

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}{2020-02-23}{ }
6 {L3 backend support: dvipdfmx}
7 </dvipdfmx>
8 <*dvips>
9 {l3backend-dvips.def}{2020-02-23}{ }
10 {L3 backend support: dvips}
11 </dvips>
12 <*dvisvgm>
13 {l3backend-dvisvgm.def}{2020-02-23}{ }
14 {L3 backend support: dvisvgm}
15 </dvisvgm>
16 <*pdfmode>
17 {l3backend-pdfmode.def}{2020-02-23}{ }
18 {L3 backend support: PDF mode}
19 </pdfmode>
20 <*xdvipdfmx>
21 {l3backend-xdvipdfmx.def}{2020-02-23}{ }
22 {L3 backend support: xdvipdfmx}
23 </xdvipdfmx>
24 </package>
```

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

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

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

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

(End definition for `__kernel_backend_literal:e`.)

1.1 dvips backend

29 `<*dvips>`

`_kernel_backend_literal_postscript:n` Literal PostScript can be included using a few low-level formats. Here, we use the form with no positioning; this is overall more convenient as a wrapper. Note that this does require that where position is important, an appropriate wrapper is included.

```
30 \cs_new_protected:Npn \__kernel_backend_literal_postscript:n #1
31 { \__kernel_backend_literal:n { ps:: #1 } }
32 \cs_generate_variant:Nn \__kernel_backend_literal_postscript:n { x }
```

(End definition for `__kernel_backend_literal_postscript:n`.)

`_kernel_backend_postscript:n` PostScript data that does have positioning, and also applying a shift to `SDict` (which is not done automatically by `ps:` or `ps::`, in contrast to `!` or `"`).

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

(End definition for `__kernel_backend_postscript:n`.)

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 { \AtBeginDvi }
38 { \use:n }
39 {
40   \bool_lazy_and:nnT
41     { \cs_if_exist:Np \g__kernel_backend_header_bool }
42     { \g__kernel_backend_header_bool }
43     { \__kernel_backend_literal:n { header = l3backend-dvips.pro } }
44 }
```

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

```
45 \cs_new_protected:Npn \__kernel_backend_align_begin:
46 {
47   \__kernel_backend_literal:n { ps::[begin] }
48   \__kernel_backend_literal_postscript:n { currentpoint }
49   \__kernel_backend_literal_postscript:n { currentpoint~translate }
50 }
51 \cs_new_protected:Npn \__kernel_backend_align_end:
52 {
53   \__kernel_backend_literal_postscript:n { neg~exch~neg~exch~translate }
54   \__kernel_backend_literal:n { ps::[end] }
55 }
```

(End definition for `__kernel_backend_align_begin:` and `__kernel_backend_align_end:.`)

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

```

56 \cs_new_protected:Npn \_kernel_backend_scope_begin:
57   { \_kernel_backend_literal:n { ps:gsave } }
58 \cs_new_protected:Npn \_kernel_backend_scope_end:
59   { \_kernel_backend_literal:n { ps:grestore } }

```

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

```

60 \</dvips>

```

1.2 pdfmode backend

```

61 \*pdfmode>

```

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

`_kernel_backend_literal_pdf:n` This is equivalent to `\special{pdf:}` but the engine can track it. Without the `direct`
`_kernel_backend_literal_pdf:x` keyword everything is kept in sync: the transformation matrix is set to the current point automatically. Note that this is still inside the text (BT ...ET block).

```

62 \cs_new_protected:Npx \_kernel_backend_literal_pdf:n #1
63   {
64     \cs_if_exist:NTF \tex_pdfextension:D
65       { \tex_pdfextension:D literal }
66       { \tex_pdfliteral:D }
67       { \exp_not:N \exp_not:n {#1} }
68   }
69 \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.

```

70 \cs_new_protected:Npx \_kernel_backend_literal_page:n #1
71   {
72     \cs_if_exist:NTF \tex_pdfextension:D
73       { \tex_pdfextension:D literal ~ }
74       { \tex_pdfliteral:D }
75     page
76     { \exp_not:N \exp_not:n {#1} }
77   }

```

(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:`

```

78 \cs_new_protected:Npx \_kernel_backend_scope_begin:
79   {
80     \cs_if_exist:NTF \tex_pdfextension:D
81       { \tex_pdfextension:D save \scan_stop: }
82       { \tex_pdfsave:D }
83   }
84 \cs_new_protected:Npx \_kernel_backend_scope_end:
85   {

```

```

86 \cs_if_exist:NTF \tex_pdfextension:D
87 { \tex_pdfextension:D restore \scan_stop: }
88 { \tex_pdfrestore:D }
89 }

```

(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.

```

90 \cs_new_protected:Npx \__kernel_backend_matrix:n #1
91 {
92   \cs_if_exist:NTF \tex_pdfextension:D
93   { \tex_pdfextension:D setmatrix }
94   { \tex_pdfsetmatrix:D }
95   { \exp_not:N \exp_not:n {#1} }
96 }
97 \cs_generate_variant:Nn \__kernel_backend_matrix:n { x }

```

(End definition for `__kernel_backend_matrix:n.`)

```

98 </pdfmode>

```

1.3 dvipdfmx backend

```

99 <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.

```

100 \cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1
101 { \__kernel_backend_literal:n { pdf:literal~ #1 } }
102 \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!

```

103 \cs_new_protected:Npn \__kernel_backend_literal_page:n #1
104 { \__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:
105 \cs_new_protected:Npn \__kernel_backend_scope_begin:
106 { \__kernel_backend_literal:n { x:gsave } }
107 \cs_new_protected:Npn \__kernel_backend_scope_end:
108 { \__kernel_backend_literal:n { x:grestore } }

```

(End definition for `__kernel_backend_scope_begin:` and `__kernel_backend_scope_end:.`)

```

109 </dvipdfmx | xdvipdfmx>

```

1.4 dvisvgm backend

110 $\langle *dvisvgm \rangle$

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

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.

111 $\backslash cs_new_protected:Npn _kernel_backend_literal_svg:n \#1$
 112 $\{ _kernel_backend_literal:n \{ dvisvgm:raw~ \#1 \{ ?nl \} \} \}$
 113 $\backslash cs_generate_variant:Nn _kernel_backend_literal_svg:n \{ x \}$

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

$\backslash_kernel_backend_scope_begin:$
 $\backslash_kernel_backend_scope_end:$

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

114 $\backslash cs_new_protected:Npn _kernel_backend_scope_begin:$
 115 $\{ _kernel_backend_literal_svg:n \{ <g> \} \}$
 116 $\backslash cs_new_protected:Npn _kernel_backend_scope_end:$
 117 $\{ _kernel_backend_literal_svg:n \{ </g> \} \}$

(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).

118 $\backslash cs_new_protected:Npn _kernel_backend_scope_begin:n \#1$
 119 $\{ _kernel_backend_literal_svg:n \{ <g~ \#1 > \} \}$
 120 $\backslash cs_generate_variant:Nn _kernel_backend_scope_begin:n \{ x \}$

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

121 $\langle /dvisvgm \rangle$
 122 $\langle /initex | package \rangle$

2 l3backend-box Implementation

123 $\langle *initex | package \rangle$
 124 $\langle @@=box \rangle$

2.1 dvips backend

125 $\langle *dvips \rangle$

$\backslash_box_backend_clip:N$

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.

126 $\backslash cs_new_protected:Npn _box_backend_clip:N \#1$
 127 $\{$
 128 $\backslash_kernel_backend_scope_begin:$
 129 $\backslash_kernel_backend_align_begin:$
 130 $\backslash_kernel_backend_literal_postscript:n \{ matrix~currentmatrix \}$
 131 $\backslash_kernel_backend_literal_postscript:n$

```

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

```

(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.

```

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

```

(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.

```

165 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
166 {
167   \__kernel_backend_scope_begin:
168   \__kernel_backend_align_begin:
169   \__kernel_backend_literal_postscript:x
170   {
171     \fp_eval:n { round ( #2 , 5 ) } ~
172     \fp_eval:n { round ( #3 , 5 ) } ~
173     scale
174   }
175   \__kernel_backend_align_end:

```

```

176     \hbox_overlap_right:n { \box_use:N #1 }
177     \__kernel_backend_scope_end:
178 }

```

(End definition for __box_backend_scale:Nnn.)

```

179 </dvips>

```

2.2 pdfmode backend

```

180 <*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.

```

181 \cs_new_protected:Npn \__box_backend_clip:N #1
182 {
183   \__kernel_backend_scope_begin:
184   \__kernel_backend_literal_pdf:x
185   {
186     0~
187     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
188     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
189     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
190     re~W~n
191   }
192   \hbox_overlap_right:n { \box_use:N #1 }
193   \__kernel_backend_scope_end:
194   \skip_horizontal:n { \box_wd:N #1 }
195 }

```

(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.

```

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

```

```

209     \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp
210     { 0~0 }
211     {
212         \fp_use:N \l__box_backend_sin_fp
213         \c_space_tl
214         \fp_eval:n { -\l__box_backend_sin_fp }
215     }
216     \c_space_tl
217     \fp_use:N \l__box_backend_cos_fp
218 }
219 \box_use:N #1
220 \__kernel_backend_scope_end:
221 }
222 \fp_new:N \l__box_backend_cos_fp
223 \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.

```

224 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
225 {
226     \__kernel_backend_scope_begin:
227     \__kernel_backend_matrix:x
228     {
229         \fp_eval:n { round ( #2 , 5 ) } ~
230         0~0~
231         \fp_eval:n { round ( #3 , 5 ) }
232     }
233     \hbox_overlap_right:n { \box_use:N #1 }
234     \__kernel_backend_scope_end:
235 }

```

(End definition for __box_backend_scale:Nnn.)

236 </pdfmode>

2.3 dvipdfmx backend

237 <*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.

```

238 \cs_new_protected:Npn \__box_backend_clip:N #1
239 {
240     \__kernel_backend_scope_begin:
241     \__kernel_backend_literal_pdf:x
242     {
243         0~
244         \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
245         \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
246         \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
247         re~W~n
248     }
249     \hbox_overlap_right:n { \box_use:N #1 }
250     \__kernel_backend_scope_end:

```



```

251     \skip_horizontal:n { \box_wd:N #1 }
252 }

```

(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.

```

253 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
254 { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
255 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
256 {
257   \__kernel_backend_scope_begin:
258   \__kernel_backend_literal:x
259   {
260     x:rotate~
261     \fp_compare:nNnTF {#2} = \c_zero_fp
262     { 0 }
263     { \fp_eval:n { round ( #2 , 5 ) } }
264   }
265   \box_use:N #1
266   \__kernel_backend_scope_end:
267 }

```

(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.

```

268 \cs_new_protected:Npn \_box_backend_scale:Nnn #1#2#3
269 {
270   \__kernel_backend_scope_begin:
271   \__kernel_backend_literal:x
272   {
273     x:scale~
274     \fp_eval:n { round ( #2 , 5 ) } ~
275     \fp_eval:n { round ( #3 , 5 ) }
276   }
277   \hbox_overlap_right:n { \box_use:N #1 }
278   \__kernel_backend_scope_end:
279 }

```

(End definition for `_box_backend_scale:Nnn`.)

```

280 </dvipdfmx | xdvipdfmx>

```

2.4 dvisvgm backend

```

281 <*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!

```

282 \cs_new_protected:Npn \__box_backend_clip:N #1
283 {
284   \int_gincr:N \g__box_clip_path_int
285   \__kernel_backend_literal_svg:x
286   { < clipPath~id = " l3cp \int_use:N \g__box_clip_path_int " > }
287   \__kernel_backend_literal_svg:x
288   {
289     <
290     path ~ d =
291     "
292       M ~ 0 ~
293       \dim_to_decimal:n { -\box_dp:N #1 } ~
294       L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
295       \dim_to_decimal:n { -\box_dp:N #1 } ~
296       L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
297       \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
298       L ~ 0 ~
299       \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
300       Z
301     "
302     />
303   }
304   \__kernel_backend_literal_svg:n
305   { < /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.

```

306   \__kernel_backend_scope_begin:n
307   {
308     transform =
309     "
310       translate ( { ?x } , { ?y } ) ~
311       scale ( 1 , -1 )
312     "
313   }
314   \__kernel_backend_scope_begin:x
315   {
316     clip-path =
317     "url ( \c_hash_str l3cp \int_use:N \g__box_clip_path_int ) "
318   }
319   \__kernel_backend_scope_begin:n
320   {
321     transform =
322     "
323       scale ( -1 , 1 ) ~
324       translate ( { ?x } , { ?y } ) ~
325       scale ( -1 , -1 )

```

```

326         "
327     }
328     \box_use:N #1
329     \__kernel_backend_scope_end:
330     \__kernel_backend_scope_end:
331     \__kernel_backend_scope_end:
332 %     \skip_horizontal:n { \box_wd:N #1 }
333 }
334 \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.

```

335 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
336 {
337     \__kernel_backend_scope_begin:x
338     {
339         transform =
340         "
341             rotate
342             ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
343         "
344     }
345     \box_use:N #1
346     \__kernel_backend_scope_end:
347 }

```

(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).

```

348 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
349 {
350     \__kernel_backend_scope_begin:x
351     {
352         transform =
353         "
354             translate ( { ?x } , { ?y } ) ~
355             scale
356             (
357                 \fp_eval:n { round ( -#2 , 5 ) } ,
358                 \fp_eval:n { round ( -#3 , 5 ) }
359             ) ~
360             translate ( { ?x } , { ?y } ) ~
361             scale ( -1 )
362         "
363     }
364     \hbox_overlap_right:n { \box_use:N #1 }
365     \__kernel_backend_scope_end:
366 }

```

(End definition for `_box_backend_scale:Nnn`.)

```
367 </dvisvgm>
368 </initex | package>
```

3 I3backend-color Implementation

```
369 <*initex | package>
370 <@@=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

```
371 <*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).

