \__kernel_backend_literal:x
939   {
940     x:scale~
941     \fp_eval:n { round ( #2 , 5 ) } ~
942     \fp_eval:n { round ( #3 , 5 ) }
943   }
944   \__kernel_backend_literal:x
945   {

```

```

946      x:rotate~
947      \fp_compare:nNnTF {#4} = \c_zero_fp
948        { 0 }
949        { \fp_eval:n { round ( -#4 , 5 ) } }
950    }
951  }
952  </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.

```

953  <*dvipdfmx | xdvipdfmx>
954  \cs_new_protected:Npn \_draw_backend_cm_decompose:nnnnN #1#2#3#4#5
955  {
956    \use:x
957    {
958      \_draw_backend_cm_decompose_auxi:nnnnN
959      { \fp_eval:n { (#1 + #4) / 2 } }
960      { \fp_eval:n { (#1 - #4) / 2 } }
961      { \fp_eval:n { (#3 + #2) / 2 } }
962      { \fp_eval:n { (#3 - #2) / 2 } }
963    }
964    #5
965  }
966  \cs_new_protected:Npn \_draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
967  {
968    \use:x

```

```

969 {
970   \_draw_backend_cm_decompose_auxii:nnnnN
971   { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
972   { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
973   { \fp_eval:n { atand ( #3 , #2 ) } }
974   { \fp_eval:n { atand ( #4 , #1 ) } }
975 }
976 #5
977 }
978 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
979 {
980   \use:x
981   {
982     \_draw_backend_cm_decompose_auxiii:nnnnN
983     { \fp_eval:n { ( #4 - #3 ) / 2 } }
984     { \fp_eval:n { ( #1 + #2 ) / 2 } }
985     { \fp_eval:n { ( #1 - #2 ) / 2 } }
986     { \fp_eval:n { ( #4 + #3 ) / 2 } }
987   }
988   #5
989 }
990 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
991 {
992   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
993   { #5 {#1} {#2} {#3} {#4} }
994   { #5 {#1} {#3} {#2} {#4} }
995 }
996 </dviPDFmx | xdvipdfmx>

```

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

`_draw_backend_box_use:Nnnnn`

Inserting a \TeX box transformed to the requested position and using the current matrix is done using a mixture of \TeX and low-level manipulation. The offset can be handled by \TeX , so only any rotation/skew/scaling component needs to be done using the matrix operation. As this operation can never be cached, the scope is set directly not using the `draw` version.

```

997 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
998 {
999   \_kernel_backend_scope_begin:
1000   <*pdfmode>
1001   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1002   </pdfmode>
1003   <*dviPDFmx | xdvipdfmx>
1004   \_kernel_backend_literal:x
1005   {
1006     pdf:btrans-matrix~
1007     \fp_eval:n {#2} ~ \fp_eval:n {#3} ~
1008     \fp_eval:n {#4} ~ \fp_eval:n {#5} ~
1009     0 ~ 0
1010   }
1011   </dviPDFmx | xdvipdfmx>
1012   \hbox_overlap_right:n { \box_use:N #1 }
1013   <*dviPDFmx | xdvipdfmx>
1014   \_kernel_backend_literal:n { pdf:etrans }

```

```

1015 </dvipdfmx | xdvipdfmx>
1016     \__kernel_backend_scope_end:
1017 }

(End definition for \__draw_backend_box_use:Nnnnn.)

1018 </dvipdfmx | pdfmode | xdvipdfmx>

```

4.3 dvisvgm backend

```

1019 <*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
1020 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1021 \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

```

1022 \cs_new_protected:Npn \__draw_backend_begin:
1023 {
1024     \__draw_backend_scope_begin:
1025     \__draw_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1026 }
1027 \cs_new_protected:Npn \__draw_backend_end:
1028 { \__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.

```

1029 \cs_new_protected:Npn \__draw_backend_scope_begin:
1030 {
1031     \int_set_eq:NN
1032     \l__draw_draw_scope_int
1033     \g__draw_draw_scope_int
1034     \group_begin:
1035     \int_gzero:N \g__draw_draw_scope_int
1036 }
1037 \cs_new_protected:Npn \__draw_backend_scope_end:
1038 {
1039     \prg_replicate:nn
1040     { \g__draw_draw_scope_int }
1041     { \__draw_backend_literal:n { </g> } }
1042     \group_end:
1043     \int_gset_eq:NN
1044     \g__draw_draw_scope_int
1045     \l__draw_draw_scope_int
1046 }
1047 \cs_new_protected:Npn \__draw_backend_scope:n #1

```

```

1048 {
1049   \_draw_backend_literal:n { <g~ #1 > }
1050   \int_gincr:N \g__draw_draw_scope_int
1051 }
1052 \cs_generate_variant:Nn \_draw_backend_scope:n { x }
1053 \int_new:N \g__draw_draw_scope_int
1054 \int_new:N \l__draw_draw_scope_int

```

(End definition for _draw_backend_scope_begin: and others.)

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

```

\_draw_backend_lineto:nn
  \_draw_backend_rectangle:nnnn
  \_draw_backend_curveto:nnnnnn
  \_draw_backend_add_to_path:n
\g__draw_draw_path_tl
1055 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1056 {
1057   \_draw_backend_add_to_path:n
1058   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1059 }
1060 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1061 {
1062   \_draw_backend_add_to_path:n
1063   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1064 }
1065 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1066 {
1067   \_draw_backend_add_to_path:n
1068   {
1069     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1070     h ~ \dim_to_decimal:n {#3} ~
1071     v ~ \dim_to_decimal:n {#4} ~
1072     h ~ \dim_to_decimal:n { -#3 } ~
1073     Z
1074   }
1075 }
1076 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1077 {
1078   \_draw_backend_add_to_path:n
1079   {
1080     C ~
1081     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1082     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1083     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1084   }
1085 }
1086 \cs_new_protected:Npn \_draw_backend_add_to_path:n #1
1087 {
1088   \tl_gset:Nx \g__draw_draw_path_tl
1089   {
1090     \g__draw_draw_path_tl
1091     \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1092     #1
1093   }

```

```

1094 }
1095 \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:
1096 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1097 { \__draw_backend_scope:n { fill-rule="evenodd" } }
1098 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1099 { \__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:
1100 \cs_new_protected:Npn \__draw_backend_closepath:
1101 { \__draw_backend_add_to_path:n { Z } }
\__draw_backend_clip:
1102 \cs_new_protected:Npn \__draw_backend_path:n #1
1103 {
1104 \bool_if:NTF \g__draw_draw_clip_bool
1105 {
1106 \int_gincr:N \g__draw_clip_path_int
1107 \__draw_backend_literal:x
1108 {
1109 < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1110 { ?nl }
1111 <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1112 < /clipPath > { ? nl }
1113 <
1114 use~xlink:href =
1115 "\c_hash_str l3path \int_use:N \g__draw_path_int " ~
1116 #1
1117 />
1118 }
1119 \__draw_backend_scope:x
1120 {
1121 clip-path =
1122 "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1123 }
1124 }
1125 {
1126 \__draw_backend_literal:x
1127 { <path ~ d=" \g__draw_draw_path_tl " ~ #1 /> }
1128 }
1129 \tl_gclear:N \g__draw_draw_path_tl
1130 \bool_gset_false:N \g__draw_draw_clip_bool
1131 }
1132 \int_new:N \g__draw_path_int
1133 \cs_new_protected:Npn \__draw_backend_stroke:
1134 { \__draw_backend_path:n { style="fill:none" } }
1135 \cs_new_protected:Npn \__draw_backend_closestroke:

```

```

1136 {
1137   \__draw_backend_closepath:
1138   \__draw_backend_stroke:
1139 }
1140 \cs_new_protected:Npn \__draw_backend_fill:
1141 { \__draw_backend_path:n { style="stroke:none" } }
1142 \cs_new_protected:Npn \__draw_backend_fillstroke:
1143 { \__draw_backend_path:n { } }
1144 \cs_new_protected:Npn \__draw_backend_clip:
1145 { \bool_gset_true:N \g__draw_draw_clip_bool }
1146 \bool_new:N \g__draw_draw_clip_bool
1147 \cs_new_protected:Npn \__draw_backend_discardpath:
1148 {
1149   \bool_if:NT \g__draw_draw_clip_bool
1150   {
1151     \int_gincr:N \g__draw_clip_path_int
1152     \__draw_backend_literal:x
1153     {
1154       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1155       { ?nl }
1156       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1157       < /clipPath >
1158     }
1159     \__draw_backend_scope:x
1160     {
1161       clip-path =
1162       "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int )"
1163     }
1164   }
1165   \tl_gclear:N \g__draw_draw_path_tl
1166   \bool_gset_false:N \g__draw_draw_clip_bool
1167 }

```

(End definition for __draw_backend_path:n and others.)

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_dash_aux:nn
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1168 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1169 {
1170   \use:x
1171   {
1172     \__draw_backend_dash_aux:nn
1173     { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1174     { \dim_to_decimal:n {#2} }
1175   }
1176 }
1177 \cs_new:Npn \__draw_backend_dash:n #1
1178 { , \dim_to_decimal_in_bp:n {#1} }
1179 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1180 {
1181   \__draw_backend_scope:x
1182   {
1183     stroke-dasharray =
1184     "

```

```

1185         \tl_if_empty:oTF { \use_none:n #1 }
1186         { none }
1187         { \use_none:n #1 }
1188     " ~
1189     stroke-offset=" #2 "
1190 }
1191 }
1192 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1193 { \__draw_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1194 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1195 { \__draw_backend_scope:x { stroke-miterlimit=" \fp_eval:n {#1} " } }
1196 \cs_new_protected:Npn \__draw_backend_cap_but:
1197 { \__draw_backend_scope:n { stroke-linecap="butt" } }
1198 \cs_new_protected:Npn \__draw_backend_cap_round:
1199 { \__draw_backend_scope:n { stroke-linecap="round" } }
1200 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1201 { \__draw_backend_scope:n { stroke-linecap="square" } }
1202 \cs_new_protected:Npn \__draw_backend_join_miter:
1203 { \__draw_backend_scope:n { stroke-linejoin="miter" } }
1204 \cs_new_protected:Npn \__draw_backend_join_round:
1205 { \__draw_backend_scope:n { stroke-linejoin="round" } }
1206 \cs_new_protected:Npn \__draw_backend_join_bevel:
1207 { \__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_stroke_cmyk:nnnn
\_draw_backend_color_fill_gray:n 1208 \cs_new_protected:Npn \__draw_backend_color_fill_cmyk:nnnn #1#2#3#4
\_draw_backend_color_stroke_gray:n 1209 {
\_draw_backend_color_fill_rgb:nnn 1210     \use:x
\_draw_backend_color_stroke_rgb:nnn 1211     {
\_draw_backend_color_fill:nnn 1212         \__draw_backend_color_fill:nnn
1213         { \fp_eval:n { -100 * ( (#1) * ( 1 - (#4) ) - 1 ) } }
1214         { \fp_eval:n { -100 * ( (#2) * ( 1 - (#4) ) + #4 - 1 ) } }
1215         { \fp_eval:n { -100 * ( (#3) * ( 1 - (#4) ) + #4 - 1 ) } }
1216     }
1217 }
1218 \cs_new_protected:Npn \__draw_backend_color_stroke_cmyk:nnnn #1#2#3#4
1219 {
1220     \__draw_backend_select:x
1221     {
1222         cmyk~
1223         \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
1224         \fp_eval:n {#3} ~ \fp_eval:n {#4}
1225     }
1226 }
1227 \cs_new_protected:Npn \__draw_backend_color_fill_gray:n #1
1228 {
1229     \use:x
1230     {
1231         \__draw_backend_color_gray_aux:n
1232         { \fp_eval:n { 100 * (#1) } }
1233     }

```



```

1234 }
1235 \cs_new_protected:Npn \__draw_backend_color_gray_aux:n #1
1236 { \__draw_backend_color_fill:nnn {#1} {#1} {#1} }
1237 \cs_new_protected:Npn \__draw_backend_color_stroke_gray:n #1
1238 { \__draw_backend_select:x { gray~ \fp_eval:n {#1} } }
1239 \cs_new_protected:Npn \__draw_backend_color_fill_rgb:nnn #1#2#3
1240 {
1241   \use:x
1242   {
1243     \__draw_backend_color_fill:nnn
1244     { \fp_eval:n { 100 * (#1) } }
1245     { \fp_eval:n { 100 * (#2) } }
1246     { \fp_eval:n { 100 * (#3) } }
1247   }
1248 }
1249 \cs_new_protected:Npn \__draw_backend_color_fill:nnn #1#2#3
1250 {
1251   \__draw_backend_scope:x
1252   {
1253     fill =
1254     "
1255     rgb
1256     (
1257       #1 \c_percent_str ,
1258       #2 \c_percent_str ,
1259       #3 \c_percent_str
1260     )
1261     "
1262   }
1263 }
1264 \cs_new_protected:Npn \__draw_backend_color_stroke_rgb:nnn #1#2#3
1265 {
1266   \__draw_backend_select:x
1267   { rgb~ \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} }
1268 }

```

(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.