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

```
398 \cs_new_protected:Npn \_color_backend_cmyk:nnnn #1#2#3#4
399 {
```

```

400   \_color_backend_select:x
401   {
402     cmyk~
403     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
404     \fp_eval:n {#3} ~ \fp_eval:n {#4}
405   }
406 }
407 \cs_new_protected:Npn \_color_backend_gray:n #1
408 { \_color_backend_select:x { gray~ \fp_eval:n {#1} } }
409 \cs_new_protected:Npn \_color_backend_rgb:nnn #1#2#3
410 {
411   \_color_backend_select:x
412   { rgb~ \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} }
413 }
414 \cs_new_protected:Npn \_color_backend_spot:nn #1#2
415 { \_color_backend_select:n { #1 } }
416 \cs_new_protected:Npn \_color_backend_select:n #1
417 {
418   \_kernel_backend_literal:n { color-push~ #1 }
419   <*dvips>
420   \_kernel_backend_postscript:n { /color.fc~{ }~def }
421   </dvips>
422   \group_insert_after:N \_color_backend_reset:
423 }
424 \cs_generate_variant:Nn \_color_backend_select:n { x }
425 \cs_new_protected:Npn \_color_backend_reset:
426 { \_kernel_backend_literal:n { color~pop } }

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

427 </divisgm | dvipdfmx | dvips | xdvipdfmx>

```

3.2 pdfmode

```

428 <*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

```

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

```

```

443 \str_if_eq:nnTF {#2} { g }
444 { \tl_set:Nn #7 { gray ~ #1 } }
445 {
446 \str_if_eq:nnTF {#4} { rg }
447 { \tl_set:Nn #7 { rgb ~ #1 ~ #2 ~ #3 } }
448 {
449 \str_if_eq:nnTF {#5} { k }
450 { \tl_set:Nn #7 { cmyk ~ #1 ~ #2 ~ #3 ~ #4 } }
451 {
452 \str_if_eq:nnTF {#2} { cs }
453 {
454 \tl_set:Nx #7 { spot ~ \use_none:n #1 ~ #5 }
455 }
456 {
457 \tl_set:Nn #7 { gray ~ 0 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 </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.

```

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

```

```

488 {
489   \use:x
490   {
491     \_color_backend_rgb_aux:nnn
492     { \fp_eval:n {#1} }
493     { \fp_eval:n {#2} }
494     { \fp_eval:n {#3} }
495   }
496 }
497 \cs_new_protected:Npn \_color_backend_rgb_aux:nnn #1#2#3
498 { \_color_backend_select:n { #1 ~ #2 ~ #3 ~ rg ~ #1 ~ #2 ~ #3 ~ RG } }
499 \cs_new_protected:Npn \_color_backend_spot:nn #1#2
500 { \_color_backend_select:n { /#1 ~ cs ~ /#1 ~ CS ~ #2 ~ sc ~ #2 ~ SC } }
501 \cs_new_protected:Npx \_color_backend_select:n #1
502 {
503   \cs_if_exist:NTF \tex_pdfextension:D
504   { \tex_pdfextension:D colorstack }
505   { \tex_pdfcolorstack:D }
506   \exp_not:N \l__kernel_color_stack_int push {#1}
507   \group_insert_after:N \exp_not:N \_color_backend_reset:
508 }
509 \cs_generate_variant:Nn \_color_backend_select:n { x }
510 \cs_new_protected:Npx \_color_backend_reset:
511 {
512   \cs_if_exist:NTF \tex_pdfextension:D
513   { \tex_pdfextension:D colorstack }
514   { \tex_pdfcolorstack:D }
515   \exp_not:N \l__kernel_color_stack_int pop \scan_stop:
516 }

```

(End definition for _color_backend_cmyk:nnnn and others.)

```

517 </pdfmode>
518 </initex | package>

```

4 I3backend-draw Implementation

```

519 <*initex | package>
520 <@@=draw>

```

4.1 dvips backend

```

521 <*dvips>

```

```

\_draw_backend_literal:n The same as literal PostScript: same arguments about positioning apply her.
\_draw_backend_literal:x
522 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
523 \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.

```

524 \cs_new_protected:Npn \__draw_backend_begin:
525 {
526   \__kernel_backend_literal:n { ps::[begin] }
527   \__draw_backend_literal:n { @beginspecial }
528   \__draw_backend_literal:n { SDict ~ begin ~ /color.fc ~ { } ~ def ~ end }
529 }
530 \cs_new_protected:Npn \__draw_backend_end:
531 {
532   \__draw_backend_literal:n { @endspecial }
533   \__kernel_backend_literal:n { ps::[end] }
534 }

```

(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.

```

535 \cs_new_protected:Npn \__draw_backend_scope_begin:
536 { \__draw_backend_literal:n { save } }
537 \cs_new_protected:Npn \__draw_backend_scope_end:
538 { \__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.

```

539 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
540 {
541   \__draw_backend_literal:x
542   {
543     \dim_to_decimal_in_bp:n {#1} ~
544     \dim_to_decimal_in_bp:n {#2} ~ moveto
545   }
546 }
547 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
548 {
549   \__draw_backend_literal:x
550   {
551     \dim_to_decimal_in_bp:n {#1} ~
552     \dim_to_decimal_in_bp:n {#2} ~ lineto
553   }
554 }
555 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
556 {
557   \__draw_backend_literal:x
558   {
559     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~

```



```

560         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
561         moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
562     }
563 }
564 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
565 {
566     \__draw_backend_literal:x
567     {
568         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
569         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
570         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
571         curveto
572     }
573 }

```

(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
574 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
575 { \bool_gset_true:N \g__draw_draw_eor_bool }
576 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
577 { \bool_gset_false:N \g__draw_draw_eor_bool }
578 \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
579 \cs_new_protected:Npn \__draw_backend_closepath:
580 { \__draw_backend_literal:n { closepath } }
581 \cs_new_protected:Npn \__draw_backend_stroke:
582 {
583     \__draw_backend_literal:n { stroke }
584     \bool_if:NT \g__draw_draw_clip_bool
585     {
586         \__draw_backend_literal:x
587         {
588             \bool_if:NT \g__draw_draw_eor_bool { eo }
589             clip
590         }
591     }
592     \__draw_backend_literal:n { newpath }
593     \bool_gset_false:N \g__draw_draw_clip_bool
594 }
595 \cs_new_protected:Npn \__draw_backend_closestroke:
596 {
597     \__draw_backend_closepath:
598     \__draw_backend_stroke:
599 }

```

```

600 \cs_new_protected:Npn \__draw_backend_fill:
601 {
602   \__draw_backend_literal:n { gsave }
603   \__draw_backend_literal:n { color.fc }
604   \__draw_backend_literal:x
605   {
606     \bool_if:NT \g__draw_draw_eor_bool { eo }
607     fill
608   }
609   \__draw_backend_literal:n { grestore }
610   \bool_if:NT \g__draw_draw_clip_bool
611   {
612     \__draw_backend_literal:x
613     {
614       \bool_if:NT \g__draw_draw_eor_bool { eo }
615       clip
616     }
617   }
618   \__draw_backend_literal:n { newpath }
619   \bool_gset_false:N \g__draw_draw_clip_bool
620 }
621 \cs_new_protected:Npn \__draw_backend_fillstroke:
622 {
623   \__draw_backend_literal:n { gsave }
624   \__draw_backend_literal:n { color.fc }
625   \__draw_backend_literal:x
626   {
627     \bool_if:NT \g__draw_draw_eor_bool { eo }
628     fill
629   }
630   \__draw_backend_literal:n { grestore }
631   \__draw_backend_literal:n { stroke }
632   \bool_if:NT \g__draw_draw_clip_bool
633   {
634     \__draw_backend_literal:x
635     {
636       \bool_if:NT \g__draw_draw_eor_bool { eo }
637       clip
638     }
639   }
640   \__draw_backend_literal:n { newpath }
641   \bool_gset_false:N \g__draw_draw_clip_bool
642 }
643 \cs_new_protected:Npn \__draw_backend_clip:
644 { \bool_gset_true:N \g__draw_draw_clip_bool }
645 \bool_new:N \g__draw_draw_clip_bool
646 \cs_new_protected:Npn \__draw_backend_discardpath:
647 {
648   \bool_if:NT \g__draw_draw_clip_bool
649   {
650     \__draw_backend_literal:x
651     {
652       \bool_if:NT \g__draw_draw_eor_bool { eo }
653       clip

```

```

654     }
655   }
656   \__draw_backend_literal:n { newpath }
657   \bool_gset_false:N \g__draw_draw_clip_bool
658 }

```

(End definition for __draw_backend_closepath: and others.)

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

```

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

```

691 \cs_new_protected:Npn \__draw_backend_color_fill_cmyk:nnnn #1#2#3#4
692 {
693   \__draw_backend_color_fill:x
694   {
695     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
696     \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
697     setcmykcolor

```

```

698     }
699   }
700   \cs_new_protected:Npn \__draw_backend_color_stroke_cmyk:nnnn #1#2#3#4
701   {
702     \__draw_backend_color_stroke:x
703     {
704       cmyk ~
705       \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
706       \fp_eval:n {#3} ~ \fp_eval:n {#4}
707     }
708   }
709   \cs_new_protected:Npn \__draw_backend_color_fill_gray:n #1
710   { \__draw_backend_color_fill:x { \fp_eval:n {#1} ~ setgray } }
711   \cs_new_protected:Npn \__draw_backend_color_stroke_gray:n #1
712   { \__draw_backend_color_stroke:x { gray ~ \fp_eval:n {#1} } }
713   \cs_new_protected:Npn \__draw_backend_color_fill_rgb:nnn #1#2#3
714   {
715     \__draw_backend_color_fill:x
716     { \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} ~ setrgbcolor }
717   }
718   \cs_new_protected:Npn \__draw_backend_color_stroke_rgb:nnn #1#2#3
719   {
720     \__draw_backend_color_stroke:x
721     { rgb ~ \fp_eval:n {#1} ~ \fp_eval:n {#2} ~ \fp_eval:n {#3} }
722   }
723   \cs_new_protected:Npn \__draw_backend_color_fill:n #1
724   {
725     \__kernel_backend_postscript:n
726     { /color.fc ~ { #1 } ~ def }
727   }
728   \cs_generate_variant:Nn \__draw_backend_color_fill:n { x }
729   \cs_new_protected:Npn \__draw_backend_color_stroke:n #1
730   {
731     \__kernel_backend_literal:n { color~push~#1 }
732     \group_insert_after:N \__draw_color_reset:
733   }
734   \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.

```

735   \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
736   {
737     \__draw_backend_literal:n
738     {
739       [
740         \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
741         \fp_eval:n {#3} ~ \fp_eval:n {#4} ~
742         0 ~ 0
743       ] ~
744     concat

```

```

745     }
746 }

```

(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).

```

747 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
748 {
749   \_draw_backend_literal:n { @endspecial }
750   \_draw_backend_literal:n { [end] }
751   \_draw_backend_literal:n { [begin] }
752   \_draw_backend_literal:n { save }
753   \_draw_backend_literal:n { currentpoint }
754   \_draw_backend_literal:n { currentpoint~translate }
755   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
756   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
757   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
758   \_draw_backend_literal:n { neg~exch~neg~exch~translate }
759   \_draw_backend_literal:n { [end] }
760   \hbox_overlap_right:n { \box_use:N #1 }
761   \_draw_backend_literal:n { [begin] }
762   \_draw_backend_literal:n { restore }
763   \_draw_backend_literal:n { [end] }
764   \_draw_backend_literal:n { [begin] }
765   \_draw_backend_literal:n { @beginspecial }
766 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```

767 </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.

```

768 <*dvipdfmx | pdfmode | xdvipdfmx>

```

4.2.1 Drawing

`_draw_backend_literal:n`
`_draw_backend_literal:x`

Pass data through using a dedicated interface.

```

769 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_pdf:n
770 \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:
771 \cs_new_protected:Npn \__draw_backend_begin:
772 { \__draw_backend_scope_begin: }
773 \cs_new_protected:Npn \__draw_backend_end:
774 { \__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:
775 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
776 \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
777 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
778 {
779   \__draw_backend_literal:x
780   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
781 }
782 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
783 {
784   \__draw_backend_literal:x
785   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
786 }
787 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
788 {
789   \__draw_backend_literal:x
790   {
791     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
792     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
793     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
794     c
795   }
796 }
797 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
798 {
799   \__draw_backend_literal:x
800   {
801     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
802     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
803     re
804   }
805 }

```

(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
806 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
807 { \bool_gset_true:N \g__draw_draw_eor_bool }
808 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
809 { \bool_gset_false:N \g__draw_draw_eor_bool }
810 \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>	811 <code>\cs_new_protected:Npn __draw_backend_closepath:</code>
<code>__draw_backend_closestroke:</code>	812 <code>{ __draw_backend_literal:n { h } }</code>
<code>__draw_backend_fill:</code>	813 <code>\cs_new_protected:Npn __draw_backend_stroke:</code>
<code>__draw_backend_fillstroke:</code>	814 <code>{ __draw_backend_literal:n { S } }</code>
<code>__draw_backend_clip:</code>	815 <code>\cs_new_protected:Npn __draw_backend_closestroke:</code>
<code>__draw_backend_discardpath:</code>	816 <code>{ __draw_backend_literal:n { s } }</code>
	817 <code>\cs_new_protected:Npn __draw_backend_fill:</code>
	818 <code>{</code>
	819 <code>__draw_backend_literal:x</code>
	820 <code>{ f \bool_if:NT \g__draw_draw_eor_bool * }</code>
	821 <code>}</code>
	822 <code>\cs_new_protected:Npn __draw_backend_fillstroke:</code>
	823 <code>{</code>
	824 <code>__draw_backend_literal:x</code>
	825 <code>{ B \bool_if:NT \g__draw_draw_eor_bool * }</code>
	826 <code>}</code>
	827 <code>\cs_new_protected:Npn __draw_backend_clip:</code>
	828 <code>{</code>
	829 <code>__draw_backend_literal:x</code>
	830 <code>{ W \bool_if:NT \g__draw_draw_eor_bool * }</code>
	831 <code>}</code>
	832 <code>\cs_new_protected:Npn __draw_backend_discardpath:</code>
	833 <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>	834 <code>\cs_new_protected:Npn __draw_backend_dash_pattern:nn #1#2</code>
<code>__draw_backend_linewidth:n</code>	835 <code>{</code>
<code>__draw_backend_miterlimit:n</code>	836 <code>__draw_backend_literal:x</code>
<code>__draw_backend_cap_butt:</code>	837 <code>{</code>
<code>__draw_backend_cap_round:</code>	838 <code>[</code>
<code>__draw_backend_cap_rectangle:</code>	839 <code>\exp_args:Nf \use:n</code>
<code>__draw_backend_join_miter:</code>	840 <code>{ \clist_map_function:nN {#1} __draw_backend_dash:n }</code>
<code>__draw_backend_join_round:</code>	841 <code>] ~</code>
<code>__draw_backend_join_bevel:</code>	842 <code>\dim_to_decimal_in_bp:n {#2} ~ d</code>
	843 <code>}</code>
	844 <code>}</code>
	845 <code>\cs_new:Npn __draw_backend_dash:n #1</code>
	846 <code>{ ~ \dim_to_decimal_in_bp:n {#1} }</code>
	847 <code>\cs_new_protected:Npn __draw_backend_linewidth:n #1</code>
	848 <code>{</code>
	849 <code>__draw_backend_literal:x</code>
	850 <code>{ \dim_to_decimal_in_bp:n {#1} ~ w }</code>
	851 <code>}</code>
	852 <code>\cs_new_protected:Npn __draw_backend_miterlimit:n #1</code>
	853 <code>{ __draw_backend_literal:x { \fp_eval:n {#1} ~ M } }</code>
	854 <code>\cs_new_protected:Npn __draw_backend_cap_butt:</code>
	855 <code>{ __draw_backend_literal:n { 0 ~ J } }</code>
	856 <code>\cs_new_protected:Npn __draw_backend_cap_round:</code>

```

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

```



```

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

```

919 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
920 {
921   \<*pdfmode>
922   \__kernel_backend_matrix:x
923   {
924     \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
925     \fp_eval:n {#3} ~ \fp_eval:n {#4}
926   }
927   \</pdfmode>
928   \<*dvipdfmx | xdvipdfmx>
929   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
930   \__draw_backend_cm_aux:nnnn
931   \</dvipdfmx | xdvipdfmx>
932 }
933 \<*dvipdfmx | xdvipdfmx>
934 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
935 {
936   \__kernel_backend_literal:x
937   {
938     x:rotate~
939     \fp_compare:nNnTF {#1} = \c_zero_fp
940     { 0 }
941     { \fp_eval:n { round ( -#1 , 5 ) } }
942   }
943   \__kernel_backend_literal:x
944   {
945     x:scale~
946     \fp_eval:n { round ( #2 , 5 ) } ~
947     \fp_eval:n { round ( #3 , 5 ) }
948   }
949   \__kernel_backend_literal:x
950   {

```

```

951      x:rotate~
952      \fp_compare:nNnTF {#4} = \c_zero_fp
953        { 0 }
954        { \fp_eval:n { round ( -#4 , 5 ) } }
955    }
956  }
957  </dviptdmtx | xdvipdfmtx>

```

(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.

```

958  <*dviptdmtx | xdvipdfmtx>
959  \cs_new_protected:Npn \_draw_backend_cm_decompose:nnnnN #1#2#3#4#5
960  {
961    \use:x
962    {
963      \_draw_backend_cm_decompose_auxi:nnnnN
964      { \fp_eval:n { (#1 + #4) / 2 } }
965      { \fp_eval:n { (#1 - #4) / 2 } }
966      { \fp_eval:n { (#3 + #2) / 2 } }
967      { \fp_eval:n { (#3 - #2) / 2 } }
968    }
969    #5
970  }
971  \cs_new_protected:Npn \_draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
972  {
973    \use:x

```

```

974 {
975   \_draw_backend_cm_decompose_auxii:nnnnN
976   { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
977   { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
978   { \fp_eval:n { atand ( #3 , #2 ) } }
979   { \fp_eval:n { atand ( #4 , #1 ) } }
980 }
981 #5
982 }
983 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
984 {
985   \use:x
986   {
987     \_draw_backend_cm_decompose_auxiii:nnnnN
988     { \fp_eval:n { ( #4 - #3 ) / 2 } }
989     { \fp_eval:n { ( #1 + #2 ) / 2 } }
990     { \fp_eval:n { ( #1 - #2 ) / 2 } }
991     { \fp_eval:n { ( #4 + #3 ) / 2 } }
992   }
993   #5
994 }
995 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
996 {
997   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
998   { #5 {#1} {#2} {#3} {#4} }
999   { #5 {#1} {#3} {#2} {#4} }
1000 }
1001 \</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.

```

1002 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
1003 {
1004   \_kernel_backend_scope_begin:
1005   \<pdfmode>
1006   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1007   \</pdfmode>
1008   \<dvipdfmx | xdvipdfmx>
1009   \_kernel_backend_literal:x
1010   {
1011     pdf:btrans-matrix~
1012     \fp_eval:n {#2} ~ \fp_eval:n {#3} ~
1013     \fp_eval:n {#4} ~ \fp_eval:n {#5} ~
1014     0 ~ 0
1015   }
1016   \</dvipdfmx | xdvipdfmx>
1017   \hbox_overlap_right:n { \box_use:N #1 }
1018   \<dvipdfmx | xdvipdfmx>
1019   \_kernel_backend_literal:n { pdf:etrans }

```

```

1020 </dvipdfmx | xdvipdfmx>
1021     \__kernel_backend_scope_end:
1022 }

(End definition for \__draw_backend_box_use:Nnnnn.)

1023 </dvipdfmx | pdfmode | xdvipdfmx>

```

4.3 dvisvgm backend

```

1024 <*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
1025 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1026 \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

```

1027 \cs_new_protected:Npn \__draw_backend_begin:
1028 {
1029     \__draw_backend_scope_begin:
1030     \__draw_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1031 }
1032 \cs_new_protected:Npn \__draw_backend_end:
1033 { \__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.

```

1034 \cs_new_protected:Npn \__draw_backend_scope_begin:
1035 {
1036     \int_set_eq:NN
1037     \l__draw_draw_scope_int
1038     \g__draw_draw_scope_int
1039     \group_begin:
1040     \int_gzero:N \g__draw_draw_scope_int
1041 }
1042 \cs_new_protected:Npn \__draw_backend_scope_end:
1043 {
1044     \prg_replicate:nn
1045     { \g__draw_draw_scope_int }
1046     { \__draw_backend_literal:n { </g> } }
1047     \group_end:
1048     \int_gset_eq:NN
1049     \g__draw_draw_scope_int
1050     \l__draw_draw_scope_int
1051 }
1052 \cs_new_protected:Npn \__draw_backend_scope:n #1

```

```

1053 {
1054   \_draw_backend_literal:n { <g~ #1 > }
1055   \int_gincr:N \g__draw_draw_scope_int
1056 }
1057 \cs_generate_variant:Nn \_draw_backend_scope:n { x }
1058 \int_new:N \g__draw_draw_scope_int
1059 \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
1060 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1061 {
1062   \_draw_backend_add_to_path:n
1063   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1064 }
1065 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1066 {
1067   \_draw_backend_add_to_path:n
1068   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1069 }
1070 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1071 {
1072   \_draw_backend_add_to_path:n
1073   {
1074     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1075     h ~ \dim_to_decimal:n {#3} ~
1076     v ~ \dim_to_decimal:n {#4} ~
1077     h ~ \dim_to_decimal:n { -#3 } ~
1078     Z
1079   }
1080 }
1081 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1082 {
1083   \_draw_backend_add_to_path:n
1084   {
1085     C ~
1086     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1087     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1088     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1089   }
1090 }
1091 \cs_new_protected:Npn \_draw_backend_add_to_path:n #1
1092 {
1093   \tl_gset:Nx \g__draw_draw_path_tl
1094   {
1095     \g__draw_draw_path_tl
1096     \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1097     #1
1098   }

```

```

1099 }
1100 \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:
1101 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1102 { \__draw_backend_scope:n { fill-rule="evenodd" } }
1103 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1104 { \__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:
1105 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_clip: 1106 { \__draw_backend_add_to_path:n { Z } }
\__draw_backend_discardpath: 1107 \cs_new_protected:Npn \__draw_backend_path:n #1
\g__draw_draw_clip_bool 1108 {
\g__draw_draw_path_int 1109 \bool_if:NTF \g__draw_draw_clip_bool
1110 {
1111 \int_gincr:N \g__draw_clip_path_int
1112 \__draw_backend_literal:x
1113 {
1114 < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1115 { ?nl }
1116 <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1117 < /clipPath > { ? nl }
1118 <
1119 use~xlink:href =
1120 "\c_hash_str l3path \int_use:N \g__draw_path_int " ~
1121 #1
1122 />
1123 }
1124 \__draw_backend_scope:x
1125 {
1126 clip-path =
1127 "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1128 }
1129 }
1130 {
1131 \__draw_backend_literal:x
1132 { <path ~ d=" \g__draw_draw_path_tl " ~ #1 /> }
1133 }
1134 \tl_gclear:N \g__draw_draw_path_tl
1135 \bool_gset_false:N \g__draw_draw_clip_bool
1136 }
1137 \int_new:N \g__draw_path_int
1138 \cs_new_protected:Npn \__draw_backend_stroke:
1139 { \__draw_backend_path:n { style="fill:none" } }
1140 \cs_new_protected:Npn \__draw_backend_closestroke:

```

```

1141 {
1142   \_draw_backend_closepath:
1143   \_draw_backend_stroke:
1144 }
1145 \cs_new_protected:Npn \_draw_backend_fill:
1146 { \_draw_backend_path:n { style="stroke:none" } }
1147 \cs_new_protected:Npn \_draw_backend_fillstroke:
1148 { \_draw_backend_path:n { } }
1149 \cs_new_protected:Npn \_draw_backend_clip:
1150 { \bool_gset_true:N \g__draw_draw_clip_bool }
1151 \bool_new:N \g__draw_draw_clip_bool
1152 \cs_new_protected:Npn \_draw_backend_discardpath:
1153 {
1154   \bool_if:NT \g__draw_draw_clip_bool
1155   {
1156     \int_gincr:N \g__draw_clip_path_int
1157     \_draw_backend_literal:x
1158     {
1159       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1160       { ?nl }
1161       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1162       < /clipPath >
1163     }
1164     \_draw_backend_scope:x
1165     {
1166       clip-path =
1167       "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int )"
1168     }
1169   }
1170   \tl_gclear:N \g__draw_draw_path_tl
1171   \bool_gset_false:N \g__draw_draw_clip_bool
1172 }