```

1269 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1270 {
1271   \__draw_backend_scope:n
1272   {
1273     transform =
1274     "
1275     matrix
1276     (
1277       \fp_eval:n {#1} , \fp_eval:n {#2} ,
1278       \fp_eval:n {#3} , \fp_eval:n {#4} ,
1279       0pt , 0pt
1280     )
1281     "
1282   }
1283 }

```

(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.

```

1284 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1285 {
1286   \_kernel_backend_scope_begin:
1287   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1288   \_kernel_backend_literal_svg:n
1289   {
1290     < g~
1291     stroke="none"~
1292     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1293     >
1294   }
1295   \box_set_wd:Nn #1 { Opt }
1296   \box_set_ht:Nn #1 { Opt }
1297   \box_set_dp:Nn #1 { Opt }
1298   \box_use:N #1
1299   \_kernel_backend_literal_svg:n { </g> }
1300   \_kernel_backend_scope_end:
1301 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```

1302 </divisvgm>
1303 </initex | package>

```

5 l3backend-graphics Implementation

```

1304 <*initex | package>
1305 <@@=graphics>

```

5.1 dvips backend

```

1306 <*dvips>

```

`_graphics_backend_getbb_eps:n` Simply use the generic function.

```

1307 <*initex>
1308 \use:n
1309 </initex>
1310 <*package>
1311 \AtBeginDocument
1312 </package>
1313 { \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.

```

1314 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1315 {
1316   \_kernel_backend_literal:x
1317   {
1318     PSfile = #1 \c_space_tl
1319     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl

```

```

1320         lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1321         urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1322         ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1323     }
1324 }

(End definition for \__graphics_backend_include_eps:n.)

1325 </dvips>

```

5.2 pdfmode backend

```

1326 (*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.

```

1327 \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.

```

1328 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1329 {
1330     \int_zero:N \l_graphics_page_int
1331     \tl_clear:N \l_graphics_pagebox_tl
1332     \tl_set:Nx \l__graphics_graphics_attr_tl
1333     {
1334         \tl_if_empty:NF \l_graphics_decodearray_tl
1335         { :D \l_graphics_decodearray_tl }
1336         \bool_if:NT \l_graphics_interpolate_bool
1337         { :I }
1338     }
1339     \tl_clear:N \l__graphics_graphics_attr_tl
1340     \__graphics_backend_getbb_auxi:n {#1}
1341 }
1342 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1343 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1344 {
1345     \tl_clear:N \l_graphics_decodearray_tl
1346     \bool_set_false:N \l_graphics_interpolate_bool
1347     \tl_set:Nx \l__graphics_graphics_attr_tl
1348     {
1349         : \l_graphics_pagebox_tl
1350         \int_compare:nNnT \l_graphics_page_int > 1
1351         { :P \int_use:N \l_graphics_page_int }
1352     }
1353     \__graphics_backend_getbb_auxi:n {#1}
1354 }

```

```

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

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

_graphics_backend_include_jpg:n Images are already loaded for the measurement part of the code, so inclusion is straightforward, with only any attributes to worry about. The latter carry through from determination of the bounding box.

```

1394 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1395 {
1396   \tex_pdfrefximage:D
1397   \int_use:c { c__graphics_graphics_#1 \l__graphics_graphics_attr_tl _int }
1398 }
1399 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1400 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

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

```

\__graphics_backend_getbb_eps:n EPS graphics may be included in pdfmode by conversion to PDF: this requires restricted
\__graphics_backend_getbb_eps:nm shell escape. Modelled on the epstopdf LATEX 2ε package, but simplified, conversion takes
\__graphics_backend_include_eps:n place here if we have shell access.

\l__graphics_backend_dir_str 1401 \sys_if_shell:T
\l__graphics_backend_name_str 1402 {
\l__graphics_backend_ext_str 1403 \str_new:N \l__graphics_backend_dir_str
1404 \str_new:N \l__graphics_backend_name_str
1405 \str_new:N \l__graphics_backend_ext_str
1406 \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1407 {
1408 \file_parse_full_name:nNNN {#1}
1409 \l__graphics_backend_dir_str
1410 \l__graphics_backend_name_str
1411 \l__graphics_backend_ext_str
1412 \exp_args:Nx \__graphics_backend_getbb_eps:nn
1413 {
1414 \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1415 -converted-to.pdf
1416 }
1417 {#1}
1418 }
1419 \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1420 {
1421 \file_compare_timestamp:nNnT {#2} > {#1}
1422 {
1423 \sys_shell_now:n
1424 { repstopdf ~ #2 ~ #1 }
1425 }
1426 \tl_set:Nn \l_graphics_name_tl {#1}
1427 \__graphics_backend_getbb_pdf:n {#1}
1428 }
1429 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1430 {
1431 \file_parse_full_name:nNNN {#1}
1432 \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1433 \exp_args:Nx \__graphics_backend_include_pdf:n
1434 {
1435 \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1436 -converted-to.pdf
1437 }
1438 }
1439 }

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

1440 </pdfmode>

```

5.3 dvipdfmx backend

```

1441 <*dvipdfmx | xdvipdfmx>

\__graphics_backend_getbb_eps:n Simply use the generic functions: only for dvipdfmx in the extraction cases.
\__graphics_backend_getbb_jpg:n 1442 <*initex>
\__graphics_backend_getbb_pdf:n 1443 \use:n
\__graphics_backend_getbb_png:n 1444 </initex>

```

```

1445 <*package>
1446 \AtBeginDocument
1447 </package>
1448 { \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n }
1449 <*dvipdfmx>
1450 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1451 {
1452   \int_zero:N \l_graphics_page_int
1453   \tl_clear:N \l_graphics_pagebox_tl
1454   \graphics_extract_bb:n {#1}
1455 }
1456 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1457 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1458 {
1459   \tl_clear:N \l_graphics_decodearray_tl
1460   \bool_set_false:N \l_graphics_interpolate_bool
1461   \graphics_extract_bb:n {#1}
1462 }
1463 </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.

```

1464 \int_new:N \g__graphics_track_int

```

(End definition for \g__graphics_track_int.)

<pre> __graphics_backend_include_eps:n __graphics_backend_include_jpg:n __graphics_backend_include_pdf:n __graphics_backend_include_png:n __graphics_backend_include_auxi:nn __graphics_backend_include_auxii:nnn __graphics_backend_include_auxiii:nnn </pre>	<p>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.</p> <pre> 1465 \cs_new_protected:Npn __graphics_backend_include_eps:n #1 1466 { 1467 __kernel_backend_literal:x 1468 { 1469 PSfile = #1 \c_space_tl 1470 llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl 1471 lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl 1472 urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl 1473 ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim 1474 } 1475 } 1476 \cs_new_protected:Npn __graphics_backend_include_jpg:n #1 1477 { __graphics_backend_include_auxi:nn {#1} { image } } 1478 \cs_new_eq:NN __graphics_backend_include_png:n __graphics_backend_include_jpg:n 1479 <*dvipdfmx> 1480 \cs_new_protected:Npn __graphics_backend_include_pdf:n #1 1481 { __graphics_backend_include_auxi:nn {#1} { epdf } } 1482 </dvipdfmx> </pre>
---	--

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.

```

1483 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2

```

```

1484 {
1485   \__graphics_backend_include_auxii:xnn
1486   {
1487     \tl_if_empty:NF \l_graphics_pagebox_tl
1488     { : \l_graphics_pagebox_tl }
1489     \int_compare:nNnT \l_graphics_page_int > 1
1490     { :P \int_use:N \l_graphics_page_int }
1491     \tl_if_empty:NF \l_graphics_decodearray_tl
1492     { :D \l_graphics_decodearray_tl }
1493     \bool_if:NT \l_graphics_interpolate_bool
1494     { :I }
1495   }
1496   {#1} {#2}
1497 }
1498 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1499 {
1500   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1501   {
1502     \__kernel_backend_literal:x
1503     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1504   }
1505   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1506 }
1507 \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!

```

1508 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1509 {
1510   \int_gincr:N \g__graphics_track_int
1511   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1512   \__kernel_backend_literal:x
1513   {
1514     pdf:#3~
1515     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1516     \int_compare:nNnT \l_graphics_page_int > 1
1517     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1518     \tl_if_empty:NF \l_graphics_pagebox_tl
1519     {
1520       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1521       bbox ~
1522         \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1523         \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1524         \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1525         \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1526     }
1527     (#1)
1528     \bool_lazy_or:nnT
1529     { \l_graphics_interpolate_bool }
1530     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1531     {
1532       <<
1533       \tl_if_empty:NF \l_graphics_decodearray_tl

```

```

1534             { /Decode~[ \l_graphics_decodearray_tl ] }
1535             \bool_if:NT \l_graphics_interpolate_bool
1536             { /Interpolate~true> }
1537         >>
1538     }
1539 }
1540 }

```

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

```

1541 </dvipdfmx | xdvipdfmx>

```

5.4 xdvipdfmx backend

```

1542 < *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 $\text{Xe}\text{L}\text{A}\text{T}\text{E}\text{X}$ primitive omits the text box from the page box specification, so there is also some “trimming” to do here.

```

\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_auxvi:nNnn
\graphics_backend_getbb_pagebox:w

```

```

1543 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1544 {
1545     \int_zero:N \l_graphics_page_int
1546     \tl_clear:N \l_graphics_pagebox_tl
1547     \_graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1548 }
1549 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1550 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1551 {
1552     \tl_clear:N \l_graphics_decodearray_tl
1553     \bool_set_false:N \l_graphics_interpolate_bool
1554     \_graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1555 }
1556 \cs_new_protected:Npn \_graphics_backend_getbb_auxi:nN #1#2
1557 {
1558     \int_compare:nNnTF \l_graphics_page_int > 1
1559     { \_graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
1560     { \_graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1561 }
1562 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:nNn #1#2#3
1563 { \_graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1564 \cs_generate_variant:Nn \_graphics_backend_getbb_auxii:nNn { V }
1565 \cs_new_protected:Npn \_graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1566 {
1567     \tl_if_empty:NTF \l_graphics_pagebox_tl
1568     { \_graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1569     { \_graphics_backend_getbb_auxv:nNnn }
1570     {#1} #2 {#3} {#4}
1571 }
1572 \cs_new_protected:Npn \_graphics_backend_getbb_auxiv:nNnn #1#2#3#4#5
1573 {
1574     \use:x
1575     {

```



```

1576         \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1577         { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1578     }
1579 }
1580 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1581 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1582 {
1583     \graphics_bb_restore:nF {#1#3}
1584     { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1585 }
1586 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1587 {
1588     \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1589     \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1590     \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1591     \graphics_bb_save:n {#1#3}
1592 }
1593 \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

__graphics_backend_include_pdf:n
__graphics_backend_include_bitmap_quote:w

For PDF graphics, properly supporting the pagebox concept in X_YTeX is best done using the \tex_XeTeXpdfvfile: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.

```

1594 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1595 {
1596     \tex_XeTeXpdfvfile:D
1597     \__graphics_backend_include_pdf_quote:w #1 "#1" \q_stop \c_space_tl
1598     \int_compare:nNnT \l_graphics_page_int > 0
1599         { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1600     \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1601 }
1602 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \q_stop
1603 { " #2 " }

```

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

```

1604 \</xdvipdfmx>

```

5.5 dvisvgm backend

```

1605 <*dvisvgm>

```

__graphics_backend_getbb_eps:n Simply use the generic function.

```

1606 <*initex>
1607 \use:n
1608 </initex>
1609 <*package>
1610 \AtBeginDocument
1611 </package>
1612 { \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`

```

1613 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1614 {
1615     \int_zero:N \l_graphics_page_int
1616     \tl_clear:N \l_graphics_pagebox_tl
1617     \graphics_extract_bb:n {#1}
1618 }
1619 \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

```

1620 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1621 {
1622     \tl_clear:N \l_graphics_decodearray_tl
1623     \bool_set_false:N \l_graphics_interpolate_bool
1624     \graphics_extract_bb:n {#1}
1625 }

```

(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`

```

1626 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1627 { \_graphics_backend_include:nn { PSfile } {#1} }
1628 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1629 { \_graphics_backend_include:nn { pdffile } {#1} }
1630 \cs_new_protected:Npn \_graphics_backend_include:nn #1#2
1631 {
1632     \_kernel_backend_literal:x
1633     {
1634         #1 = #2 \c_space_tl
1635         llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1636         lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1637         urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1638         ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1639     }
1640 }

```

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

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

`_graphics_backend_include_jpg:n`

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

```

1641 \cs_new_protected:Npn \_graphics_backend_include_png:n #1
1642 {
1643     \_kernel_backend_literal:x
1644     {
1645         dvisvgm:img~
1646         \dim_to_decimal:n { \l_graphics_ury_dim } ~
1647         \dim_to_decimal:n { \l_graphics_ury_dim } ~

```

```

1648         \_graphics_backend_include_bitmap_quote:w #1 " #1 " \q_stop
1649     }
1650 }
1651 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_png:n
1652 \cs_new:Npn \_graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \q_stop
1653 { " #2 " }

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

1654 \</divsvgm>
1655 \</initex | package>

```

6 l3backend-pdf Implementation

```

1656 \<*initex | package>
1657 \<@=pdf>

```

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

6.1 Shared code

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

```

\_l\_pdf\_internal\_box

1658 \box_new:N \l\_pdf\_internal\_box

(End definition for \l\_pdf\_internal\_box.)

```

6.2 dvips backend

```

1659 \<*dvips>

\_pdf_backend_pdfmark:n Used often enough it should be a separate function.
\_pdf_backend_pdfmark:x
1660 \cs_new_protected:Npn \_pdf_backend_pdfmark:n #1
1661 { \_kernel_backend_postscript:n { mark #1 ~ pdfmark } }
1662 \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
1663 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
1664 { \_pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
1665 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
1666 { \_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`

```
1667 \int_new:N \g__pdf_backend_object_int
1668 \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`

```
1669 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
1670 {
1671   \int_gincr:N \g__pdf_backend_object_int
1672   \int_const:cn
1673     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
1674     { \g__pdf_backend_object_int }
1675   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
1676 }
1677 \cs_new:Npn \__pdf_backend_object_ref:n #1
1678 { { 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`

```
1679 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
1680 {
1681   \__pdf_backend_pdfmark:x
1682   {
1683     /_objdef ~ \__pdf_backend_object_ref:n {#1}
1684     /type
1685     \str_case_e:nn
1686       { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
1687       {
1688         { array } { /array }
1689         { dict } { /dict }
1690         { fstream } { /stream }
1691         { stream } { /stream }
1692       }
1693     /OBJ
1694   }
1695   \use:c
1696     { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
1697     { \__pdf_backend_object_ref:n {#1} } {#2}
1698 }
1699 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
1700 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
1701 {
1702   \__pdf_backend_pdfmark:x
1703     { #1 [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
1704 }
1705 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
1706 {
1707   \__pdf_backend_pdfmark:x
1708     { #1 << \exp_not:n {#2} >> /PUT }
1709 }
1710 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
```

```

1711 {
1712   \exp_args:Nx
1713   \__pdf_backend_object_write_stream:nnn {#1} #2
1714 }
1715 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
1716 {
1717   \__kernel_backend_postscript:n
1718   {
1719     [nobreak]
1720     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
1721     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
1722   }
1723 }

```

(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 1724 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
1725 {
1726   \int_gincr:N \g__pdf_backend_object_int
1727   \__pdf_backend_pdfmark:x
1728   {
1729     /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
1730     /type
1731     \str_case:nn
1732       {#1}
1733       {
1734         { array } { /array }
1735         { dict } { /dict }
1736         { fstream } { /stream }
1737         { stream } { /stream }
1738       }
1739     /OBJ
1740   }
1741   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
1742   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
1743 }
1744 \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.

```

1745 \cs_new:Npn \__pdf_backend_object_last:
1746 { { 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.

\l__pdf_backend_content_box The content of an annotation.

```

1747 \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.

1748 \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.

1749 \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 L^AT_EX 2_ε 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.

1750 \cs_new_protected:Npn _pdf_backend_annotation:nnnn #1#2#3#4

1751 {

1752 _pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4}

1753 \int_gincr:N \g__pdf_backend_object_int

1754 \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int

1755 _pdf_backend_pdfmark:x

1756 {

1757

1758 /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }

1759 pdf.rect ~

1760 #4 ~

1761 /ANN

1762 }

1763 }

1764 \cs_new_protected:Npn _pdf_backend_annotation_aux:nnnn #1#2#3#4

1765 {

1766 \box_move_down:nn {#3}

1767 { \hbox:n { _kernel_backend_postscript:n { pdf.save.ll } } }

1768 \hbox:n {#4}

1769 \box_move_up:nn {#2}

1770 {

1771 \hbox:n

1772 {

1773 \tex_kern:D \dim_eval:n {#1} \scan_stop:

1774 _kernel_backend_postscript:n { pdf.save.ur }

1775 }

1776 }

1777 \int_gincr:N \g__pdf_backend_object_int

1778 \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int

1779 _pdf_backend_pdfmark:x

1780 {

1781 /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }

1782 pdf.rect

1783 /ANN

1784 }

1785 }

(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.

```
1786 \cs_new:Npn \_pdf_backend_annotation_last:
1787 { { 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.

```
1788 \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.

```
1789 \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.

```
1790 \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.

```
1791 \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.

```
1792 \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.

```
1793 \tl_new:N \l__pdf_breaklink_pdfmark_tl
1794 \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.

```
1795 \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.

```
1796 \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.

```

1797 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
1798 { \__pdf_backend_link_begin:nw { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
1799 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
1800 { \__pdf_backend_link_begin:nw {#1#2} }
1801 \cs_new_protected:Npn \__pdf_backend_link_begin:nw #1
1802 {
1803   \bool_if:NF \g__pdf_backend_link_bool
1804   { \__pdf_backend_link_begin_aux:nw {#1} }
1805 }
1806 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
1807 {
1808   \bool_gset_true:N \g__pdf_backend_link_bool
1809   \__kernel_backend_postscript:n
1810   { /pdf.link.dict ( #1 ) def }
1811   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
1812   \__pdf_backend_link_sf_save:
1813   \mode_if_math:TF
1814   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
1815   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
1816   \hbox_set:Nw \l__pdf_backend_content_box
1817   \__pdf_backend_link_sf_restore:
1818   \bool_if:NT \g__pdf_backend_link_math_bool
1819   { \c_math_toggle_token }
1820 }
1821 \cs_new_protected:Npn \__pdf_backend_link_end:
1822 {
1823   \bool_if:NT \g__pdf_backend_link_bool
1824   { \__pdf_backend_link_end_aux: }
1825 }
1826 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
1827 {
1828   \bool_if:NT \g__pdf_backend_link_math_bool
1829   { \c_math_toggle_token }
1830   \__pdf_backend_link_sf_save:
1831   \hbox_set_end:
1832   \__pdf_backend_link_minima:
1833   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
1834   \exp_args:Nx \__pdf_backend_link_outerbox:n
1835   {

```



```

1836 <*initex>
1837     \l_galley_total_left_margin_dim
1838 </initex>
1839 <*package>
1840     \int_if_odd:nTF { \value { page } }
1841         { \oddsidemargin }
1842         { \evensidemargin }
1843 </package>
1844 }
1845 \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
1846 { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
1847 \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
1848 \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
1849 \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
1850 \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
1851 {
1852     \hbox:n
1853     { \__kernel_backend_postscript:n { pdf.save.linkur } }
1854 }
1855 \int_gincr:N \g__pdf_backend_object_int
1856 \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
1857 \__kernel_backend_postscript:x
1858 {
1859     mark
1860     /objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
1861     \g__pdf_backend_link_dict_tl \c_space_tl
1862     pdf.rect
1863     /ANN ~ \l__pdf_breaklink_pdfmark_tl
1864 }
1865 \__pdf_backend_link_sf_restore:
1866 \bool_gset_false:N \g__pdf_backend_link_bool
1867 }
1868 \cs_new_protected:Npn \__pdf_backend_link_minima:
1869 {
1870     \hbox_set:Nn \l__pdf_backend_model_box { Gg }
1871     \__kernel_backend_postscript:x
1872     {
1873         /pdf.linkdp.pad ~
1874         \dim_to_decimal:n
1875         {
1876             \dim_max:nn
1877             {
1878                 \box_dp:N \l__pdf_backend_model_box
1879                 - \box_dp:N \l__pdf_backend_content_box
1880             }
1881             { Opt }
1882         } ~
1883         pdf.pt.dvi ~ def
1884         /pdf.linkht.pad ~
1885         \dim_to_decimal:n
1886         {
1887             \dim_max:nn
1888             {
1889                 \box_ht:N \l__pdf_backend_model_box

```

```

1890         - \box_ht:N \l__pdf_backend_content_box
1891     }
1892     { Opt }
1893 } ~
1894 pdf.pt.dvi ~ def
1895 }
1896 }
1897 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
1898 {
1899     \__kernel_backend_postscript:x
1900     {
1901         /pdf.outerbox
1902         [
1903             \dim_to_decimal:n {#1} ~
1904             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
1905         (*initex)
1906             \dim_to_decimal:n { #1 + \l_galley_text_width_dim } ~
1907         (/initex)
1908         (*package)
1909             \dim_to_decimal:n { #1 + \textwidth } ~
1910         (/package)
1911             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
1912         ]
1913         [ exch { pdf.pt.dvi } forall ] def
1914         /pdf.baselineskip ~
1915         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
1916         { pdf.pt.dvi ~ def }
1917         { pop ~ pop }
1918         ifelse
1919     }
1920 }
1921 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
1922 {
1923     \int_gset:Nn \g__pdf_backend_link_sf_int
1924     {
1925         \mode_if_horizontal:TF
1926         { \tex_spacefactor:D }
1927         { 0 }
1928     }
1929 }
1930 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
1931 {
1932     \mode_if_horizontal:T
1933     {
1934         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
1935         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
1936     }
1937 }

```

(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.

```

1938 <*package>
1939 \use_none:n
1940 {
1941   \cs_if_exist:NT \@makecol@hook
1942   {
1943     \tl_put_right:Nn \@makecol@hook
1944     {
1945       \box_if_empty:NF \@cclv
1946       {
1947         \vbox_set:Nn \@cclv
1948         {
1949           \__kernel_backend_postscript:n
1950           {
1951             pdf.globaldict /pdf.brokenlink.rect ~ known
1952             { pdf.bordertracking.continue }
1953             if
1954           }
1955           \vbox_unpack_drop:N \@cclv
1956           \__kernel_backend_postscript:n
1957           { pdf.bordertracking.endpage }
1958         }
1959       }
1960     }
1961     \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
1962     \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
1963     \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
1964   }
1965 }
1966 </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.

```

1967 \cs_new:Npn \__pdf_backend_link_last:
1968 { { 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.

```

1969 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
1970 {
1971   \__kernel_backend_postscript:x
1972   {
1973     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
1974   }
1975 }

```

(End definition for __pdf_backend_link_margin:n.)

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

```

1976 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2

```

```

1977 {
1978   \_kernel_backend_postscript:n { pdf.dest.anchor }
1979   \_pdf_backend_pdfmark:x
1980   {
1981     /View
1982     [
1983       \str_case:nnF {#2}
1984       {
1985         { xyz } { /XYZ ~ pdf.dest.point ~ null }
1986         { fit } { /Fit }
1987         { fitb } { /FitB }
1988         { fitbh } { /FitBH ~ pdf.dest.y }
1989         { fitbv } { /FitBV ~ pdf.dest.x }
1990         { fith } { /FitH ~ pdf.dest.y }
1991         { fitv } { /FitV ~ pdf.dest.x }
1992       }
1993       {
1994         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
1995       }
1996     ]
1997     /Dest ( \exp_not:n {#1} ) cvn
1998     /DEST
1999   }
2000 }
2001 \cs_new_protected:Npn \_pdf_backend_destination_rectangle:nn #1#2
2002 {
2003   \group_begin:
2004     \hbox_set:Nn \l__pdf_internal_box {#2}
2005     \box_move_down:nn
2006       { \box_dp:N \l__pdf_internal_box }
2007       { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2008     \box_use:N \l__pdf_internal_box
2009     \box_move_up:nn
2010       { \box_ht:N \l__pdf_internal_box }
2011       { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ur } } }
2012     \_pdf_backend_pdfmark:n
2013     {
2014       /View
2015       [
2016         /FitR ~
2017         pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2018         pdf.urx ~ pdf.ury ~ pdf.dest2device
2019       ]
2020       /Dest ( #1 ) cvn
2021       /DEST
2022     }
2023   \group_end:
2024 }

```

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

6.2.4 Structure

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

```

2025 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2026 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1 { }
(End definition for \__pdf_backend_compresslevel:n and \__pdf_backend_compress_objects:n.)

```

```

\__pdf_backend_version_major_gset:n Data not available!
\__pdf_backend_version_minor_gset:n
2027 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2028 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1 { }
(End definition for \__pdf_backend_version_major_gset:n and \__pdf_backend_version_minor_gset:n.)

```

```

\__pdf_backend_version_major: Data not available!
\__pdf_backend_version_minor:
2029 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2030 \cs_new:Npn \__pdf_backend_version_minor: { -1 }
(End definition for \__pdf_backend_version_major: and \__pdf_backend_version_minor:.)

```

6.2.5 Marked content

```

\__pdf_backend_bdc:nn Simple wrappers.
\__pdf_backend_emc:
2031 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2032 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2033 \cs_new_protected:Npn \__pdf_backend_emc:
2034 { \__pdf_backend_pdfmark:n { /EMC } }
(End definition for \__pdf_backend_bdc:nn and \__pdf_backend_emc:.)
2035 </dvips>

```

6.3 pdfmode backend

```

2036 <*pdfmode>

```

6.3.1 Annotations

```

\__pdf_backend_annotation:nnnn Simply pass the raw data through, just dealing with evaluation of dimensions.
2037 \cs_new_protected:Npx \__pdf_backend_annotation:nnnn #1#2#3#4
2038 {
2039   \cs_if_exist:NTF \tex_pdfextension:D
2040   { \tex_pdfextension:D annot ~ }
2041   { \tex_pdfannot:D }
2042   width ~ \exp_not:N \dim_eval:n {#1} ~
2043   height ~ \exp_not:N \dim_eval:n {#2} ~
2044   depth ~ \exp_not:N \dim_eval:n {#3} ~
2045   {#4}
2046 }
(End definition for \__pdf_backend_annotation:nnnn.)