```

(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:
1173 \cs_new_protected:Npn \_draw_backend_dash_pattern:nn #1#2
1174 {
1175   \use:x
1176   {
1177     \_draw_backend_dash_aux:nn
1178     { \clist_map_function:nn {#1} \_draw_backend_dash:n }
1179     { \dim_to_decimal:n {#2} }
1180   }
1181 }
1182 \cs_new:Npn \_draw_backend_dash:n #1
1183 { , \dim_to_decimal_in_bp:n {#1} }
1184 \cs_new_protected:Npn \_draw_backend_dash_aux:nn #1#2
1185 {
1186   \_draw_backend_scope:x
1187   {
1188     stroke-dasharray =
1189     "

```

```

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

```

(End definition for __draw_backend_dash_pattern:nn and others.)

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

```

\_draw_backend_color_fill_cmyk:nnnn 1213 \cs_new_protected:Npn \__draw_backend_color_fill_cmyk:nnnn #1#2#3#4
\_draw_backend_color_stroke_cmyk:nnnn 1214 {
\_draw_backend_color_fill_gray:n 1215     \use:x
\_draw_backend_color_stroke_gray:n 1216     {
\_draw_backend_color_fill_rgb:nnn 1217         \__draw_backend_color_fill:nnn
\_draw_backend_color_stroke_rgb:nnn 1218         { \fp_eval:n { -100 * ( (#1) * ( 1 - (#4) ) - 1 ) } }
\_draw_backend_color_fill:nnn 1219         { \fp_eval:n { -100 * ( (#2) * ( 1 - (#4) ) + #4 - 1 ) } }
1220         { \fp_eval:n { -100 * ( (#3) * ( 1 - (#4) ) + #4 - 1 ) } }
1221     }
1222 }
1223 \cs_new_protected:Npn \__draw_backend_color_stroke_cmyk:nnnn #1#2#3#4
1224 {
1225     \__draw_backend_select:x
1226     {
1227         cmyk~
1228         \fp_eval:n {#1} ~ \fp_eval:n {#2} ~
1229         \fp_eval:n {#3} ~ \fp_eval:n {#4}
1230     }
1231 }
1232 \cs_new_protected:Npn \__draw_backend_color_fill_gray:n #1
1233 {
1234     \use:x
1235     {
1236         \__draw_backend_color_gray_aux:n
1237         { \fp_eval:n { 100 * (#1) } }
1238     }

```



```

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

```

(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.

```

1274 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1275 {
1276   \__draw_backend_scope:n
1277   {
1278     transform =
1279     "
1280     matrix
1281     (
1282       \fp_eval:n {#1} , \fp_eval:n {#2} ,
1283       \fp_eval:n {#3} , \fp_eval:n {#4} ,
1284       0pt , 0pt
1285     )
1286     "
1287   }
1288 }

```

(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.

```

1289 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1290 {
1291   \_kernel_backend_scope_begin:
1292   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1293   \_kernel_backend_literal_svg:n
1294   {
1295     < g~
1296     stroke="none"~
1297     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1298     >
1299   }
1300   \box_set_wd:Nn #1 { Opt }
1301   \box_set_ht:Nn #1 { Opt }
1302   \box_set_dp:Nn #1 { Opt }
1303   \box_use:N #1
1304   \_kernel_backend_literal_svg:n { </g> }
1305   \_kernel_backend_scope_end:
1306 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```

1307 </divisvgm>
1308 </initex | package>

```

5 l3backend-graphics Implementation

```

1309 <*initex | package>
1310 <@@=graphics>

```

5.1 dvips backend

```

1311 <*dvips>

```

`_graphics_backend_getbb_eps:n` Simply use the generic function.

```

1312 <*initex>
1313 \use:n
1314 </initex>
1315 <*package>
1316 \AtBeginDocument
1317 </package>
1318 { \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.

```

1319 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1320 {
1321   \_kernel_backend_literal:x
1322   {
1323     PSfile = #1 \c_space_tl
1324     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl

```

```

1325         lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1326         urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1327         ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1328     }
1329 }

(End definition for \__graphics_backend_include_eps:n.)

1330 </dvips>

```

5.2 pdfmode backend

```

1331 (*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.

```

1332 \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.

```

1333 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1334 {
1335     \int_zero:N \l_graphics_page_int
1336     \tl_clear:N \l_graphics_pagebox_tl
1337     \tl_set:Nx \l__graphics_graphics_attr_tl
1338     {
1339         \tl_if_empty:NF \l_graphics_decodearray_tl
1340         { :D \l_graphics_decodearray_tl }
1341         \bool_if:NT \l_graphics_interpolate_bool
1342         { :I }
1343     }
1344     \tl_clear:N \l__graphics_graphics_attr_tl
1345     \__graphics_backend_getbb_auxi:n {#1}
1346 }
1347 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1348 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1349 {
1350     \tl_clear:N \l_graphics_decodearray_tl
1351     \bool_set_false:N \l_graphics_interpolate_bool
1352     \tl_set:Nx \l__graphics_graphics_attr_tl
1353     {
1354         : \l_graphics_pagebox_tl
1355         \int_compare:nNnT \l_graphics_page_int > 1
1356         { :P \int_use:N \l_graphics_page_int }
1357     }
1358     \__graphics_backend_getbb_auxi:n {#1}
1359 }

```

```

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

```

(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.

```

1399 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1400 {
1401   \tex_pdfrefximage:D
1402   \int_use:c { c__graphics_graphics_#1 \l__graphics_graphics_attr_tl _int }
1403 }
1404 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1405 \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
\l__graphics_backend_name_str
\l__graphics_backend_ext_str

1406 \sys_if_shell:T
1407 {
1408   \str_new:N \l__graphics_backend_dir_str
1409   \str_new:N \l__graphics_backend_name_str
1410   \str_new:N \l__graphics_backend_ext_str
1411   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1412   {
1413     \file_parse_full_name:nNNN {#1}
1414     \l__graphics_backend_dir_str
1415     \l__graphics_backend_name_str
1416     \l__graphics_backend_ext_str
1417     \exp_args:Nx \__graphics_backend_getbb_eps:nn
1418     {
1419       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1420       -converted-to.pdf
1421     }
1422     {#1}
1423   }
1424   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1425   {
1426     \file_compare_timestamp:nNnT {#2} > {#1}
1427     {
1428       \sys_shell_now:n
1429       { repstopdf ~ #2 ~ #1 }
1430     }
1431     \tl_set:Nn \l_graphics_name_tl {#1}
1432     \__graphics_backend_getbb_pdf:n {#1}
1433   }
1434   \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1435   {
1436     \file_parse_full_name:nNNN {#1}
1437     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1438     \exp_args:Nx \__graphics_backend_include_pdf:n
1439     {
1440       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1441       -converted-to.pdf
1442     }
1443   }
1444 }

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

1445 </pdfmode>

```

5.3 dvipdfmx backend

```

1446 <*dvipdfmx | xdvipdfmx>

\__graphics_backend_getbb_eps:n Simply use the generic functions: only for dvipdfmx in the extraction cases.
\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
1447 <*initex>
1448 \use:n
1449 </initex>

```

```

1450 <*package>
1451 \AtBeginDocument
1452 </package>
1453 { \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n }
1454 <*dvipdfmx>
1455 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1456 {
1457   \int_zero:N \l_graphics_page_int
1458   \tl_clear:N \l_graphics_pagebox_tl
1459   \graphics_extract_bb:n {#1}
1460 }
1461 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1462 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1463 {
1464   \tl_clear:N \l_graphics_decodearray_tl
1465   \bool_set_false:N \l_graphics_interpolate_bool
1466   \graphics_extract_bb:n {#1}
1467 }
1468 </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.

```

1469 \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> 1470 \cs_new_protected:Npn __graphics_backend_include_eps:n #1 1471 { 1472 __kernel_backend_literal:x 1473 { 1474 PSfile = #1 \c_space_tl 1475 llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl 1476 lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl 1477 urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl 1478 ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim 1479 } 1480 } 1481 \cs_new_protected:Npn __graphics_backend_include_jpg:n #1 1482 { __graphics_backend_include_auxi:nn {#1} { image } } 1483 \cs_new_eq:NN __graphics_backend_include_png:n __graphics_backend_include_jpg:n 1484 <*dvipdfmx> 1485 \cs_new_protected:Npn __graphics_backend_include_pdf:n #1 1486 { __graphics_backend_include_auxi:nn {#1} { epdf } } 1487 </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.

```

1488 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2

```

```

1489 {
1490   \__graphics_backend_include_auxii:xnn
1491   {
1492     \tl_if_empty:NF \l_graphics_pagebox_tl
1493     { : \l_graphics_pagebox_tl }
1494     \int_compare:nNnT \l_graphics_page_int > 1
1495     { :P \int_use:N \l_graphics_page_int }
1496     \tl_if_empty:NF \l_graphics_decodearray_tl
1497     { :D \l_graphics_decodearray_tl }
1498     \bool_if:NT \l_graphics_interpolate_bool
1499     { :I }
1500   }
1501   {#1} {#2}
1502 }
1503 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1504 {
1505   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1506   {
1507     \__kernel_backend_literal:x
1508     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1509   }
1510   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1511 }
1512 \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!

```

1513 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1514 {
1515   \int_gincr:N \g__graphics_track_int
1516   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1517   \__kernel_backend_literal:x
1518   {
1519     pdf:#3~
1520     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1521     \int_compare:nNnT \l_graphics_page_int > 1
1522     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1523     \tl_if_empty:NF \l_graphics_pagebox_tl
1524     {
1525       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1526       bbox ~
1527         \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1528         \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1529         \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1530         \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1531     }
1532     (#1)
1533     \bool_lazy_or:nnT
1534     { \l_graphics_interpolate_bool }
1535     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1536     {
1537       <<
1538       \tl_if_empty:NF \l_graphics_decodearray_tl

```

```

1539         { /Decode~[ \l_graphics_decodearray_tl ] }
1540         \bool_if:NT \l_graphics_interpolate_bool
1541         { /Interpolate~true> }
1542     >>
1543 }
1544 }
1545 }

```

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

```

1546 </dvipdfmx | xdvipdfmx>

```

5.4 xdvipdfmx backend

```

1547 < *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

```

```

1548 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1549 {
1550     \int_zero:N \l_graphics_page_int
1551     \tl_clear:N \l_graphics_pagebox_tl
1552     \_graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1553 }
1554 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1555 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1556 {
1557     \tl_clear:N \l_graphics_decodearray_tl
1558     \bool_set_false:N \l_graphics_interpolate_bool
1559     \_graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1560 }
1561 \cs_new_protected:Npn \_graphics_backend_getbb_auxi:nN #1#2
1562 {
1563     \int_compare:nNnTF \l_graphics_page_int > 1
1564     { \_graphics_backend_getbb_auxii:nNn \l_graphics_page_int {#1} #2 }
1565     { \_graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1566 }
1567 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:nNn #1#2#3
1568 { \_graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1569 \cs_generate_variant:Nn \_graphics_backend_getbb_auxii:nNn { V }
1570 \cs_new_protected:Npn \_graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1571 {
1572     \tl_if_empty:NTF \l_graphics_pagebox_tl
1573     { \_graphics_backend_getbb_auxiv:nNnn \l_graphics_pagebox_tl }
1574     { \_graphics_backend_getbb_auxv:nNnn }
1575     {#1} #2 {#3} {#4}
1576 }
1577 \cs_new_protected:Npn \_graphics_backend_getbb_auxiv:nNnn #1#2#3#4#5
1578 {
1579     \use:x
1580     {

```



```

1581         \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1582         { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1583     }
1584 }
1585 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1586 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1587 {
1588     \graphics_bb_restore:nF {#1#3}
1589     { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1590 }
1591 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1592 {
1593     \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1594     \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1595     \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1596     \graphics_bb_save:n {#1#3}
1597 }
1598 \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

__graphics_backend_include_pdf:n
__graphics_backend_include_bitmap_quote:w

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

```

1599 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1600 {
1601     \tex_XeTeXpdffile:D
1602     \__graphics_backend_include_pdf_quote:w #1 "#1" \q_stop \c_space_tl
1603     \int_compare:nNnT \l_graphics_page_int > 0
1604     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1605     \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1606 }
1607 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \q_stop
1608 { " #2 " }

```

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

```

1609 </xdvipdfmx>

```

5.5 dvisvgm backend

```

1610 <*dvisvgm>

```

__graphics_backend_getbb_eps:n Simply use the generic function.

```

1611 <*initex>
1612 \use:n
1613 </initex>
1614 <*package>
1615 \AtBeginDocument
1616 </package>
1617 { \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`

```

1618 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1619 {
1620   \int_zero:N \l_graphics_page_int
1621   \tl_clear:N \l_graphics_pagebox_tl
1622   \graphics_extract_bb:n {#1}
1623 }
1624 \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

```

1625 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1626 {
1627   \tl_clear:N \l_graphics_decodearray_tl
1628   \bool_set_false:N \l_graphics_interpolate_bool
1629   \graphics_extract_bb:n {#1}
1630 }