```

```

\__pdf_backend_annotation_last: A tiny amount of extra data gets added here.
2047 \cs_new:Npx \__pdf_backend_annotation_last:
2048 {
2049   \exp_not:N \int_value:w
2050   \cs_if_exist:NTF \tex_pdffeedback:D
2051   { \exp_not:N \tex_pdffeedback:D lastannot ~ }
2052   { \exp_not:N \tex_pdflastannot:D }
2053   \c_space_tl 0 ~ R
2054 }

```

(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:
2055 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2056 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2057 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2058 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2059 \cs_new_protected:Npx \_pdf_backend_link_begin:nnnw #1#2#3
2060 {
2061   \cs_if_exist:NTF \tex_pdfextension:D
2062   { \tex_pdfextension:D startlink ~ }
2063   { \tex_pdfstartlink:D }
2064   attr {#1}
2065   #2 {#3}
2066 }
2067 \cs_new_protected:Npx \_pdf_backend_link_end:
2068 {
2069   \cs_if_exist:NTF \tex_pdfextension:D
2070   { \tex_pdfextension:D endlink \scan_stop: }
2071   { \tex_pdfendlink:D }
2072 }

```

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

`_pdf_backend_link_last:` Formatted for direct use.

```

2073 \cs_new:Npx \_pdf_backend_link_last:
2074 {
2075   \exp_not:N \int_value:w
2076   \cs_if_exist:NTF \tex_pdffeedback:D
2077   { \exp_not:N \tex_pdffeedback:D lastlink ~ }
2078   { \exp_not:N \tex_pdflastlink:D }
2079   \c_space_tl 0 ~ R
2080 }

```

(End definition for `_pdf_backend_link_last:`)

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

```

2081 \cs_new_protected:Npx \_pdf_backend_link_margin:n #1
2082 {
2083   \cs_if_exist:NTF \tex_pdfvariable:D
2084   { \exp_not:N \tex_pdfvariable:D linkmargin }
2085   { \exp_not:N \tex_pdflinkmargin:D }
2086   \exp_not:N \dim_eval:n {#1} \scan_stop:
2087 }

```

(End definition for `_pdf_backend_link_margin:n`)

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

```

2088 \cs_new_protected:Npx \_pdf_backend_destination:nn #1#2
2089 {
2090   \cs_if_exist:NTF \tex_pdfextension:D
2091   { \exp_not:N \tex_pdfextension:D dest ~ }

```

```

2092 { \exp_not:N \tex_pdfdest:D }
2093   name {#1}
2094   \exp_not:N \str_case:nnF {#2}
2095   {
2096     { xyz } { xyz }
2097     { fit } { fit }
2098     { fitb } { fitb }
2099     { fitbh } { fitbh }
2100     { fitbv } { fitbv }
2101     { fith } { fith }
2102     { fitv } { fitv }
2103   }
2104   { xyz ~ zoom \exp_not:N \fp_eval:n { #2 * 10 } }
2105   \scan_stop:
2106 }
2107 \cs_new_protected:Npx \__pdf_backend_destination_rectangle:nn #1#2
2108 {
2109   \group_begin:
2110   \hbox_set:Nn \l__pdf_internal_box {#2}
2111   \cs_if_exist:NTF \tex_pdfextension:D
2112   { \exp_not:N \tex_pdfextension:D dest ~ }
2113   { \exp_not:N \tex_pdfdest:D }
2114   name {#1}
2115   fitr ~
2116   width \exp_not:N \box_wd:N \l__pdf_internal_box
2117   height \exp_not:N \box_ht:N \l__pdf_internal_box
2118   depth \exp_not:N \box_dp:N \l__pdf_internal_box
2119   \box_use:N \l__pdf_internal_box
2120   \group_end:
2121 }

```

(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
2122 \cs_new_protected:Npx \__pdf_backend_catalog_gput:nn #1#2
2123 {
2124   \cs_if_exist:NTF \tex_pdfextension:D
2125   { \tex_pdfextension:D catalog }
2126   { \tex_pdfcatalog:D }
2127   { / #1 ~ #2 }
2128 }
2129 \cs_new_protected:Npx \__pdf_backend_info_gput:nn #1#2
2130 {
2131   \cs_if_exist:NTF \tex_pdfextension:D
2132   { \tex_pdfextension:D info }
2133   { \tex_pdfinfo:D }
2134   { / #1 ~ #2 }
2135 }

```

(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.

2136 `\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`

```

2137 \group_begin:
2138 \cs_set_protected:Npn \__pdf_tmp:w #1#2
2139 {
2140   \cs_new_protected:Npx \__pdf_backend_object_new:nn ##1##2
2141   {
2142     #1 reserveobjnum ~
2143     \int_const:cn
2144     { c__pdf_backend_object_ \exp_not:N \tl_to_str:n {##1} _int }
2145     {#2}
2146     \prop_gput:Nnn \exp_not:N \g__pdf_backend_object_prop {##1} {##2}
2147   }
2148 }
2149 \cs_if_exist:NTF \tex_pdfextension:D
2150 {
2151   \__pdf_tmp:w
2152   { \tex_pdfextension:D obj ~ }
2153   { \exp_not:N \tex_pdffeedback:D lastobj }
2154 }
2155 { \__pdf_tmp:w { \tex_pdfobj:D } { \tex_pdflastobj:D } }
2156 \group_end:
2157 \cs_new:Npn \__pdf_backend_object_ref:n #1
2158 { \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

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

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

```

\__pdf_backend_object_write:nx
\__pdf_exp_not_i:nn
\__pdf_exp_not_ii:nn
2159 \group_begin:
2160 \cs_set_protected:Npn \__pdf_tmp:w #1
2161 {
2162   \cs_new_protected:Npn \__pdf_backend_object_write:nn ##1##2
2163   {
2164     \tex_immediate:D #1 useobjnum ~
2165     \int_use:c
2166     { c__pdf_backend_object_ \tl_to_str:n {##1} _int }
2167     \str_case_e:nn
2168     { \prop_item:Nn \g__pdf_backend_object_prop {##1} }
2169     {
2170       { array } { { [ ~ \exp_not:n {##2} ~ ] } }
2171       { dict } { { < ~ \exp_not:n {##2} ~ > } }
2172       { fstream }
2173       {
2174         stream ~ attr ~ { \__pdf_exp_not_i:nn ##2 } ~
2175         file ~ { \__pdf_exp_not_ii:nn ##2 }
2176       }
2177     } stream }
2178     {
2179       stream ~ attr ~ { \__pdf_exp_not_i:nn ##2 } ~

```



```

2180         { \_pdf_exp_not_ii:nn ##2 }
2181     }
2182 }
2183 }
2184 }
2185 \cs_if_exist:NTF \tex_pdfextension:D
2186 { \_pdf_tmp:w { \tex_pdfextension:D obj ~ } }
2187 { \_pdf_tmp:w { \tex_pdfobj:D } }
2188 \group_end:
2189 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
2190 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2191 \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
2192 \group_begin:
2193 \cs_set_protected:Npn \_pdf_tmp:w #1
2194 {
2195     \cs_new_protected:Npn \_pdf_backend_object_now:nn ##1##2
2196     {
2197         \tex_immediate:D #1
2198         \str_case:nn
2199         {##1}
2200         {
2201             { array } { { [ ~ \exp_not:n {##2} ~ ] } }
2202             { dict } { { << ~ \exp_not:n {##2} ~ >> } }
2203             { fstream }
2204             {
2205                 stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2206                 file ~ { \_pdf_exp_not_ii:nn ##2 }
2207             }
2208             { stream }
2209             {
2210                 stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2211                 { \_pdf_exp_not_ii:nn ##2 }
2212             }
2213         }
2214     }
2215 }
2216 \cs_if_exist:NTF \tex_pdfextension:D
2217 { \_pdf_tmp:w { \tex_pdfextension:D obj ~ } }
2218 { \_pdf_tmp:w { \tex_pdfobj:D } }
2219 \group_end:
2220 \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.

```

2221 \cs_new:Npx \_pdf_backend_object_last:
2222 {
2223     \exp_not:N \int_value:w
2224     \cs_if_exist:NTF \tex_pdffeedback:D
2225     { \exp_not:N \tex_pdffeedback:D lastobj ~ }

```

```

2226     { \exp_not:N \tex_pdflastobj:D }
2227     \c_space_tl 0 ~ R
2228   }

```

(End definition for `__pdf_backend_object_last:.`)

6.3.4 Structure

Simply pass data to the engine.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
\__pdf_backend_objcompresslevel:n
2229 \cs_new_protected:Npx \__pdf_backend_compresslevel:n #1
2230 {
2231   \exp_not:N \tex_global:D
2232   \cs_if_exist:NTF \tex_pdfcompresslevel:D
2233   { \tex_pdfcompresslevel:D }
2234   { \tex_pdfvariable:D compresslevel }
2235   \exp_not:N \int_value:w \exp_not:N \int_eval:n {#1} \scan_stop:
2236 }
2237 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2238 {
2239   \bool_if:nTF {#1}
2240   { \__pdf_backend_objcompresslevel:n { 2 } }
2241   { \__pdf_backend_objcompresslevel:n { 0 } }
2242 }
2243 \cs_new_protected:Npx \__pdf_backend_objcompresslevel:n #1
2244 {
2245   \exp_not:N \tex_global:D
2246   \cs_if_exist:NTF \tex_pdfobjcompresslevel:D
2247   { \tex_pdfobjcompresslevel:D }
2248   { \tex_pdfvariable:D objcompresslevel }
2249   #1 \scan_stop:
2250 }

```

(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
2251 \cs_new_protected:Npx \__pdf_backend_version_major_gset:n #1
2252 {
2253   \cs_if_exist:NTF \tex_pdfvariable:D
2254   {
2255     \int_compare:nNnT \tex luatexversion:D > { 106 }
2256     {
2257       \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2258       \exp_not:N \int_eval:n {#1} \scan_stop:
2259     }
2260   }
2261   {
2262     \cs_if_exist:NT \tex_pdfmajorversion:D
2263     {
2264       \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2265       \exp_not:N \int_eval:n {#1} \scan_stop:
2266     }
2267   }

```

```

2268 }
2269 \cs_new_protected:Npx \__pdf_backend_version_minor_gset:n #1
2270 {
2271   \exp_not:N \tex_global:D
2272   \cs_if_exist:NTF \tex_pdfminorversion:D
2273   { \exp_not:N \tex_pdfminorversion:D }
2274   { \tex_pdfvariable:D minorversion }
2275   \exp_not:N \int_eval:n {#1} \scan_stop:
2276 }

```

(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: 2277 \cs_new:Npx \__pdf_backend_version_major:
2278 {
2279   \cs_if_exist:NTF \tex_pdfvariable:D
2280   {
2281     \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2282     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2283     { 1 }
2284   }
2285   {
2286     \cs_if_exist:NTF \tex_pdfmajorversion:D
2287     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2288     { 1 }
2289   }
2290 }
2291 \cs_new:Npx \__pdf_backend_version_minor:
2292 {
2293   \exp_not:N \tex_the:D
2294   \cs_if_exist:NTF \tex_pdfminorversion:D
2295   { \exp_not:N \tex_pdfminorversion:D }
2296   { \tex_pdfvariable:D minorversion }
2297 }

```

(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>.

```

2298 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2299 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2300 \cs_new_protected:Npn \__pdf_backend_emc:
2301 { \__kernel_backend_literal_page:n { EMC } }

```

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

2302 </pdfmode>

6.4 dvipdfmx backend

```

2303 <*dvipdfmx | xdvipdfmx>

\__pdf_backend:n A generic function for the backend PDF specials: used where we can.
\__pdf_backend:x
2304 \cs_new_protected:Npx \__pdf_backend:n #1
2305 { \__kernel_backend_literal:n { pdf: #1 } }
2306 \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
2307 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2308 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2309 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2310 { \__pdf_backend:n { docinfo << /#1 ~ #2 >> } }

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

```

6.4.2 Objects

```

\g__pdf_backend_object_int For tracking objects to allow finalisation.
\g__pdf_backend_object_prop
2311 \int_new:N \g__pdf_backend_object_int
2312 \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
2313 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2314 {
2315   \int_gincr:N \g__pdf_backend_object_int
2316   \int_const:cn
2317   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2318   { \g__pdf_backend_object_int }
2319   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2320 }
2321 \cs_new:Npn \__pdf_backend_object_ref:n #1
2322 { @pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } }

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

```

```

\__pdf_backend_object_write:nn This is where we choose the actual type.
\__pdf_backend_object_write:nx
\__pdf_backend_object_write:nnn
\__pdf_backend_object_write_array:nn
\__pdf_backend_object_write_dict:nn
\__pdf_backend_object_write_fstream:nn
\__pdf_backend_object_write_stream:nn
\__pdf_backend_object_write_stream:nnn
2323 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2324 {
2325   \exp_args:Nx \__pdf_backend_object_write:nnn
2326   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2327 }
2328 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2329 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2330 {
2331   \use:c { \__pdf_backend_object_write_ #1 :nn }
2332   { \__pdf_backend_object_ref:n {#2} } {#3}
2333 }

```

```

2334 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2335 {
2336   \__pdf_backend:x
2337   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2338 }
2339 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2340 {
2341   \__pdf_backend:x
2342   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2343 }
2344 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2345 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2346 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2347 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2348 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2349 {
2350   \__pdf_backend:x
2351   {
2352     #1 stream ~ #2 ~
2353     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2354   }
2355 }

```

(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
2356 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2357 {
2358   \int_gincr:N \g__pdf_backend_object_int
2359   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2360   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2361   {#2}
2362 }
2363 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2364 \cs_new:Npn \__pdf_backend_object_last:
2365 { @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.