```

(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`

```

1631 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1632 { \_graphics_backend_include:nn { PSfile } {#1} }
1633 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1634 { \_graphics_backend_include:nn { pdffile } {#1} }
1635 \cs_new_protected:Npn \_graphics_backend_include:nn #1#2
1636 {
1637   \__kernel_backend_literal:x
1638   {
1639     #1 = #2 \c_space_tl
1640     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1641     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1642     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1643     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1644   }
1645 }

```

(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.

```

1646 \cs_new_protected:Npn \_graphics_backend_include_png:n #1
1647 {
1648   \__kernel_backend_literal:x
1649   {
1650     dvisvgm:img~
1651     \dim_to_decimal:n { \l_graphics_ury_dim } ~
1652     \dim_to_decimal:n { \l_graphics_ury_dim } ~

```

```

1653         \_graphics_backend_include_bitmap_quote:w #1 " #1 " \q_stop
1654     }
1655 }
1656 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_png:n
1657 \cs_new:Npn \_graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \q_stop
1658 { " #2 " }

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

1659 \</divsvgm>
1660 \</initex | package>

```

6 l3backend-pdf Implementation

```

1661 <*initex | package>
1662 <@@=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
1663 \box_new:N \l\_pdf\_internal\_box

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

```

6.2 dvips backend

```

1664 <*dvips>

\_pdf\_backend\_pdfmark:n Used often enough it should be a separate function.
\_pdf\_backend\_pdfmark:x
1665 \cs_new_protected:Npn \_pdf\_backend\_pdfmark:n #1
1666 { \_kernel_backend_postscript:n { mark #1 ~ pdfmark } }
1667 \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
1668 \cs_new_protected:Npn \_pdf\_backend\_catalog\_gput:nn #1#2
1669 { \_pdf\_backend\_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
1670 \cs_new_protected:Npn \_pdf\_backend\_info\_gput:nn #1#2
1671 { \_pdf\_backend\_pdfmark:n { /#1 ~ #2 /DOCINFO } }

(End definition for \_pdf\_backend\_catalog\_gput:nn and \_pdf\_backend\_info\_gput:nn.)

```

6.2.2 Objects

`\g__pdf_backend_object_int`
`\g__pdf_backend_object_prop`

For tracking objects to allow finalisation.

```
1672 \int_new:N \g__pdf_backend_object_int
1673 \prop_new:N \g__pdf_backend_object_prop
```

(End definition for `\g__pdf_backend_object_int` and `\g__pdf_backend_object_prop`.)

`__pdf_backend_object_new:nn`
`__pdf_backend_object_ref:n`

Tracking objects is similar to dvipdfmx.

```
1674 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
1675 {
1676   \int_gincr:N \g__pdf_backend_object_int
1677   \int_const:cn
1678     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
1679     { \g__pdf_backend_object_int }
1680   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
1681 }
1682 \cs_new:Npn \__pdf_backend_object_ref:n #1
1683 { { pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } } }
```

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

`__pdf_backend_object_write:nn`
`__pdf_backend_object_write:nx`
`__pdf_backend_object_write_array:nn`
`__pdf_backend_object_write_dict:nn`
`__pdf_backend_object_write_stream:nn`
`__pdf_backend_object_write_stream:nnn`

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

```
1684 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
1685 {
1686   \__pdf_backend_pdfmark:x
1687   {
1688     /_objdef ~ \__pdf_backend_object_ref:n {#1}
1689     /type
1690     \str_case_e:nn
1691       { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
1692       {
1693         { array } { /array }
1694         { dict } { /dict }
1695         { fstream } { /stream }
1696         { stream } { /stream }
1697       }
1698     /OBJ
1699   }
1700   \use:c
1701     { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
1702     { \__pdf_backend_object_ref:n {#1} } {#2}
1703 }
1704 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
1705 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
1706 {
1707   \__pdf_backend_pdfmark:x
1708     { #1 [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
1709 }
1710 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
1711 {
1712   \__pdf_backend_pdfmark:x
1713     { #1 << \exp_not:n {#2} >> /PUT }
1714 }
1715 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
```

```

1716 {
1717   \exp_args:Nx
1718   \__pdf_backend_object_write_stream:nnn {#1} #2
1719 }
1720 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
1721 {
1722   \__kernel_backend_postscript:n
1723   {
1724     [nobreak]
1725     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
1726     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
1727   }
1728 }

```

(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 1729 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
1730 {
1731   \int_gincr:N \g__pdf_backend_object_int
1732   \__pdf_backend_pdfmark:x
1733   {
1734     /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
1735     /type
1736     \str_case:nn
1737       {#1}
1738       {
1739         { array } { /array }
1740         { dict } { /dict }
1741         { fstream } { /stream }
1742         { stream } { /stream }
1743       }
1744     /OBJ
1745   }
1746   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
1747   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
1748 }
1749 \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.

```

1750 \cs_new:Npn \__pdf_backend_object_last:
1751 { { 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.

```

1752 \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.

1753 \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.

1754 \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.

1755 \cs_new_protected:Npn _pdf_backend_annotation:nnnn #1#2#3#4

1756 {

1757 _pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4}

1758 \int_gincr:N \g__pdf_backend_object_int

1759 \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int

1760 _pdf_backend_pdfmark:x

1761 {

1762

1763 /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }

1764 pdf.rect ~

1765 #4 ~

1766 /ANN

1767 }

1768 }

1769 \cs_new_protected:Npn _pdf_backend_annotation_aux:nnnn #1#2#3#4

1770 {

1771 \box_move_down:nn {#3}

1772 { \hbox:n { _kernel_backend_postscript:n { pdf.save.ll } } }

1773 \hbox:n {#4}

1774 \box_move_up:nn {#2}

1775 {

1776 \hbox:n

1777 {

1778 \tex_kern:D \dim_eval:n {#1} \scan_stop:

1779 _kernel_backend_postscript:n { pdf.save.ur }

1780 }

1781 }

1782 \int_gincr:N \g__pdf_backend_object_int

1783 \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int

1784 _pdf_backend_pdfmark:x

1785 {

1786 /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }

1787 pdf.rect

1788 /ANN

1789 }

1790 }

(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.

```
1791 \cs_new:Npn \_pdf_backend_annotation_last:
1792   { { 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.

```
1793 \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.

```
1794 \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.

```
1795 \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.

```
1796 \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.

```
1797 \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.

```
1798 \tl_new:N \l__pdf_breaklink_pdfmark_tl
1799 \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.

```
1800 \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.

```
1801 \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.

```

1802 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
1803 { \__pdf_backend_link_begin:nw { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
1804 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
1805 { \__pdf_backend_link_begin:nw {#1#2} }
1806 \cs_new_protected:Npn \__pdf_backend_link_begin:nw #1
1807 {
1808   \bool_if:NF \g__pdf_backend_link_bool
1809   { \__pdf_backend_link_begin_aux:nw {#1} }
1810 }
1811 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
1812 {
1813   \bool_gset_true:N \g__pdf_backend_link_bool
1814   \__kernel_backend_postscript:n
1815   { /pdf.link.dict ( #1 ) def }
1816   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
1817   \__pdf_backend_link_sf_save:
1818   \mode_if_math:TF
1819   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
1820   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
1821   \hbox_set:Nw \l__pdf_backend_content_box
1822   \__pdf_backend_link_sf_restore:
1823   \bool_if:NT \g__pdf_backend_link_math_bool
1824   { \c_math_toggle_token }
1825 }
1826 \cs_new_protected:Npn \__pdf_backend_link_end:
1827 {
1828   \bool_if:NT \g__pdf_backend_link_bool
1829   { \__pdf_backend_link_end_aux: }
1830 }
1831 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
1832 {
1833   \bool_if:NT \g__pdf_backend_link_math_bool
1834   { \c_math_toggle_token }
1835   \__pdf_backend_link_sf_save:
1836   \hbox_set_end:
1837   \__pdf_backend_link_minima:
1838   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
1839   \exp_args:Nx \__pdf_backend_link_outerbox:n
1840   {

```



```

1841 <*initex>
1842     \l_galley_total_left_margin_dim
1843 </initex>
1844 <*package>
1845     \int_if_odd:nTF { \value { page } }
1846         { \oddsidemargin }
1847         { \evensidemargin }
1848 </package>
1849 }
1850 \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
1851 { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
1852 \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
1853 \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
1854 \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
1855 \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
1856 {
1857     \hbox:n
1858     { \__kernel_backend_postscript:n { pdf.save.linkur } }
1859 }
1860 \int_gincr:N \g__pdf_backend_object_int
1861 \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
1862 \__kernel_backend_postscript:x
1863 {
1864     mark
1865     /objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
1866     \g__pdf_backend_link_dict_tl \c_space_tl
1867     pdf.rect
1868     /ANN ~ \l__pdf_breaklink_pdfmark_tl
1869 }
1870 \__pdf_backend_link_sf_restore:
1871 \bool_gset_false:N \g__pdf_backend_link_bool
1872 }
1873 \cs_new_protected:Npn \__pdf_backend_link_minima:
1874 {
1875     \hbox_set:Nn \l__pdf_backend_model_box { Gg }
1876     \__kernel_backend_postscript:x
1877     {
1878         /pdf.linkdp.pad ~
1879         \dim_to_decimal:n
1880         {
1881             \dim_max:nn
1882             {
1883                 \box_dp:N \l__pdf_backend_model_box
1884                 - \box_dp:N \l__pdf_backend_content_box
1885             }
1886             { Opt }
1887         } ~
1888         pdf.pt.dvi ~ def
1889         /pdf.linkht.pad ~
1890         \dim_to_decimal:n
1891         {
1892             \dim_max:nn
1893             {
1894                 \box_ht:N \l__pdf_backend_model_box

```

```

1895         - \box_ht:N \l__pdf_backend_content_box
1896     }
1897     { Opt }
1898 } ~
1899 pdf.pt.dvi ~ def
1900 }
1901 }
1902 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
1903 {
1904     \__kernel_backend_postscript:x
1905     {
1906         /pdf.outerbox
1907         [
1908             \dim_to_decimal:n {#1} ~
1909             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
1910 (*initex)
1911             \dim_to_decimal:n { #1 + \l_galley_text_width_dim } ~
1912 
1913 (*package)
1914             \dim_to_decimal:n { #1 + \textwidth } ~
1915 
1916             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
1917         ]
1918         [ exch { pdf.pt.dvi } forall ] def
1919         /pdf.baselineskip ~
1920         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
1921         { pdf.pt.dvi ~ def }
1922         { pop ~ pop }
1923         ifelse
1924     }
1925 }
1926 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
1927 {
1928     \int_gset:Nn \g__pdf_backend_link_sf_int
1929     {
1930         \mode_if_horizontal:TF
1931         { \tex_spacefactor:D }
1932         { 0 }
1933     }
1934 }
1935 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
1936 {
1937     \mode_if_horizontal:T
1938     {
1939         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
1940         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
1941     }
1942 }

```

(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.

```

1943 <*package>
1944 \use_none:n
1945 {
1946   \cs_if_exist:NT \@makecol@hook
1947   {
1948     \tl_put_right:Nn \@makecol@hook
1949     {
1950       \box_if_empty:NF \@cclv
1951       {
1952         \vbox_set:Nn \@cclv
1953         {
1954           \__kernel_backend_postscript:n
1955           {
1956             pdf.globaldict /pdf.brokenlink.rect ~ known
1957             { pdf.bordertracking.continue }
1958             if
1959             }
1960             \vbox_unpack_drop:N \@cclv
1961             \__kernel_backend_postscript:n
1962             { pdf.bordertracking.endpage }
1963           }
1964         }
1965       }
1966       \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
1967       \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
1968       \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
1969     }
1970   }
1971 </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.

```

1972 \cs_new:Npn \__pdf_backend_link_last:
1973 { { 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.

```

1974 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
1975 {
1976   \__kernel_backend_postscript:x
1977   {
1978     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
1979   }
1980 }

```

(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.

```

1981 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2

```

```

1982 {
1983   \_kernel_backend_postscript:n { pdf.dest.anchor }
1984   \_pdf_backend_pdfmark:x
1985   {
1986     /View
1987     [
1988       \str_case:nnF {#2}
1989       {
1990         { xyz } { /XYZ ~ pdf.dest.point ~ null }
1991         { fit } { /Fit }
1992         { fitb } { /FitB }
1993         { fitbh } { /FitBH ~ pdf.dest.y }
1994         { fitbv } { /FitBV ~ pdf.dest.x }
1995         { fith } { /FitH ~ pdf.dest.y }
1996         { fitv } { /FitV ~ pdf.dest.x }
1997       }
1998       {
1999         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2000       }
2001     ]
2002     /Dest ( \exp_not:n {#1} ) cvn
2003     /DEST
2004   }
2005 }
2006 \cs_new_protected:Npn \_pdf_backend_destination_rectangle:nn #1#2
2007 {
2008   \group_begin:
2009     \hbox_set:Nn \l__pdf_internal_box {#2}
2010     \box_move_down:nn
2011       { \box_dp:N \l__pdf_internal_box }
2012       { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2013     \box_use:N \l__pdf_internal_box
2014     \box_move_up:nn
2015       { \box_ht:N \l__pdf_internal_box }
2016       { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ur } } }
2017     \_pdf_backend_pdfmark:n
2018     {
2019       /View
2020       [
2021         /FitR ~
2022         pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2023         pdf.urx ~ pdf.ury ~ pdf.dest2device
2024       ]
2025       /Dest ( #1 ) cvn
2026       /DEST
2027     }
2028   \group_end:
2029 }

```

(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

```

2030 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2031 \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
2032 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2033 \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:
2034 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2035 \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:
2036 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2037 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2038 \cs_new_protected:Npn \__pdf_backend_emc:
2039 { \__pdf_backend_pdfmark:n { /EMC } }
(End definition for \__pdf_backend_bdc:nn and \__pdf_backend_emc:.)
2040 </dvips>

```

6.3 pdfmode backend

```

2041 <*pdfmode>

```

6.3.1 Annotations

```

\__pdf_backend_annotation:nnnn Simply pass the raw data through, just dealing with evaluation of dimensions.
2042 \cs_new_protected:Npx \__pdf_backend_annotation:nnnn #1#2#3#4
2043 {
2044   \cs_if_exist:NTF \tex_pdfextension:D
2045   { \tex_pdfextension:D annot ~ }
2046   { \tex_pdfannot:D }
2047   width ~ \exp_not:N \dim_eval:n {#1} ~
2048   height ~ \exp_not:N \dim_eval:n {#2} ~
2049   depth ~ \exp_not:N \dim_eval:n {#3} ~
2050   {#4}
2051 }
(End definition for \__pdf_backend_annotation:nnnn.)

```

```

\__pdf_backend_annotation_last: A tiny amount of extra data gets added here.
2052 \cs_new:Npx \__pdf_backend_annotation_last:
2053 {
2054   \exp_not:N \int_value:w
2055   \cs_if_exist:NTF \tex_pdffeedback:D
2056   { \exp_not:N \tex_pdffeedback:D lastannot ~ }
2057   { \exp_not:N \tex_pdflastannot:D }
2058   \c_space_tl 0 ~ R
2059 }

```

(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:
2060 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2061 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2062 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2063 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2064 \cs_new_protected:Npx \_pdf_backend_link_begin:nnnw #1#2#3
2065 {
2066   \cs_if_exist:NTF \tex_pdfextension:D
2067   { \tex_pdfextension:D startlink ~ }
2068   { \tex_pdfstartlink:D }
2069   attr {#1}
2070   #2 {#3}
2071 }
2072 \cs_new_protected:Npx \_pdf_backend_link_end:
2073 {
2074   \cs_if_exist:NTF \tex_pdfextension:D
2075   { \tex_pdfextension:D endlink \scan_stop: }
2076   { \tex_pdfendlink:D }
2077 }

```

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

`_pdf_backend_link_last:` Formatted for direct use.

```

2078 \cs_new:Npx \_pdf_backend_link_last:
2079 {
2080   \exp_not:N \int_value:w
2081   \cs_if_exist:NTF \tex_pdffeedback:D
2082   { \exp_not:N \tex_pdffeedback:D lastlink ~ }
2083   { \exp_not:N \tex_pdflastlink:D }
2084   \c_space_tl 0 ~ R
2085 }

```

(End definition for `_pdf_backend_link_last:`)

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

```

2086 \cs_new_protected:Npx \_pdf_backend_link_margin:n #1
2087 {
2088   \cs_if_exist:NTF \tex_pdfvariable:D
2089   { \exp_not:N \tex_pdfvariable:D linkmargin }
2090   { \exp_not:N \tex_pdflinkmargin:D }
2091   \exp_not:N \dim_eval:n {#1} \scan_stop:
2092 }

```

(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.

```

2093 \cs_new_protected:Npx \_pdf_backend_destination:nn #1#2
2094 {
2095   \cs_if_exist:NTF \tex_pdfextension:D
2096   { \exp_not:N \tex_pdfextension:D dest ~ }

```

```

2097 { \exp_not:N \tex_pdfdest:D }
2098   name {#1}
2099   \exp_not:N \str_case:nnF {#2}
2100   {
2101     { xyz } { xyz }
2102     { fit } { fit }
2103     { fitb } { fitb }
2104     { fitbh } { fitbh }
2105     { fitbv } { fitbv }
2106     { fith } { fith }
2107     { fitv } { fitv }
2108   }
2109   { xyz ~ zoom \exp_not:N \fp_eval:n { #2 * 10 } }
2110   \scan_stop:
2111 }
2112 \cs_new_protected:Npx \__pdf_backend_destination_rectangle:nn #1#2
2113 {
2114   \group_begin:
2115   \hbox_set:Nn \l__pdf_internal_box {#2}
2116   \cs_if_exist:NTF \tex_pdfextension:D
2117   { \exp_not:N \tex_pdfextension:D dest ~ }
2118   { \exp_not:N \tex_pdfdest:D }
2119   name {#1}
2120   fitr ~
2121   width \exp_not:N \box_wd:N \l__pdf_internal_box
2122   height \exp_not:N \box_ht:N \l__pdf_internal_box
2123   depth \exp_not:N \box_dp:N \l__pdf_internal_box
2124   \box_use:N \l__pdf_internal_box
2125   \group_end:
2126 }

```

(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
2127 \cs_new_protected:Npx \__pdf_backend_catalog_gput:nn #1#2
2128 {
2129   \cs_if_exist:NTF \tex_pdfextension:D
2130   { \tex_pdfextension:D catalog }
2131   { \tex_pdfcatalog:D }
2132   { / #1 ~ #2 }
2133 }
2134 \cs_new_protected:Npx \__pdf_backend_info_gput:nn #1#2
2135 {
2136   \cs_if_exist:NTF \tex_pdfextension:D
2137   { \tex_pdfextension:D info }
2138   { \tex_pdfinfo:D }
2139   { / #1 ~ #2 }
2140 }

```

(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.

```
2142 \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
2142 \group_begin:
2143 \cs_set_protected:Npn \__pdf_tmp:w #1#2
2144 {
2145   \cs_new_protected:Npx \__pdf_backend_object_new:nn ##1##2
2146   {
2147     #1 reserveobjnum ~
2148     \int_const:cn
2149     { c__pdf_backend_object_ \exp_not:N \tl_to_str:n {##1} _int }
2150     {#2}
2151     \prop_gput:Nnn \exp_not:N \g__pdf_backend_object_prop {##1} {##2}
2152   }
2153 }
2154 \cs_if_exist:NTF \tex_pdfextension:D
2155 {
2156   \__pdf_tmp:w
2157   { \tex_pdfextension:D obj ~ }
2158   { \exp_not:N \tex_pdffeedback:D lastobj }
2159 }
2160 { \__pdf_tmp:w { \tex_pdfobj:D } { \tex_pdflastobj:D } }
2161 \group_end:
2162 \cs_new:Npn \__pdf_backend_object_ref:n #1
2163 { \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
2164 \group_begin:
2165 \cs_set_protected:Npn \__pdf_tmp:w #1
2166 {
2167   \cs_new_protected:Npn \__pdf_backend_object_write:nn ##1##2
2168   {
2169     \tex_immediate:D #1 useobjnum ~
2170     \int_use:c
2171     { c__pdf_backend_object_ \tl_to_str:n {##1} _int }
2172     \str_case_e:nn
2173     { \prop_item:Nn \g__pdf_backend_object_prop {##1} }
2174     {
2175       { array } { { [ ~ \exp_not:n {##2} ~ ] } }
2176       { dict } { { < ~ \exp_not:n {##2} ~ > } }
2177       { fstream }
2178       {
2179         stream ~ attr ~ { \__pdf_exp_not_i:nn ##2 } ~
2180         file ~ { \__pdf_exp_not_ii:nn ##2 }
2181       }
2182     } stream }
2183     {
2184       stream ~ attr ~ { \__pdf_exp_not_i:nn ##2 } ~
```



```

2185         { \_pdf_exp_not_ii:nn ##2 }
2186     }
2187 }
2188 }
2189 }
2190 \cs_if_exist:NTF \tex_pdfextension:D
2191 { \_pdf_tmp:w { \tex_pdfextension:D obj ~ } }
2192 { \_pdf_tmp:w { \tex_pdfobj:D } }
2193 \group_end:
2194 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
2195 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2196 \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
2197 \group_begin:
2198 \cs_set_protected:Npn \_pdf_tmp:w #1
2199 {
2200     \cs_new_protected:Npn \_pdf_backend_object_now:nn ##1##2
2201     {
2202         \tex_immediate:D #1
2203         \str_case:nn
2204             {##1}
2205             {
2206                 { array } { { [ ~ \exp_not:n {##2} ~ ] } }
2207                 { dict } { { << ~ \exp_not:n {##2} ~ >> } }
2208                 { fstream }
2209                 {
2210                     stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2211                     file ~ { \_pdf_exp_not_ii:nn ##2 }
2212                 }
2213                 { stream }
2214                 {
2215                     stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2216                     { \_pdf_exp_not_ii:nn ##2 }
2217                 }
2218             }
2219     }
2220 }
2221 \cs_if_exist:NTF \tex_pdfextension:D
2222 { \_pdf_tmp:w { \tex_pdfextension:D obj ~ } }
2223 { \_pdf_tmp:w { \tex_pdfobj:D } }
2224 \group_end:
2225 \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.

```

2226 \cs_new:Npx \_pdf_backend_object_last:
2227 {
2228     \exp_not:N \int_value:w
2229     \cs_if_exist:NTF \tex_pdffeedback:D
2230     { \exp_not:N \tex_pdffeedback:D lastobj ~ }

```

```

2231     { \exp_not:N \tex_pdflastobj:D }
2232     \c_space_tl 0 ~ R
2233   }

```

(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
2234 \cs_new_protected:Npx \__pdf_backend_compresslevel:n #1
2235 {
2236   \exp_not:N \tex_global:D
2237   \cs_if_exist:NTF \tex_pdfcompresslevel:D
2238   { \tex_pdfcompresslevel:D }
2239   { \tex_pdfvariable:D compresslevel }
2240   \exp_not:N \int_value:w \exp_not:N \int_eval:n {#1} \scan_stop:
2241 }
2242 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2243 {
2244   \bool_if:nTF {#1}
2245   { \__pdf_backend_objcompresslevel:n { 2 } }
2246   { \__pdf_backend_objcompresslevel:n { 0 } }
2247 }
2248 \cs_new_protected:Npx \__pdf_backend_objcompresslevel:n #1
2249 {
2250   \exp_not:N \tex_global:D
2251   \cs_if_exist:NTF \tex_pdfobjcompresslevel:D
2252   { \tex_pdfobjcompresslevel:D }
2253   { \tex_pdfvariable:D objcompresslevel }
2254   #1 \scan_stop:
2255 }

```

(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
2256 \cs_new_protected:Npx \__pdf_backend_version_major_gset:n #1
2257 {
2258   \cs_if_exist:NTF \tex_pdfvariable:D
2259   {
2260     \int_compare:nNnT \tex luatexversion:D > { 106 }
2261     {
2262       \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2263       \exp_not:N \int_eval:n {#1} \scan_stop:
2264     }
2265   }
2266   {
2267     \cs_if_exist:NT \tex_pdfmajorversion:D
2268     {
2269       \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2270       \exp_not:N \int_eval:n {#1} \scan_stop:
2271     }
2272   }

```

```

2273 }
2274 \cs_new_protected:Npx \__pdf_backend_version_minor_gset:n #1
2275 {
2276   \exp_not:N \tex_global:D
2277   \cs_if_exist:NTF \tex_pdfminorversion:D
2278     { \exp_not:N \tex_pdfminorversion:D }
2279     { \tex_pdfvariable:D minorversion }
2280   \exp_not:N \int_eval:n {#1} \scan_stop:
2281 }

```

(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: 2282 \cs_new:Npx \__pdf_backend_version_major:
2283 {
2284   \cs_if_exist:NTF \tex_pdfvariable:D
2285     {
2286       \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2287       { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2288       { 1 }
2289     }
2290     {
2291       \cs_if_exist:NTF \tex_pdfmajorversion:D
2292       { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2293       { 1 }
2294     }
2295   }
2296 \cs_new:Npx \__pdf_backend_version_minor:
2297 {
2298   \exp_not:N \tex_the:D
2299   \cs_if_exist:NTF \tex_pdfminorversion:D
2300     { \exp_not:N \tex_pdfminorversion:D }
2301     { \tex_pdfvariable:D minorversion }
2302 }

```

(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>.