```

2366 \bool_new:N \g__pdf_landscape_bool
2367 <*package>
2368 \AtBeginDocument
2369 {
2370   \cs_if_exist:NT \landscape
2371   {
2372     \tl_put_right:Nn \landscape

```

```

2373         { \bool_gset_true:N \g__pdf_landscape_bool }
2374     \tl_put_left:Nn \endlandscape
2375         { \bool_gset_false:N \g__pdf_landscape_bool }
2376     }
2377 }
2378 \</package>

```

(End definition for \g__pdf_landscape_bool.)

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

```

2379 \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.

```

2380 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2381 {
2382     \bool_if:NTF \g__pdf_landscape_bool
2383     {
2384         \box_move_up:nn {#2}
2385         {
2386             \vbox:n
2387             {
2388                 \__pdf_backend_annotation_aux:nnnn
2389                 { #2 + #3 } {#1} { Opt } {#4}
2390             }
2391         }
2392     }
2393     { \__pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4} }
2394 }
2395 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2396 {
2397     \int_gincr:N \g__pdf_backend_object_int
2398     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2399     \__pdf_backend:x
2400     {
2401         ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
2402         width ~ \dim_eval:n {#1} ~
2403         height ~ \dim_eval:n {#2} ~
2404         depth ~ \dim_eval:n {#3} ~
2405         <</Type/Annot #4 >>
2406     }
2407 }

```

(End definition for __pdf_backend_annotation:nnnn and __pdf_backend_annotation_aux:nnnn.)

__pdf_backend_annotation_last:

```

2408 \cs_new:Npn \__pdf_backend_annotation_last:
2409 { @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.

```

2410 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2411 { \__pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2412 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
2413 { \__pdf_backend_link_begin:n {#1#2} }
2414 \cs_new_protected:Npn \__pdf_backend_link_begin:n #1
2415 {
2416   \__pdf_backend:n
2417   {
2418     bann
2419     <<
2420     /Type /Annot
2421     #1
2422     >>
2423   }
2424 }
2425 \cs_new_protected:Npn \__pdf_backend_link_end:
2426 { \__pdf_backend:n { eann } }

```

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

```

\__pdf_backend_link_last:

```

Data not available.

```

2427 \cs_new:Npn \__pdf_backend_link_last: { }

```

(End definition for `__pdf_backend_link_last:`.)

```

\__pdf_backend_link_margin:n

```

Pass to `dvipdfmx`.

```

2428 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2429 { \__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`.

```

2430 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2431 {
2432   \__pdf_backend:x
2433   {
2434     dest ~ ( \exp_not:n {#1} )
2435     [
2436       @thispage
2437       \str_case:nnF {#2}
2438       {
2439         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2440         { fit } { /Fit }
2441         { fitb } { /FitB }
2442         { fitbh } { /FitBH }
2443         { fitbv } { /FitBV ~ @xpos }
2444         { fith } { /FitH ~ @ypos }
2445         { fitv } { /FitV ~ @xpos }
2446       }
2447       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2448     ]

```

```

2449     }
2450   }
2451   \cs_new_protected:Npn \__pdf_backend_destination_rectangle:nn #1#2
2452   {
2453     \group_begin:
2454     \hbox_set:Nn \l__pdf_internal_box {#2}
2455     \box_move_down:nn { \box_dp:N \l__pdf_internal_box }
2456     {
2457       \hbox:n
2458       {
2459         \__pdf_backend:n { obj ~ @pdf_ #1 _llx ~ @xpos }
2460         \__pdf_backend:n { obj ~ @pdf_ #1 _lly ~ @ypos }
2461       }
2462     }
2463     \box_use:N \l__pdf_internal_box
2464     \box_move_up:nn { \box_ht:N \l__pdf_internal_box }
2465     {
2466       \hbox:n
2467       {
2468         \__pdf_backend:n
2469         {
2470           dest ~ (#1)
2471           [
2472             @thispage
2473             /FitR ~
2474             @pdf_ #1 _llx ~ @pdf_ #1 _lly ~
2475             @xpos ~ @ypos
2476           ]
2477         }
2478       }
2479     }
2480   \group_end:
2481 }

```

(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
2482 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2483 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2484 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2485 {
2486   \bool_if:nF {#1}
2487   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2488 }

```

(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
2489 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2490 {
2491   \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
2492   \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }

```



```

2493 }
2494 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2495 {
2496   \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
2497   \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
2498 }

```

(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:
2499 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2500 \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:

```

2501 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2502 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2503 \cs_new_protected:Npn \__pdf_backend_emc:
2504 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2505 </dvipdfmx | xdvipdfmx>

```

6.5 dvisvgm backend

```

2506 <*dvisvgm>

```

6.5.1 Catalogue entries

__pdf_backend_catalog_gput:nn No-op.

__pdf_backend_info_gput:nn

```

2507 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2 { }
2508 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2 { }

```

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

6.5.2 Objects

__pdf_backend_object_new:nn All no-ops here.

__pdf_backend_object_ref:n

__pdf_backend_object_write:nn

__pdf_backend_object_write:nx

__pdf_backend_object_now:nn

__pdf_backend_object_now:nx

__pdf_backend_object_last:

```

2509 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2 { }
2510 \cs_new:Npn \__pdf_backend_object_ref:n #1 { }
2511 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2 { }
2512 \cs_new_protected:Npn \__pdf_backend_object_write:nx #1#2 { }
2513 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2 { }
2514 \cs_new_protected:Npn \__pdf_backend_object_now:nx #1#2 { }
2515 \cs_new:Npn \__pdf_backend_object_last: { }

```

(End definition for __pdf_backend_object_new:nn and others.)

6.5.3 Structure

```

\_pdf_backend_compresslevel:n These are all no-ops.
\_pdf_backend_compress_objects:n
2516 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2517 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1 { }

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

\_pdf_backend_version_major_gset:n Data not available!
\_pdf_backend_version_minor_gset:n
2518 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2519 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1 { }

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

\_pdf_backend_version_major: Data not available!
\_pdf_backend_version_minor:
2520 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2521 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

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

\_pdf_backend_bdc:nn More no-ops.
\_pdf_backend_emc:
2522 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2523 \cs_new_protected:Npn \__pdf_backend_emc: { }

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

2524 </dvisvgm>
2525 </initex | package>

```

7 l3backend-header Implementation

```

2526 <*dvips & header>

pdf.globaldict A small global dictionary for backend use.
2527 true setglobal
2528 /pdf.globaldict 4 dict def
2529 false setglobal

(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
2530
2531 /pdf.cvs { 65534 string cvs } def
2532 /pdf.dvi.pt { 72.27 mul Resolution div } def
2533 /pdf.pt.dvi { 72.27 div Resolution mul } def
2534 /pdf.rect.ht { dup 1 get neg exch 3 get add } def

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

pdf.linkmargin Settings which are defined up-front in SDict.
pdf.linkdp.pad
2535 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad
2536 /pdf.linkdp.pad { 0 } def
2537 /pdf.linkht.pad { 0 } def

```

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

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

```
pdf.save.linkll 2538 /pdf.rect
pdf.save.linkur 2539 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.llx 2540 /pdf.save.ll
pdf.lly 2541 {
pdf.urx 2542     currentpoint
pdf.ury 2543     /pdf.lly exch def
2544     /pdf.llx exch def
2545 }
2546 def
2547 /pdf.save.ur
2548 {
2549     currentpoint
2550     /pdf.ury exch def
2551     /pdf.urx exch def
2552 }
2553 def
2554 /pdf.save.linkll
2555 {
2556     currentpoint
2557     pdf.linkmargin add
2558     pdf.linkdp.pad add
2559     /pdf.lly exch def
2560     pdf.linkmargin sub
2561     /pdf.llx exch def
2562 }
2563 def
2564 /pdf.save.linkur
2565 {
2566     currentpoint
2567     pdf.linkmargin sub
2568     pdf.linkht.pad sub
2569     /pdf.ury exch def
2570     pdf.linkmargin add
2571     /pdf.urx exch def
2572 }
2573 def
```

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

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

```
pdf.dev.x 2574 /pdf.dest.anchor
pdf.dev.y 2575 {
pdf.tmpa 2576     currentpoint exch
pdf.tmpb 2577     pdf.dvi.pt 72 add
pdf.tmpc
pdf.tmpd
```

```

2578     /pdf.dest.x exch def
2579     pdf.dvi.pt
2580     vsize 72 sub exch sub
2581     /pdf.dest.y exch def
2582   }
2583   def
2584 /pdf.dest.point
2585   { pdf.dest.x pdf.dest.y } def
2586 /pdf.dest2device
2587   {
2588     /pdf.dest.y exch def
2589     /pdf.dest.x exch def
2590     matrix currentmatrix
2591     matrix defaultmatrix
2592     matrix invertmatrix
2593     matrix concatmatrix
2594     cvx exec
2595     /pdf.dev.y exch def
2596     /pdf.dev.x exch def
2597     /pdf.tmpd exch def
2598     /pdf.tmpc exch def
2599     /pdf.tmpb exch def
2600     /pdf.tmpa exch def
2601     pdf.dest.x pdf.tmpa mul
2602       pdf.dest.y pdf.tmpc mul add
2603       pdf.dev.x add
2604     pdf.dest.x pdf.tmpb mul
2605       pdf.dest.y pdf.tmpd mul add
2606       pdf.dev.y add
2607   }
2608   def

```

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

pdf.bordertracking	To know where a breakable link can go, we need to track the boundary rectangle. That
pdf.bordertracking.begin	can be done by hooking into a and x operations: those names have to be retained. The
pdf.bordertracking.end	boundary is stored at the end of the operation. Special effort is needed at the start and
pdf.leftboundary	end of pages (or rather galleys), such that everything works properly.
pdf.rightboundary	
pdf.brokenlink.rect	2609 /pdf.bordertracking false def
pdf.brokenlink.skip	2610 /pdf.bordertracking.begin
pdf.brokenlink.dict	2611 {
pdf.bordertracking.endpage	2612 SDict /pdf.bordertracking true put
pdf.bordertracking.continue	2613 SDict /pdf.leftboundary undef
pdf.originx	2614 SDict /pdf.rightboundary undef
pdf.originy	2615 /a where
	2616 {
	2617 /a
	2618 {
	2619 currentpoint pop
	2620 SDict /pdf.rightboundary known dup
	2621 {
	2622 SDict /pdf.rightboundary get 2 index lt
	2623 { not }
	2624 if

```

2625     }
2626     if
2627     { pop }
2628     { SDict exch /pdf.rightboundary exch put }
2629     ifelse
2630     moveto
2631     currentpoint pop
2632     SDict /pdf.leftboundary known dup
2633     {
2634         SDict /pdf.leftboundary get 2 index gt
2635         { not }
2636         if
2637     }
2638     if
2639     { pop }
2640     { SDict exch /pdf.leftboundary exch put }
2641     ifelse
2642 }
2643 put
2644 }
2645 if
2646 }
2647 def
2648 /pdf.bordertracking.end
2649 {
2650     /a where { /a { moveto } put } if
2651     /x where { /x { 0 exch rmoveto } put } if
2652     SDict /pdf.leftboundary known
2653     { pdf.outerbox 0 pdf.leftboundary put }
2654     if
2655     SDict /pdf.rightboundary known
2656     { pdf.outerbox 2 pdf.rightboundary put }
2657     if
2658     SDict /pdf.bordertracking false put
2659 }
2660 def
2661 /pdf.bordertracking.endpage
2662 {
2663     pdf.bordertracking
2664     {
2665         pdf.bordertracking.end
2666         true setglobal
2667         pdf.globaldict
2668         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
2669         pdf.globaldict
2670         /pdf.brokenlink.skip pdf.baselineskip put
2671         pdf.globaldict
2672         /pdf.brokenlink.dict
2673         pdf.link.dict pdf.cvs put
2674         false setglobal
2675         mark pdf.link.dict cvx exec /Rect
2676         [
2677             pdf.llx
2678             pdf.lly

```

```

2679         pdf.outerbox 2 get pdf.linkmargin add
2680         currentpoint exch pop
2681         pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
2682     ]
2683     /ANN pdf.pdfmark
2684 }
2685 if
2686 }
2687 def
2688 /pdf.bordertracking.continue
2689 {
2690     /pdf.link.dict pdf.globaldict
2691     /pdf.brokenlink.dict get def
2692     /pdf.outerbox pdf.globaldict
2693     /pdf.brokenlink.rect get def
2694     /pdf.baselineskip pdf.globaldict
2695     /pdf.brokenlink.skip get def
2696     pdf.globaldict dup dup
2697     /pdf.brokenlink.dict undef
2698     /pdf.brokenlink.skip undef
2699     /pdf.brokenlink.rect undef
2700     currentpoint
2701     /pdf.originy exch def
2702     /pdf.originx exch def
2703     /a where
2704     {
2705         /a
2706         {
2707             moveto
2708             SDict
2709             begin
2710                 currentpoint pdf.originy ne exch
2711                 pdf.originx ne or
2712                 {
2713                     pdf.save.linkll
2714                     /pdf.lly
2715                     pdf.lly pdf.outerbox 1 get sub def
2716                     pdf.bordertracking.begin
2717                 }
2718                 if
2719                 end
2720             }
2721             put
2722         }
2723     if
2724     /x where
2725     {
2726         /x
2727         {
2728             0 exch rmoveto
2729             SDict~
2730             begin
2731                 currentpoint
2732                 pdf.originy ne exch pdf.originx ne or

```

```

2733         {
2734             pdf.save.linkll
2735             /pdf.lly
2736             pdf.lly pdf.outerbox 1 get sub def
2737             pdf.bordertracking.begin
2738         }
2739         if
2740         end
2741     }
2742     put
2743 }
2744 if
2745 }
2746 def

```

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

<p>pdf.breaklink</p> <p>pdf.breaklink.write</p> <p>pdf.count</p> <p>pdf.currentrect</p>	<p>Dealing with link breaking itself has multiple stage. The first step is to find the Rect entry in the dictionary, looping over key-value pairs. The first line is handled first, adjusting the rectangle to stay inside the text area. The second phase is a loop over the height of the bulk of the link area, done on the basis of a number of baselines. Finally, the end of the link area is tidied up, again from the boundary of the text area.</p>
---	--

```

2747 /pdf.breaklink
2748 {
2749     pop
2750     counttomark 2 mod 0 eq
2751     {
2752         counttomark /pdf.count exch def
2753         {
2754             pdf.count 0 eq { exit } if
2755             counttomark 2 roll
2756             1 index /Rect eq
2757             {
2758                 dup 4 array copy
2759                 dup dup
2760                 1 get
2761                 pdf.outerbox pdf.rect.ht
2762                 pdf.linkmargin 2 mul add sub
2763                 3 exch put
2764                 dup
2765                 pdf.outerbox 2 get
2766                 pdf.linkmargin add
2767                 2 exch put
2768                 dup dup
2769                 3 get
2770                 pdf.outerbox pdf.rect.ht
2771                 pdf.linkmargin 2 mul add add
2772                 1 exch put
2773             } /pdf.currentrect exch def
2774             pdf.breaklink.write
2775             {
2776                 pdf.currentrect
2777                 dup
2778                 pdf.outerbox 0 get

```

```

2779         pdf.linkmargin sub
2780         0 exch put
2781     dup
2782         pdf.outerbox 2 get
2783         pdf.linkmargin add
2784         2 exch put
2785     dup dup
2786         1 get
2787         pdf.baselineskip add
2788         1 exch put
2789     dup dup
2790         3 get
2791         pdf.baselineskip add
2792         3 exch put
2793     /pdf.currentrect exch def
2794     pdf.breaklink.write
2795 }
2796 1 index 3 get
2797 pdf.linkmargin 2 mul add
2798 pdf.outerbox pdf.rect.ht add
2799 2 index 1 get sub
2800 pdf.baselineskip div round cvi 1 sub
2801 exch
2802 repeat
2803 pdf.currentrect
2804 dup
2805     pdf.outerbox 0 get
2806     pdf.linkmargin sub
2807     0 exch put
2808     dup dup
2809         1 get
2810         pdf.baselineskip add
2811         1 exch put
2812     dup dup
2813         3 get
2814         pdf.baselineskip add
2815         3 exch put
2816     dup 2 index 2 get 2 exch put
2817     /pdf.currentrect exch def
2818     pdf.breaklink.write
2819     SDict /pdf.pdfmark.good false put
2820     exit
2821 }
2822 { pdf.count 2 sub /pdf.count exch def }
2823 ifelse
2824 }
2825 loop
2826 }
2827 if
2828 /ANN
2829 }
2830 def
2831 /pdf.breaklink.write
2832 {

```



```

2833     counttomark 1 sub
2834     index /_objdef eq
2835     {
2836         counttomark -2 roll
2837         dup wcheck
2838         {
2839             readonly
2840             counttomark 2 roll
2841         }
2842         { pop pop }
2843         ifelse
2844     }
2845     if
2846     counttomark 1 add copy
2847     pop pdf.currentrect
2848     /ANN pdfmark
2849 }
2850 def

```

(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.

```

2851 /pdf.pdfmark
2852 {
2853     SDict /pdf.pdfmark.good true put
2854     dup /ANN eq
2855     {
2856         pdf.pdfmark.store
2857         pdf.pdfmark.dict
2858         begin
2859             Subtype /Link eq
2860             currentdict /Rect known and
2861             SDict /pdf.outerbox known and
2862             SDict /pdf.baselineskip known and
2863             {
2864                 Rect 3 get
2865                 pdf.linkmargin 2 mul add
2866                 pdf.outerbox pdf.rect.ht add
2867                 Rect 1 get sub
2868                 pdf.baselineskip div round cvi 0 gt
2869                 { pdf.breaklink }
2870                 if
2871             }
2872             if
2873             end
2874             SDict /pdf.outerbox undef
2875             SDict /pdf.baselineskip undef
2876             currentdict /pdf.pdfmark.dict undef
2877         }
2878         if

```

```

2879     pdf.pdfmark.good
2880     { pdfmark }
2881     { cleartomark }
2882     ifelse
2883   }
2884   def
2885 /pdf.pdfmark.store
2886 {
2887   /pdf.pdfmark.dict 65534 dict def
2888   counttomark 1 add copy
2889   pop
2890   {
2891     dup mark eq
2892     {
2893       pop
2894       exit
2895     }
2896     {
2897       pdf.pdfmark.dict
2898       begin def end
2899     }
2900     ifelse
2901   }
2902   loop
2903 }
2904 def

```

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

```

2905 </dvips & header>

```

Index

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

- A**
- `\AtBeginDocument` 369, 426, 1311, 1446, 1610, 2368
 - `\AtBeginDvi` 36, 37
- B**
- `\begin` 1360, 1365
 - bool commands:
 - `\bool_gset_false:N` 572, 588, 614, 636, 652, 804, 1130, 1166, 1815, 1866, 2375
 - `\bool_gset_true:N` 570, 639, 802, 1145, 1808, 1814, 2373
 - `\bool_if:NTF` 579, 583, 601, 605, 609, 622, 627, 631, 643, 647, 815, 820, 825, 1104, 1149, 1336, 1377, 1493, 1535, 1803, 1818, 1823, 1828, 2382
 - `\bool_if:nTF` 2239, 2486
 - `\bool_lazy_or:nnTF` 1369, 1528
 - `\bool_new:N` 573, 640, 805, 1146, 1791, 1792, 2366
 - `\bool_set_false:N` 1346, 1460, 1553, 1623
 - box commands:
 - `\box_dp:N` 132, 134, 182, 184, 239, 241, 288, 290, 292, 294, 1845, 1878, 1879, 1904, 2006, 2118, 2455
 - `\box_ht:N` 134, 184, 241, 292, 294, 1389, 1590, 1850, 1889, 1890, 1911, 2010, 2117, 2464
 - `\box_if_empty:NTF` 1945
 - `\box_move_down:nn` 1766, 1845, 2005, 2455
 - `\box_move_up:nn` 1769, 1850, 2009, 2384, 2464
 - `\box_new:N` 1658, 1747, 1748
 - `\box_set_dp:Nn` 1297
 - `\box_set_ht:Nn` 1296
 - `\box_set_wd:Nn` 196, 1295
 - `\box_use:N` . 139, 157, 171, 187, 214, 228, 244, 260, 272, 323, 340, 359, 755, 1012, 1298, 1796, 2008, 2119, 2463
 - `\box_wd:N` .. 133, 141, 183, 189, 240, 246, 289, 291, 327, 1388, 1589, 2116
 - box internal commands:
 - `__box_backend_clip:N` 121, 176, 233, 277
 - `\l__box_backend_cos_fp` 191
 - `__box_backend_rotate:Nn` 143, 191, 248, 330
 - `__box_backend_rotate_aux:Nn` ... 143, 191, 248
 - `__box_backend_scale:Nnn` 160, 219, 263, 343
 - `\l__box_backend_sin_fp` 191
 - `\g__box_clip_path_int` 277
- C**
- clist commands:
 - `\clist_map_function:nN` 660, 835
 - `\clist_map_function:nn` 1173
 - color internal commands:
 - `__color_backend_cmyk:nnnn` . 393, 462
 - `__color_backend_cmyk_aux:nnnn` . 462
 - `__color_backend_gray:n` 393, 462
 - `__color_backend_gray_aux:n` ... 462
 - `__color_backend_pickup:N` .. 367, 424
 - `__color_backend_pickup:w` 13, 367, 424
 - `__color_backend_reset:` 393, 462
 - `__color_backend_rgb:nnn` ... 393, 462
 - `__color_backend_rgb_aux:nnn` .. 462
 - `__color_backend_select:n` .. 393, 462
 - `__color_backend_spot:nn` ... 393, 462
 - `color.fc` 393, 519
 - cs commands:
 - `\cs_generate_variant:Nn` 28, 32, 35, 64, 92, 97, 108, 115, 419, 504, 518, 723, 729, 765, 913, 1021, 1052, 1507, 1564, 1580, 1662, 1699, 1744, 2189, 2220, 2306, 2328, 2363
 - `\cs_gset:Npx` 2491, 2496
 - `\cs_if_exist:NTF` ... 36, 59, 67, 75, 81, 87, 371, 428, 498, 507, 896, 904, 1941, 2039, 2050, 2061, 2069, 2076, 2083, 2090, 2111, 2124, 2131, 2149, 2185, 2216, 2224, 2232, 2246, 2253, 2262, 2272, 2279, 2286, 2294, 2370
 - `\cs_new:Npn` 665, 840, 1177, 1593, 1602, 1652, 1677, 1745, 1786, 1967, 2029, 2030, 2157, 2190, 2191, 2321, 2364, 2408, 2427, 2499, 2500, 2510, 2515, 2520, 2521
 - `\cs_new:Npx` 2047, 2073, 2221, 2277, 2291
 - `\cs_new_eq:NN` 25, 517, 764, 770, 771, 911, 1020,

1313, 1342, 1399, 1400, 1448, 1456,
1478, 1549, 1612, 1619, 1651, 1796

D

dim commands:

\dim_eval:n 1773, 2042, 2043,
2044, 2086, 2402, 2403, 2404, 2429
\dim_max:nn 1876, 1887
\dim_set:Nn 1388, 1389, 1589, 1590
\dim_to_decimal:n 288, 289, 290, 291,
292, 294, 1058, 1063, 1069, 1070,
1071, 1072, 1081, 1082, 1083, 1174,
1193, 1646, 1647, 1874, 1885, 1903,
1904, 1906, 1909, 1911, 1915, 1973
\dim_to_decimal_in_bp:n . 132, 133,
134, 182, 183, 184, 239, 240, 241,
538, 539, 546, 547, 554, 555, 563,
564, 565, 662, 666, 670, 775, 780,
786, 787, 788, 796, 797, 837, 841,
845, 1178, 1319, 1320, 1321, 1322,
1470, 1471, 1472, 1473, 1522, 1523,
1524, 1525, 1635, 1636, 1637, 1638

draw internal commands:

__draw_align_currentpoint... . . 21
__draw_backend_add_to_path:n . . .
. 1055, 1101
__draw_backend_begin: 519, 766, 1022
__draw_backend_box_use:Nnnnn . . .
. 16, 742, 997, 1284
__draw_backend_cap_but:
. 654, 829, 1168
__draw_backend_cap_rectangle: . . .
. 654, 829, 1168
__draw_backend_cap_round:
. 654, 829, 1168
__draw_backend_clip: 574, 806, 1100
__draw_backend_closepath:
. 574, 806, 1100
__draw_backend_closestroke: . . .
. 574, 806, 1100
__draw_backend_cm:nnnn 730,
750, 751, 752, 914, 1001, 1269, 1287
__draw_backend_cm_aux:nnnn 914
__draw_backend_cm_decompose:nnnnN
. 924, 953
__draw_backend_cm_decompose_-
auxi:nnnnN 953
__draw_backend_cm_decompose_-
auxii:nnnnN 953
__draw_backend_cm_decompose_-
auxiii:nnnnN 953
__draw_backend_color_fill:n . . . 686
__draw_backend_color_fill:nnn 1208
__draw_backend_color_fill_-
cmyk:nnnn 686, 861, 1208
__draw_backend_color_fill_-
gray:n 686, 861, 1208