```

2303 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2304 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2305 \cs_new_protected:Npn \__pdf_backend_emc:
2306 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2307 </pdfmode>

```

6.4 dvipdfmx backend

```

2308 \*dvipdfmx | xdvipdfmx\
\__pdf_backend:n A generic function for the backend PDF specials: used where we can.
\__pdf_backend:x
2309 \cs_new_protected:Npx \__pdf_backend:n #1
2310 { \__kernel_backend_literal:n { pdf: #1 } }
2311 \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
2312 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2313 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2314 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2315 { \__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
2316 \int_new:N \g__pdf_backend_object_int
2317 \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
2318 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2319 {
2320   \int_gincr:N \g__pdf_backend_object_int
2321   \int_const:cn
2322     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2323     { \g__pdf_backend_object_int }
2324   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2325 }
2326 \cs_new:Npn \__pdf_backend_object_ref:n #1
2327 { @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
2328 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2329 {
2330   \exp_args:Nx \__pdf_backend_object_write:nnn
2331   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2332 }
2333 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2334 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2335 {
2336   \use:c { \__pdf_backend_object_write_ #1 :nn }
2337   { \__pdf_backend_object_ref:n {#2} } {#3}
2338 }

```

```

2339 \cs_new_protected:Npn \__pdf_backend_object_write_array: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_dict:nn #1#2
2345 {
2346   \__pdf_backend:x
2347   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2348 }
2349 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2350 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2351 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2352 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2353 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2354 {
2355   \__pdf_backend:x
2356   {
2357     #1 stream ~ #2 ~
2358     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2359   }
2360 }

```

(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
2361 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2362 {
2363   \int_gincr:N \g__pdf_backend_object_int
2364   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2365   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2366   {#2}
2367 }
2368 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2369 \cs_new:Npn \__pdf_backend_object_last:
2370 { @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.

```

2371 \bool_new:N \g__pdf_landscape_bool
2372 <*package>
2373 \AtBeginDocument
2374 {
2375   \cs_if_exist:NT \landscape
2376   {
2377     \tl_put_right:Nn \landscape

```

```

2378         { \bool_gset_true:N \g__pdf_landscape_bool }
2379     \tl_put_left:Nn \endlandscape
2380         { \bool_gset_false:N \g__pdf_landscape_bool }
2381     }
2382 }
2383 \</package>

```

(End definition for \g__pdf_landscape_bool.)

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

```

2384 \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.

```

2385 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2386 {
2387     \bool_if:NTF \g__pdf_landscape_bool
2388     {
2389         \box_move_up:nn {#2}
2390         {
2391             \vbox:n
2392             {
2393                 \__pdf_backend_annotation_aux:nnnn
2394                 { #2 + #3 } {#1} { Opt } {#4}
2395             }
2396         }
2397     }
2398     { \__pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4} }
2399 }
2400 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2401 {
2402     \int_gincr:N \g__pdf_backend_object_int
2403     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2404     \__pdf_backend:x
2405     {
2406         ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
2407         width ~ \dim_eval:n {#1} ~
2408         height ~ \dim_eval:n {#2} ~
2409         depth ~ \dim_eval:n {#3} ~
2410         <</Type/Annot #4 >>
2411     }
2412 }

```

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

__pdf_backend_annotation_last:

```

2413 \cs_new:Npn \__pdf_backend_annotation_last:
2414 { @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:
2415 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2416 { \__pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2417 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
2418 { \__pdf_backend_link_begin:n {#1#2} }
2419 \cs_new_protected:Npn \__pdf_backend_link_begin:n #1
2420 {
2421   \__pdf_backend:n
2422   {
2423     bann
2424     <<
2425     /Type /Annot
2426     #1
2427     >>
2428   }
2429 }
2430 \cs_new_protected:Npn \__pdf_backend_link_end:
2431 { \__pdf_backend:n { eann } }

```

All created using the same internals.

(End definition for __pdf_backend_link_begin_goto:nnw and others.)

```

\__pdf_backend_link_last: Data not available.
2432 \cs_new:Npn \__pdf_backend_link_last: { }

```

(End definition for __pdf_backend_link_last:.)

```

\__pdf_backend_link_margin:n Pass to dvipdfmx.
2433 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2434 { \__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 T_EX by using the backend data for @xpos and @ypos.

```

2435 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2436 {
2437   \__pdf_backend:x
2438   {
2439     dest ~ ( \exp_not:n {#1} )
2440     [
2441       @thispage
2442       \str_case:nnF {#2}
2443       {
2444         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2445         { fit } { /Fit }
2446         { fitb } { /FitB }
2447         { fitbh } { /FitBH }
2448         { fitbv } { /FitBV ~ @xpos }
2449         { fith } { /FitH ~ @ypos }
2450         { fitv } { /FitV ~ @xpos }
2451       }
2452       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2453     ]

```

```

2454     }
2455   }
2456   \cs_new_protected:Npn \__pdf_backend_destination_rectangle:nn #1#2
2457   {
2458     \group_begin:
2459     \hbox_set:Nn \l__pdf_internal_box {#2}
2460     \box_move_down:nn { \box_dp:N \l__pdf_internal_box }
2461     {
2462       \hbox:n
2463       {
2464         \__pdf_backend:n { obj ~ @pdf_ #1 _llx ~ @xpos }
2465         \__pdf_backend:n { obj ~ @pdf_ #1 _lly ~ @ypos }
2466       }
2467     }
2468     \box_use:N \l__pdf_internal_box
2469     \box_move_up:nn { \box_ht:N \l__pdf_internal_box }
2470     {
2471       \hbox:n
2472       {
2473         \__pdf_backend:n
2474         {
2475           dest ~ (#1)
2476           [
2477             @thispage
2478             /FitR ~
2479             @pdf_ #1 _llx ~ @pdf_ #1 _lly ~
2480             @xpos ~ @ypos
2481           ]
2482         }
2483       }
2484     }
2485     \group_end:
2486   }

```

(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
2487 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2488 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2489 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2490 {
2491   \bool_if:nF {#1}
2492   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2493 }

```

(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
2494 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2495 {
2496   \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
2497   \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }

```



```

2498 }
2499 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2500 {
2501   \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
2502   \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
2503 }

```

(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:
2504 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2505 \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:

```

2506 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2507 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2508 \cs_new_protected:Npn \__pdf_backend_emc:
2509 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2510 </dvipdfmx | xdvipdfmx>

```

6.5 dvisvgm backend

```

2511 <*dvisvgm>

```

6.5.1 Catalogue entries

__pdf_backend_catalog_gput:nn No-op.

__pdf_backend_info_gput:nn

```

2512 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2 { }
2513 \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:

```

2514 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2 { }
2515 \cs_new:Npn \__pdf_backend_object_ref:n #1 { }
2516 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2 { }
2517 \cs_new_protected:Npn \__pdf_backend_object_write:nx #1#2 { }
2518 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2 { }
2519 \cs_new_protected:Npn \__pdf_backend_object_now:nx #1#2 { }
2520 \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
2521 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2522 \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
2523 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2524 \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:
2525 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2526 \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:
2527 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2528 \cs_new_protected:Npn \__pdf_backend_emc: { }

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

2529 </dvisvgm>
2530 </initex | package>

```

7 l3backend-header Implementation

```

2531 <*dvips & header>

pdf.globaldict A small global dictionary for backend use.
2532 true setglobal
2533 /pdf.globaldict 4 dict def
2534 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
2535
2536 /pdf.cvs { 65534 string cvs } def
2537 /pdf.dvi.pt { 72.27 mul Resolution div } def
2538 /pdf.pt.dvi { 72.27 div Resolution mul } def
2539 /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
2540 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad
2541 /pdf.linkdp.pad { 0 } def
2542 /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 2543 /pdf.rect
pdf.save.linkur 2544 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.llx 2545 /pdf.save.ll
pdf.lly 2546 {
pdf.urx 2547     currentpoint
pdf.ury 2548     /pdf.lly exch def
2549     /pdf.llx exch def
2550 }
2551 def
2552 /pdf.save.ur
2553 {
2554     currentpoint
2555     /pdf.ury exch def
2556     /pdf.urx exch def
2557 }
2558 def
2559 /pdf.save.linkll
2560 {
2561     currentpoint
2562     pdf.linkmargin add
2563     pdf.linkdp.pad add
2564     /pdf.lly exch def
2565     pdf.linkmargin sub
2566     /pdf.llx exch def
2567 }
2568 def
2569 /pdf.save.linkur
2570 {
2571     currentpoint
2572     pdf.linkmargin sub
2573     pdf.linkht.pad sub
2574     /pdf.ury exch def
2575     pdf.linkmargin add
2576     /pdf.urx exch def
2577 }
2578 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 2579 /pdf.dest.anchor
pdf.dev.y 2580 {
pdf.tmpa 2581     currentpoint exch
pdf.tmpb 2582     pdf.dvi.pt 72 add
pdf.tmpc
pdf.tmpd
```

```

2583     /pdf.dest.x exch def
2584     pdf.dvi.pt
2585     vsize 72 sub exch sub
2586     /pdf.dest.y exch def
2587   }
2588   def
2589 /pdf.dest.point
2590   { pdf.dest.x pdf.dest.y } def
2591 /pdf.dest2device
2592   {
2593     /pdf.dest.y exch def
2594     /pdf.dest.x exch def
2595     matrix currentmatrix
2596     matrix defaultmatrix
2597     matrix invertmatrix
2598     matrix concatmatrix
2599     cvx exec
2600     /pdf.dev.y exch def
2601     /pdf.dev.x exch def
2602     /pdf.tmpd exch def
2603     /pdf.tmpc exch def
2604     /pdf.tmpb exch def
2605     /pdf.tmpa exch def
2606     pdf.dest.x pdf.tmpa mul
2607       pdf.dest.y pdf.tmpc mul add
2608       pdf.dev.x add
2609     pdf.dest.x pdf.tmpb mul
2610       pdf.dest.y pdf.tmpd mul add
2611       pdf.dev.y add
2612   }
2613   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	2614 /pdf.bordertracking false def
pdf.brokenlink.skip	2615 /pdf.bordertracking.begin
pdf.brokenlink.dict	2616 {
pdf.bordertracking.endpage	2617 SDict /pdf.bordertracking true put
pdf.bordertracking.continue	2618 SDict /pdf.leftboundary undef
pdf.originx	2619 SDict /pdf.rightboundary undef
pdf.originy	2620 /a where
	2621 {
	2622 /a
	2623 {
	2624 currentpoint pop
	2625 SDict /pdf.rightboundary known dup
	2626 {
	2627 SDict /pdf.rightboundary get 2 index lt
	2628 { not }
	2629 if

```

2630     }
2631     if
2632     { pop }
2633     { SDict exch /pdf.rightboundary exch put }
2634     ifelse
2635     moveto
2636     currentpoint pop
2637     SDict /pdf.leftboundary known dup
2638     {
2639         SDict /pdf.leftboundary get 2 index gt
2640         { not }
2641         if
2642     }
2643     if
2644     { pop }
2645     { SDict exch /pdf.leftboundary exch put }
2646     ifelse
2647 }
2648 put
2649 }
2650 if
2651 }
2652 def
2653 /pdf.bordertracking.end
2654 {
2655     /a where { /a { moveto } put } if
2656     /x where { /x { 0 exch rmoveto } put } if
2657     SDict /pdf.leftboundary known
2658     { pdf.outerbox 0 pdf.leftboundary put }
2659     if
2660     SDict /pdf.rightboundary known
2661     { pdf.outerbox 2 pdf.rightboundary put }
2662     if
2663     SDict /pdf.bordertracking false put
2664 }
2665 def
2666 /pdf.bordertracking.endpage
2667 {
2668     pdf.bordertracking
2669     {
2670         pdf.bordertracking.end
2671         true setglobal
2672         pdf.globaldict
2673         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
2674         pdf.globaldict
2675         /pdf.brokenlink.skip pdf.baselineskip put
2676         pdf.globaldict
2677         /pdf.brokenlink.dict
2678         pdf.link.dict pdf.cvs put
2679         false setglobal
2680         mark pdf.link.dict cvx exec /Rect
2681         [
2682             pdf.llx
2683             pdf.lly

```

```

2684         pdf.outerbox 2 get pdf.linkmargin add
2685         currentpoint exch pop
2686         pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
2687     ]
2688     /ANN pdf.pdfmark
2689 }
2690 if
2691 }
2692 def
2693 /pdf.bordertracking.continue
2694 {
2695     /pdf.link.dict pdf.globaldict
2696     /pdf.brokenlink.dict get def
2697     /pdf.outerbox pdf.globaldict
2698     /pdf.brokenlink.rect get def
2699     /pdf.baselineskip pdf.globaldict
2700     /pdf.brokenlink.skip get def
2701     pdf.globaldict dup dup
2702     /pdf.brokenlink.dict undef
2703     /pdf.brokenlink.skip undef
2704     /pdf.brokenlink.rect undef
2705     currentpoint
2706     /pdf.originy exch def
2707     /pdf.originx exch def
2708     /a where
2709     {
2710         /a
2711         {
2712             moveto
2713             SDict
2714             begin
2715                 currentpoint pdf.originy ne exch
2716                 pdf.originx ne or
2717                 {
2718                     pdf.save.linkll
2719                     /pdf.lly
2720                     pdf.lly pdf.outerbox 1 get sub def
2721                     pdf.bordertracking.begin
2722                 }
2723                 if
2724                 end
2725             }
2726             put
2727         }
2728     if
2729     /x where
2730     {
2731         /x
2732         {
2733             0 exch rmoveto
2734             SDict~
2735             begin
2736                 currentpoint
2737                 pdf.originy ne exch pdf.originx ne or

```