\cs_new_protected:Npn
. 26, 30, 33, 40, 46, 51, 53,
95, 98, 100, 102, 106, 109, 111, 113,
121, 143, 145, 160, 176, 191, 193,
219, 233, 248, 250, 263, 277, 330,
343, 368, 388, 393, 402, 404, 409,
411, 420, 425, 435, 462, 473, 478,
480, 482, 492, 494, 519, 525, 530,
532, 534, 542, 550, 559, 569, 571,
574, 576, 590, 595, 616, 638, 641,
654, 667, 672, 674, 676, 678, 680,
682, 684, 686, 695, 704, 706, 708,
713, 718, 724, 730, 742, 766, 768,
772, 777, 782, 792, 801, 803, 806,
808, 810, 812, 817, 822, 827, 829,
842, 847, 849, 851, 853, 855, 857,
859, 861, 870, 879, 881, 883, 888,
914, 929, 954, 966, 978, 990, 997,
1022, 1027, 1029, 1037, 1047, 1055,
1060, 1065, 1076, 1086, 1096, 1098,
1100, 1102, 1133, 1135, 1140, 1142,
1144, 1147, 1168, 1179, 1192, 1194,
1196, 1198, 1200, 1202, 1204, 1206,
1208, 1218, 1227, 1235, 1237, 1239,
1249, 1264, 1269, 1284, 1314, 1328,
1343, 1355, 1366, 1394, 1406, 1419,
1429, 1450, 1457, 1465, 1476, 1480,
1483, 1498, 1508, 1543, 1550, 1556,
1562, 1565, 1572, 1581, 1586, 1594,
1613, 1620, 1626, 1628, 1630, 1641,
1660, 1663, 1665, 1669, 1679, 1700,
1705, 1710, 1715, 1724, 1750, 1764,
1795, 1797, 1799, 1801, 1806, 1821,
1826, 1868, 1897, 1921, 1930, 1969,
1976, 2001, 2025, 2026, 2027, 2028,
2031, 2033, 2055, 2057, 2162, 2195,
2237, 2298, 2300, 2307, 2309, 2313,
2323, 2329, 2334, 2339, 2344, 2346,
2348, 2356, 2380, 2395, 2410, 2412,
2414, 2425, 2428, 2430, 2451, 2482,
2484, 2489, 2494, 2501, 2503, 2507,
2508, 2509, 2511, 2512, 2513, 2514,
2516, 2517, 2518, 2519, 2522, 2523

\cs_new_protected:Npx 57, 65, 73, 79,
85, 496, 505, 894, 902, 2037, 2059,
2067, 2081, 2088, 2107, 2122, 2129,
2140, 2229, 2243, 2251, 2269, 2304

\cs_set_eq:NN 1962, 1963

\cs_set_protected:Npn
. 373, 430, 2138, 2160, 2193

_draw_backend_color_fill_-	_draw_backend_nonzero_rule: ...
rgb:nnn 686 , 861 , 1208 569 , 801 , 1096
_draw_backend_color_gray_aux:n	_draw_backend_path:n 1100
..... 1231 , 1235	_draw_backend_rectangle:nnnn ..
_draw_backend_color_reset: .. 861 534 , 772 , 1055
_draw_backend_color_select:n . 861	_draw_backend_scope:n
_draw_backend_color_stroke:n . 686	... 1025 , 1029 , 1097 , 1099 , 1119 ,
_draw_backend_color_stroke_-	1159 , 1181 , 1193 , 1195 , 1197 , 1199 ,
cmyk:nnnn 686 , 861 , 1208	1201 , 1203 , 1205 , 1207 , 1251 , 1271
_draw_backend_color_stroke_-	_draw_backend_scope_begin: ...
gray:n 686 , 861 , 1208 530 , 767 , 770 , 1024 , 1029
_draw_backend_color_stroke_-	_draw_backend_scope_end:
rgb:nnn 686 , 861 , 1208 530 , 769 , 770 , 1028 , 1029
_draw_backend_curveto:nnnnnn ..	_draw_backend_select:n
..... 534 , 772 , 1055 1220 , 1238 , 1266
_draw_backend_dash:n 654 , 829 , 1168	_draw_backend_stroke: 574 , 806 , 1100
_draw_backend_dash_aux:nn .. 1168	\g_draw_clip_path_int
_draw_backend_dash_pattern:nn .	.. 1106 , 1109 , 1122 , 1151 , 1154 , 1162
..... 654 , 829 , 1168	_draw_color_reset: 727
_draw_backend_discardpath: ...	\g_draw_draw_clip_bool ... 574 , 1100
..... 574 , 806 , 1100	\g_draw_draw_eor_bool
_draw_backend_end: . 519 , 766 , 1022 569 , 583 , 601 ,
_draw_backend_evenodd_rule: ...	609 , 622 , 631 , 647 , 801 , 815 , 820 , 825
..... 569 , 801 , 1096	\g_draw_draw_path_int 1100
_draw_backend_fill: 574 , 806 , 1100	\g_draw_draw_path_tl
_draw_backend_fillstroke: 1055 , 1111 , 1127 , 1129 , 1156 , 1165
..... 574 , 806 , 1100	\g_draw_draw_scope_int 1029
_draw_backend_join_bevel:	\l_draw_draw_scope_int 1029
..... 654 , 829 , 1168	\g_draw_path_int 1115 , 1132
_draw_backend_join_miter:	
..... 654 , 829 , 1168	
_draw_backend_join_round:	
..... 654 , 829 , 1168	
_draw_backend_lineto:nn	
..... 534 , 772 , 1055	
_draw_backend_linewidth:n	
..... 654 , 829 , 1168	
_draw_backend_literal:n 517 , 522 ,	
523 , 527 , 531 , 533 , 536 , 544 , 552 ,	
561 , 575 , 578 , 581 , 587 , 597 , 598 ,	
599 , 604 , 607 , 613 , 618 , 619 , 620 ,	
625 , 626 , 629 , 635 , 645 , 651 , 656 ,	
669 , 673 , 675 , 677 , 679 , 681 , 683 ,	
685 , 732 , 744 , 745 , 746 , 747 , 748 ,	
749 , 753 , 754 , 756 , 757 , 758 , 759 ,	
760 , 764 , 774 , 779 , 784 , 794 , 807 ,	
809 , 811 , 814 , 819 , 824 , 828 , 831 ,	
844 , 848 , 850 , 852 , 854 , 856 , 858 ,	
860 , 1020 , 1041 , 1049 , 1107 , 1126 , 1152	
_draw_backend_miterlimit:n ...	
..... 654 , 829 , 1168	
_draw_backend_moveto:nn	
..... 534 , 772 , 1055	

E	
\endlandscape	2374
\evensidemargin	1842
exp commands:	
\exp_after:wN	380 , 1600
\exp_args:Nf	659 , 834
\exp_args:NNf	144 , 192 , 249
\exp_args:Nnx	1741 , 2359
\exp_args:NV	375
\exp_args:Nx	
... 479 , 1412 , 1433 , 1712 , 1834 , 2325	
\exp_last_unbraced:Nx	384 , 432
\exp_not:N	37 ,
62 , 71 , 90 , 501 , 502 , 510 , 899 , 900 ,	
907 , 2042 , 2043 , 2044 , 2049 , 2051 ,	
2052 , 2075 , 2077 , 2078 , 2084 , 2085 ,	
2086 , 2091 , 2092 , 2094 , 2104 , 2112 ,	
2113 , 2116 , 2117 , 2118 , 2144 , 2146 ,	
2153 , 2223 , 2225 , 2226 , 2231 , 2235 ,	
2245 , 2257 , 2258 , 2264 , 2265 , 2271 ,	
2273 , 2275 , 2282 , 2287 , 2293 , 2295	
\exp_not:n ...	27 , 62 , 71 , 90 , 1703 ,
1708 , 1997 , 2170 , 2171 , 2190 , 2191 ,	
2201 , 2202 , 2337 , 2342 , 2353 , 2434	

F

file commands:

\file_compare_timestamp:nNnTF . 1421
 \file_parse_full_name:nNNN 1408, 1431

fp commands:

\fp_compare:nNnTF
 151, 198, 204, 256, 934, 947, 992
 \fp_eval:n 144, 153, 166,
 167, 192, 209, 224, 226, 249, 258,
 269, 270, 337, 352, 353, 398, 399,
 403, 407, 467, 468, 469, 470, 479,
 487, 488, 489, 673, 690, 691, 700,
 701, 705, 707, 711, 716, 735, 736,
 848, 865, 866, 874, 875, 880, 882,
 886, 891, 919, 920, 936, 941, 942,
 949, 959, 960, 961, 962, 971, 972,
 973, 974, 983, 984, 985, 986, 1007,
 1008, 1195, 1213, 1214, 1215, 1223,
 1224, 1232, 1238, 1244, 1245, 1246,
 1267, 1277, 1278, 1994, 2104, 2447
 \fp_new:N 217, 218
 \fp_set:Nn 197, 200
 \fp_use:N 203, 207, 212
 \fp_zero:N 199
 \c_zero_fp . 151, 198, 204, 256, 934, 947

G

galley commands:

\l_galley_text_width_dim 1906
 \l_galley_total_left_margin_dim 1837

graphics commands:

\graphics_bb_restore:nTF . 1357, 1583
 \graphics_bb_save:n 1392, 1591
 \l_graphics_decodearray_tl
 1334, 1335,
 1345, 1371, 1375, 1376, 1459, 1491,
 1492, 1530, 1533, 1534, 1552, 1622
 \graphics_extract_bb:n
 1454, 1461, 1617, 1624
 \l_graphics_interpolate_bool ...
 1336, 1346, 1370, 1377,
 1460, 1493, 1529, 1535, 1553, 1623
 \l_graphics_llx_dim
 1319, 1470, 1522, 1635
 \l_graphics_lly_dim
 1320, 1471, 1523, 1636
 \l_graphics_name_tl 1426
 \l_graphics_page_int
 1330, 1350, 1351, 1381,
 1382, 1452, 1489, 1490, 1516, 1517,
 1545, 1558, 1559, 1598, 1599, 1615
 \l_graphics_pagebox_tl
 41, 1331, 1349,

1383, 1384, 1453, 1487, 1488, 1518,
 1520, 1546, 1567, 1568, 1600, 1616
 \graphics_read_bb:n . 1313, 1448, 1612
 \l_graphics_urx_dim
 .. 1321, 1388, 1472, 1524, 1589, 1637
 \l_graphics_ury_dim .. 1322, 1389,
 1473, 1525, 1590, 1638, 1646, 1647

graphics internal commands:

\l__graphics_backend_dir_str . 1401
 \l__graphics_backend_ext_str . 1401
 __graphics_backend_getbb_auxi:n
 1328
 __graphics_backend_getbb_-
 auxi:nN 1543
 __graphics_backend_getbb_-
 auxii:n 1328
 __graphics_backend_getbb_-
 auxii:nnN 1543
 __graphics_backend_getbb_-
 auxiii:nNnn 1543
 __graphics_backend_getbb_-
 auxiv:nnNnn 1543
 __graphics_backend_getbb_-
 auxv:nNnn 1543
 __graphics_backend_getbb_-
 auxvi:nNnn 1584, 1586
 __graphics_backend_getbb_eps:n .
 1307, 1401, 1442, 1606
 __graphics_backend_getbb_eps:nm
 1401
 __graphics_backend_getbb_eps:nn
 1412, 1419
 __graphics_backend_getbb_jpg:n .
 1328, 1442, 1543, 1613
 __graphics_backend_getbb_-
 pagebox:w 1543, 1600
 __graphics_backend_getbb_pdf:n .
 1328, 1427, 1442, 1543, 1620
 __graphics_backend_getbb_png:n .
 1328, 1442, 1543, 1613
 __graphics_backend_include:nn 1626
 __graphics_backend_include_-
 auxi:nn 1465
 __graphics_backend_include_-
 auxii:nnn 1465
 __graphics_backend_include_-
 auxiii:nnn 1465
 __graphics_backend_include_-
 bitmap_quote:w 1594, 1641
 __graphics_backend_include_-
 eps:n 1314, 1401, 1465, 1626
 __graphics_backend_include_-
 jpg:n 1394, 1465, 1641

<code>__graphics_backend_include_-pdf:n</code> .. 1394 , 1433 , 1465 , 1594 , 1626	<code>\int_set_eq:NN</code> 1031 , 1935
<code>__graphics_backend_include_pdf_-quote:w</code> 1597 , 1602	<code>\int_use:N</code> 281 , 312 , 1109 , 1115 , 1122 , 1154 , 1162 , 1351 , 1382 , 1397 , 1490 , 1503 , 1515 , 1517 , 1599 , 1678 , 1729 , 1742 , 1746 , 1758 , 1781 , 1787 , 1860 , 1968 , 2158 , 2165 , 2322 , 2360 , 2365 , 2401 , 2409
<code>__graphics_backend_include_-png:n</code> 1394 , 1465 , 1641	<code>\int_value:w</code> .. 2049 , 2075 , 2223 , 2235
<code>\l__graphics_backend_name_str</code> . 1401	<code>\int_zero:N</code> ... 1330 , 1452 , 1545 , 1615
<code>\l__graphics_graphics_attr_tl</code> ... 1327 , 1332 , 1339 , 1347 , 1357 , 1390 , 1392 , 1397	
<code>\l__graphics_internal_box</code> 1386 , 1388 , 1389 , 1588 , 1589 , 1590	K
<code>\g__graphics_track_int</code> 1464 , 1510 , 1511	kernel internal commands:
group commands:	<code>__kernel_backend_align_begin:</code> .. 40 , 124 , 148 , 163
<code>\group_begin:</code> 1034 , 2003 , 2109 , 2137 , 2159 , 2192 , 2453	<code>__kernel_backend_align_end:</code> ... 40 , 138 , 156 , 170
<code>\group_end:</code> 1042 , 2023 , 2120 , 2156 , 2188 , 2219 , 2480	<code>__kernel_backend_literal:n</code> 25 , 31 , 34 , 39 , 42 , 49 , 52 , 54 , 96 , 99 , 101 , 103 , 107 , 253 , 266 , 413 , 421 , 521 , 528 , 726 , 931 , 938 , 944 , 1004 , 1014 , 1316 , 1467 , 1502 , 1512 , 1632 , 1643 , 2305 , 2429 , 2483 , 2487 , 2492 , 2497
<code>\group_insert_after:N</code> 417 , 502 , 727 , 900	<code>__kernel_backend_literal_page:n</code> 65 , 98 , 2299 , 2301 , 2502 , 2504
H	<code>__kernel_backend_literal_pdf:n</code> . 57 , 95 , 179 , 236 , 764 , 911
hbox commands:	<code>__kernel_backend_literal_-postscript:n</code> ... 30 , 43 , 44 , 48 , 125 , 126 , 128 , 129 , 137 , 149 , 164 , 517
<code>\hbox:n</code> 1767 , 1768 , 1771 , 1846 , 1852 , 2007 , 2011 , 2457 , 2466	<code>__kernel_backend_literal_svg:n</code> . 106 , 110 , 112 , 114 , 280 , 282 , 299 , 1020 , 1288 , 1299
<code>\hbox_overlap_right:n</code> 139 , 171 , 187 , 228 , 244 , 272 , 359 , 755 , 1012	<code>__kernel_backend_matrix:n</code> 85 , 201 , 222 , 917
<code>\hbox_set:Nn</code> 1386 , 1588 , 1833 , 1870 , 2004 , 2110 , 2454	<code>__kernel_backend_postscript:n</code> .. 33 , 415 , 720 , 1661 , 1717 , 1767 , 1774 , 1809 , 1846 , 1853 , 1857 , 1871 , 1899 , 1949 , 1956 , 1962 , 1971 , 1978 , 2007 , 2011
<code>\hbox_set:Nw</code> 1816	<code>__kernel_backend_scope_begin:</code> 5 , 51 , 73 , 100 , 109 , 123 , 147 , 162 , 178 , 195 , 221 , 235 , 252 , 265 , 770 , 999 , 1286
<code>\hbox_set_end:</code> 1831	<code>__kernel_backend_scope_begin:n</code> . 113 , 301 , 309 , 314 , 332 , 345
<code>\hbox_unpack:N</code> 1963	<code>__kernel_backend_scope_end:</code> ... 51 , 73 , 100 , 109 , 140 , 158 , 172 , 188 , 215 , 229 , 245 , 261 , 273 , 324 , 325 , 326 , 341 , 360 , 771 , 1016 , 1300
I	<code>\l__kernel_color_stack_int</code> 461 , 501 , 510 , 899 , 907
int commands:	
<code>\int_compare:nNnTF</code> 1350 , 1381 , 1489 , 1516 , 1558 , 1598 , 1934 , 2255 , 2281	L
<code>\int_const:Nn</code> 1390 , 1511 , 1672 , 2143 , 2316	<code>\landscape</code> 2370 , 2372
<code>\int_eval:n</code> 2235 , 2258 , 2265 , 2275 , 2483 , 2491 , 2496	
<code>\int_gincr:N</code> 279 , 1050 , 1106 , 1151 , 1510 , 1671 , 1726 , 1753 , 1777 , 1855 , 2315 , 2358 , 2397	
<code>\int_gset:Nn</code> 1923	
<code>\int_gset_eq:NN</code> 1043 , 1754 , 1778 , 1856 , 2398	
<code>\int_gzero:N</code> 1035	
<code>\int_if_exist:NTF</code> 1500	
<code>\int_if_odd:nTF</code> 1840	
<code>\int_new:N</code> 329 , 461 , 1053 , 1054 , 1132 , 1464 , 1667 , 1749 , 1788 , 1790 , 2311 , 2379	

M

math commands:

`\c_math_toggle_token` 1819, 1829

mode commands:

`\mode_if_horizontal:TF` . . . 1925, 1932

`\mode_if_math:TF` 1813

O

`\oddsidemargin` 1841

P

pdf internal commands:

`__pdf_backend:n` 2304,
2308, 2310, 2336, 2341, 2350, 2399,
2416, 2426, 2432, 2459, 2460, 2468

`__pdf_backend_annotation:nnnn` . .
. 1750, 2037, 2380

`__pdf_backend_annotation_-
aux:nnnn` 1750, 2380

`\g__pdf_backend_annotation_int` . .
. 1749,
1754, 1778, 1787, 2379, 2398, 2409

`__pdf_backend_annotation_last:` .
. 1786, 2047, 2408

`__pdf_backend_bdc:nn`
. 2031, 2298, 2501, 2522

`__pdf_backend_catalog_gput:nn` . .
. 1663, 2122, 2307, 2507

`__pdf_backend_compress_objects:n`
. 2025, 2229, 2482, 2516

`__pdf_backend_compresslevel:n` . .
. 2025, 2229, 2482, 2516

`\l__pdf_backend_content_box` 1747,
1816, 1845, 1848, 1850, 1879, 1890

`__pdf_backend_destination:nn` . .
. 1976, 2088, 2430

`__pdf_backend_destination_-
rectangle:nn` 1976, 2088, 2430

`__pdf_backend_emc:`
. 2031, 2298, 2501, 2522

`__pdf_backend_info_gput:nn`
. 1663, 2122, 2307, 2507

`__pdf_backend_link:nw` 1797

`__pdf_backend_link_aux:nw` . . . 1797

`__pdf_backend_link_begin:n` . . 2410

`__pdf_backend_link_begin:nnnw` 2055

`__pdf_backend_link_begin:nw` . .
. 1798, 1800, 1801

`__pdf_backend_link_begin_aux:nw`
. 1804, 1806

`__pdf_backend_link_begin_-
goto:nnw` 1797, 2055, 2410

`__pdf_backend_link_begin_-
user:nnw` 1797, 2055, 2410

`\g__pdf_backend_link_bool`
. 1792, 1803, 1808, 1823, 1866

`\g__pdf_backend_link_dict_tl` . . .
. 1789, 1811, 1861

`__pdf_backend_link_end:`
. 1797, 2055, 2410

`__pdf_backend_link_end_aux:` . 1797

`\g__pdf_backend_link_int`
. 1788, 1856, 1860, 1968

`__pdf_backend_link_last:`
. 1967, 2073, 2427

`__pdf_backend_link_margin:n` . . .
. 1969, 2081, 2428

`\g__pdf_backend_link_math_bool` . .
. 1791, 1814, 1815, 1818, 1828

`__pdf_backend_link_minima:` . . 1797

`__pdf_backend_link_outerbox:n` 1797

`\g__pdf_backend_link_sf_int`
. 1790, 1923, 1934, 1935

`__pdf_backend_link_sf_restore:` 1797

`__pdf_backend_link_sf_save:` . 1797

`\l__pdf_backend_model_box` . 1748,
1833, 1870, 1878, 1889, 1904, 1911

`__pdf_backend_objcompresslevel:n`
. 2229

`\g__pdf_backend_object_int`
1667, 1671, 1674, 1726, 1729, 1742,
1746, 1753, 1754, 1758, 1777, 1778,
1781, 1855, 1856, 2311, 2315, 2318,
2358, 2360, 2365, 2397, 2398, 2401

`__pdf_backend_object_last:`
. 1745, 2221, 2364, 2509

`__pdf_backend_object_new:nn` . . .
. 1669, 2137, 2313, 2509

`__pdf_backend_object_now:nn` . . .
. 1724, 2192, 2356, 2509

`\g__pdf_backend_object_prop`
. 1667, 1675, 1686, 1696,
2136, 2146, 2168, 2311, 2319, 2326

`__pdf_backend_object_ref:n` 1669,
1683, 1697, 2137, 2313, 2332, 2509

`__pdf_backend_object_write:nn` . .
. 1679, 2159, 2323, 2509

`__pdf_backend_object_write:nnn` 2323

`__pdf_backend_object_write_-
array:nn` 1679, 2323

`__pdf_backend_object_write_-
dict:nn` 1679, 2323

`__pdf_backend_object_write_-
fstream:nn` 2323

`__pdf_backend_object_write_-
stream:nn` 1679, 2323

`__pdf_backend_object_write_-
stream:nnn` 1679

T

TeX and L^AT_EX 2_ε commands:

\ccclv 1945, 1947, 1955
 \@makecol@hook 1938
 \current@color . 13, 375, 380, 385, 433
 \special 1

tex commands:

\tex_baselineskip:D 1915
 \tex_global:D
 2231, 2245, 2257, 2264, 2271
 \tex_immediate:D 1368, 2164, 2197
 \tex_kern:D 1773
 \tex_luatexversion:D 2255, 2281
 \tex_pdfannot:D 2041
 \tex_pdfcatalog:D 2126
 \tex_pdfcolorstack:D 500, 509, 898, 906
 \tex_pdfcompresslevel:D .. 2232, 2233
 \tex_pdfdest:D 2092, 2113
 \tex_pdfendlink:D 2071
 \tex_pdfextension:D 59, 60, 67, 68,
 75, 76, 81, 82, 87, 88, 498, 499, 507,
 508, 896, 897, 904, 905, 2039, 2040,
 2061, 2062, 2069, 2070, 2090, 2091,
 2111, 2112, 2124, 2125, 2131, 2132,
 2149, 2152, 2185, 2186, 2216, 2217
 \tex_pdffeedback:D 2050,
 2051, 2076, 2077, 2153, 2224, 2225
 \tex_pdfinfo:D 2133
 \tex_pdflastannot:D 2052
 \tex_pdflastlink:D 2078
 \tex_pdflastobj:D 2155, 2226
 \tex_pdflastximage:D 1387, 1391
 \tex_pdflinkmargin:D 2085
 \tex_pdfliteral:D 61, 69
 \tex_pdfmajorversion:D
 2262, 2264, 2286, 2287
 \tex_pdfminorversion:D
 2272, 2273, 2294, 2295
 \tex_pdfobj:D 2155, 2187, 2218
 \tex_pdfobjcompresslevel:D 2246, 2247
 \tex_pdfrefximage:D 1387, 1396
 \tex_pdfrestore:D 83
 \tex_pdfsave:D 77
 \tex_pdfsetmatrix:D 89
 \tex_pdfstartlink:D 2063
 \tex_pdfvariable:D
 2083, 2084, 2234, 2248,
 2253, 2257, 2274, 2279, 2282, 2296

\tex_pdfximage:D 1368
 \tex_pdfximagebbox:D 1362
 \tex_spacefactor:D 1926, 1935
 \tex_special:D 25
 \tex_the:D 1391, 2282, 2287, 2293
 \tex_XeTeXpdffile:D 1554, 1596
 \tex_XeTeXpicfile:D 1547
 \textwidth 1909

tl commands:

\c_space_tl
 .. 203, 208, 211, 380, 1091, 1318,
 1319, 1320, 1321, 1469, 1470, 1471,
 1472, 1517, 1520, 1522, 1523, 1524,
 1525, 1597, 1599, 1634, 1635, 1636,
 1637, 1861, 2053, 2079, 2227, 2401
 \tl_clear:N 1331, 1339, 1345,
 1453, 1459, 1546, 1552, 1616, 1622
 \tl_gclear:N 1129, 1165
 \tl_gset:Nn 1088, 1811
 \tl_if_empty:NTF . 1091, 1334, 1375,
 1383, 1487, 1491, 1518, 1533, 1567
 \tl_if_empty:nTF 1185
 \tl_if_empty_p:N 1371, 1530
 \tl_if_head_is_space:nTF 375
 \tl_new:N 1095, 1327, 1789, 1793
 \tl_put_left:Nn 2374
 \tl_put_right:Nn 1943, 2372
 \tl_set:Nn . 377, 389, 439, 442, 445,
 449, 452, 1332, 1347, 1426, 1794, 1961
 \tl_to_str:n 1673,
 1678, 2144, 2158, 2166, 2317, 2322

U

use commands:

\use:N 1695, 1741, 2331, 2359
 \use:n 38, 380, 464, 484,
 659, 834, 956, 968, 980, 1170, 1210,
 1229, 1241, 1308, 1443, 1574, 1607
 \use_none:n 449, 1185, 1187, 1939

V

\value 1840
 vbox commands:
 \vbox:n 2386
 \vbox_set:Nn 1947
 \vbox_unpack_drop:N 1955