```

2738         {
2739             pdf.save.linkll
2740             /pdf.lly
2741             pdf.lly pdf.outerbox 1 get sub def
2742             pdf.bordertracking.begin
2743         }
2744         if
2745         end
2746     }
2747     put
2748 }
2749 if
2750 }
2751 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>
---	--

```

2752 /pdf.breaklink
2753 {
2754     pop
2755     counttomark 2 mod 0 eq
2756     {
2757         counttomark /pdf.count exch def
2758         {
2759             pdf.count 0 eq { exit } if
2760             counttomark 2 roll
2761             1 index /Rect eq
2762             {
2763                 dup 4 array copy
2764                 dup dup
2765                 1 get
2766                 pdf.outerbox pdf.rect.ht
2767                 pdf.linkmargin 2 mul add sub
2768                 3 exch put
2769                 dup
2770                 pdf.outerbox 2 get
2771                 pdf.linkmargin add
2772                 2 exch put
2773                 dup dup
2774                 3 get
2775                 pdf.outerbox pdf.rect.ht
2776                 pdf.linkmargin 2 mul add add
2777                 1 exch put
2778             } /pdf.currentrect exch def
2779             pdf.breaklink.write
2780             {
2781                 pdf.currentrect
2782                 dup
2783                 pdf.outerbox 0 get

```

```

2784         pdf.linkmargin sub
2785         0 exch put
2786     dup
2787         pdf.outerbox 2 get
2788         pdf.linkmargin add
2789         2 exch put
2790     dup dup
2791         1 get
2792         pdf.baselineskip add
2793         1 exch put
2794     dup dup
2795         3 get
2796         pdf.baselineskip add
2797         3 exch put
2798     /pdf.currentrect exch def
2799     pdf.breaklink.write
2800 }
2801 1 index 3 get
2802 pdf.linkmargin 2 mul add
2803 pdf.outerbox pdf.rect.ht add
2804 2 index 1 get sub
2805 pdf.baselineskip div round cvi 1 sub
2806 exch
2807 repeat
2808 pdf.currentrect
2809 dup
2810     pdf.outerbox 0 get
2811     pdf.linkmargin sub
2812     0 exch put
2813     dup dup
2814         1 get
2815         pdf.baselineskip add
2816         1 exch put
2817     dup dup
2818         3 get
2819         pdf.baselineskip add
2820         3 exch put
2821     dup 2 index 2 get 2 exch put
2822     /pdf.currentrect exch def
2823     pdf.breaklink.write
2824     SDict /pdf.pdfmark.good false put
2825     exit
2826 }
2827 { pdf.count 2 sub /pdf.count exch def }
2828 ifelse
2829 }
2830 loop
2831 }
2832 if
2833 /ANN
2834 }
2835 def
2836 /pdf.breaklink.write
2837 {

```



```

2838     counttomark 1 sub
2839     index /_objdef eq
2840     {
2841         counttomark -2 roll
2842         dup wcheck
2843         {
2844             readonly
2845             counttomark 2 roll
2846         }
2847         { pop pop }
2848         ifelse
2849     }
2850     if
2851     counttomark 1 add copy
2852     pop pdf.currentrect
2853     /ANN pdfmark
2854 }
2855 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.

```

2856 /pdf.pdfmark
2857 {
2858     SDict /pdf.pdfmark.good true put
2859     dup /ANN eq
2860     {
2861         pdf.pdfmark.store
2862         pdf.pdfmark.dict
2863         begin
2864             Subtype /Link eq
2865             currentdict /Rect known and
2866             SDict /pdf.outerbox known and
2867             SDict /pdf.baselineskip known and
2868             {
2869                 Rect 3 get
2870                 pdf.linkmargin 2 mul add
2871                 pdf.outerbox pdf.rect.ht add
2872                 Rect 1 get sub
2873                 pdf.baselineskip div round cvi 0 gt
2874                 { pdf.breaklink }
2875                 if
2876             }
2877             if
2878             end
2879             SDict /pdf.outerbox undef
2880             SDict /pdf.baselineskip undef
2881             currentdict /pdf.pdfmark.dict undef
2882         }
2883     if

```

```

2884     pdf.pdfmark.good
2885     { pdfmark }
2886     { cleartomark }
2887     ifelse
2888   }
2889   def
2890 /pdf.pdfmark.store
2891 {
2892   /pdf.pdfmark.dict 65534 dict def
2893   counttomark 1 add copy
2894   pop
2895   {
2896     dup mark eq
2897     {
2898       pop
2899       exit
2900     }
2901     {
2902       pdf.pdfmark.dict
2903       begin def end
2904     }
2905     ifelse
2906   }
2907   loop
2908 }
2909 def

```

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

```

2910 </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	
<code>\AtBeginDocument</code>	
.	<u>374</u> , <u>431</u> , <u>1316</u> , <u>1451</u> , <u>1615</u> , <u>2373</u>
<code>\AtBeginDvi</code>	<u>36</u> , <u>37</u>
B	
<code>\begin</code>	<u>1365</u> , <u>1370</u>
bool commands:	
<code>\bool_gset_false:N</code>	
.	<u>577</u> , <u>593</u> , <u>619</u> , <u>641</u> ,
.	<u>657</u> , <u>809</u> , <u>1135</u> , <u>1171</u> , <u>1820</u> , <u>1871</u> , <u>2380</u>
<code>\bool_gset_true:N</code>	
.	<u>575</u> , <u>644</u> , <u>807</u> , <u>1150</u> , <u>1813</u> , <u>1819</u> , <u>2378</u>
<code>\bool_if:NTF</code>	<u>584</u> , <u>588</u> , <u>606</u> , <u>610</u> , <u>614</u> ,
.	<u>627</u> , <u>632</u> , <u>636</u> , <u>648</u> , <u>652</u> , <u>820</u> , <u>825</u> ,
.	<u>830</u> , <u>1109</u> , <u>1154</u> , <u>1341</u> , <u>1382</u> , <u>1498</u> ,
.	<u>1540</u> , <u>1808</u> , <u>1823</u> , <u>1828</u> , <u>1833</u> , <u>2387</u>
<code>\bool_if:nTF</code>	<u>2244</u> , <u>2491</u>
<code>\bool_lazy_and:nnTF</code>	<u>40</u>
<code>\bool_lazy_or:nnTF</code>	<u>1374</u> , <u>1533</u>
<code>\bool_new:N</code>	
.	<u>578</u> , <u>645</u> , <u>810</u> , <u>1151</u> , <u>1796</u> , <u>1797</u> , <u>2371</u>
<code>\bool_set_false:N</code>	
.	<u>1351</u> , <u>1465</u> , <u>1558</u> , <u>1628</u>
box commands:	
<code>\box_dp:N</code>	<u>137</u> , <u>139</u> , <u>187</u> , <u>189</u> ,
.	<u>244</u> , <u>246</u> , <u>293</u> , <u>295</u> , <u>297</u> , <u>299</u> , <u>1850</u> ,
.	<u>1883</u> , <u>1884</u> , <u>1909</u> , <u>2011</u> , <u>2123</u> , <u>2460</u>
<code>\box_ht:N</code>	<u>139</u> , <u>189</u> ,
.	<u>246</u> , <u>297</u> , <u>299</u> , <u>1394</u> , <u>1595</u> , <u>1855</u> ,
.	<u>1894</u> , <u>1895</u> , <u>1916</u> , <u>2015</u> , <u>2122</u> , <u>2469</u>
<code>\box_if_empty:NTF</code>	<u>1950</u>
<code>\box_move_down:nn</code>	
.	<u>1771</u> , <u>1850</u> , <u>2010</u> , <u>2460</u>
<code>\box_move_up:nn</code>	
.	<u>1774</u> , <u>1855</u> , <u>2014</u> , <u>2389</u> , <u>2469</u>
<code>\box_new:N</code>	<u>1663</u> , <u>1752</u> , <u>1753</u>
<code>\box_set_dp:Nn</code>	<u>1302</u>
<code>\box_set_ht:Nn</code>	<u>1301</u>
<code>\box_set_wd:Nn</code>	<u>201</u> , <u>1300</u>
<code>\box_use:N</code>	<u>144</u> , <u>162</u> , <u>176</u> , <u>192</u> , <u>219</u> ,
.	<u>233</u> , <u>249</u> , <u>265</u> , <u>277</u> , <u>328</u> , <u>345</u> , <u>364</u> ,
.	<u>760</u> , <u>1017</u> , <u>1303</u> , <u>1801</u> , <u>2013</u> , <u>2124</u> , <u>2468</u>
<code>\box_wd:N</code>	<u>138</u> , <u>146</u> , <u>188</u> , <u>194</u> , <u>245</u> ,
.	<u>251</u> , <u>294</u> , <u>296</u> , <u>332</u> , <u>1393</u> , <u>1594</u> , <u>2121</u>
box internal commands:	
<code>__box_backend_clip:N</code>	
.	<u>126</u> , <u>181</u> , <u>238</u> , <u>282</u>
<code>\l__box_backend_cos_fp</code>	<u>196</u>
<code>__box_backend_rotate:Nn</code>	
.	<u>148</u> , <u>196</u> , <u>253</u> , <u>335</u>
<code>__box_backend_rotate_aux:Nn</code>	
.	<u>148</u> , <u>196</u> , <u>253</u>
<code>__box_backend_scale:Nnn</code>	
.	<u>165</u> , <u>224</u> , <u>268</u> , <u>348</u>
<code>\l__box_backend_sin_fp</code>	<u>196</u>
<code>\g__box_clip_path_int</code>	<u>282</u>
C	
clist commands:	
<code>\clist_map_function:nN</code>	<u>665</u> , <u>840</u>
<code>\clist_map_function:nn</code>	<u>1178</u>
color internal commands:	
<code>__color_backend_cmyk:nnnn</code>	<u>398</u> , <u>467</u>
<code>__color_backend_cmyk_aux:nnnn</code>	<u>467</u>
<code>__color_backend_gray:n</code>	<u>398</u> , <u>467</u>
<code>__color_backend_gray_aux:n</code>	<u>467</u>
<code>__color_backend_pickup:N</code>	<u>372</u> , <u>429</u>
<code>__color_backend_pickup:w</code>	<u>13</u> , <u>372</u> , <u>429</u>
<code>__color_backend_reset:</code>	<u>398</u> , <u>467</u>
<code>__color_backend_rgb:nnn</code>	<u>398</u> , <u>467</u>
<code>__color_backend_rgb_aux:nnn</code>	<u>467</u>
<code>__color_backend_select:n</code>	<u>398</u> , <u>467</u>
<code>__color_backend_spot:nn</code>	<u>398</u> , <u>467</u>
color.fc	<u>398</u> , <u>524</u>
cs commands:	
<code>\cs_generate_variant:Nn</code>	<u>28</u> ,
.	<u>32</u> , <u>35</u> , <u>69</u> , <u>97</u> , <u>102</u> , <u>113</u> , <u>120</u> , <u>424</u> ,
.	<u>509</u> , <u>523</u> , <u>728</u> , <u>734</u> , <u>770</u> , <u>918</u> , <u>1026</u> ,
.	<u>1057</u> , <u>1512</u> , <u>1569</u> , <u>1585</u> , <u>1667</u> , <u>1704</u> ,
.	<u>1749</u> , <u>2194</u> , <u>2225</u> , <u>2311</u> , <u>2333</u> , <u>2368</u>
<code>\cs_gset:Npx</code>	<u>2496</u> , <u>2501</u>
<code>\cs_if_exist:NTF</code>	<u>36</u> , <u>64</u> , <u>72</u> , <u>80</u> ,
.	<u>86</u> , <u>92</u> , <u>376</u> , <u>433</u> , <u>503</u> , <u>512</u> , <u>901</u> , <u>909</u> ,
.	<u>1946</u> , <u>2044</u> , <u>2055</u> , <u>2066</u> , <u>2074</u> , <u>2081</u> ,
.	<u>2088</u> , <u>2095</u> , <u>2116</u> , <u>2129</u> , <u>2136</u> , <u>2154</u> ,
.	<u>2190</u> , <u>2221</u> , <u>2229</u> , <u>2237</u> , <u>2251</u> , <u>2258</u> ,
.	<u>2267</u> , <u>2277</u> , <u>2284</u> , <u>2291</u> , <u>2299</u> , <u>2375</u>
<code>\cs_if_exist_p:N</code>	<u>41</u>
<code>\cs_new:Npn</code>	<u>670</u> ,
.	<u>845</u> , <u>1182</u> , <u>1598</u> , <u>1607</u> , <u>1657</u> , <u>1682</u> ,
.	<u>1750</u> , <u>1791</u> , <u>1972</u> , <u>2034</u> , <u>2035</u> , <u>2162</u> ,
.	<u>2195</u> , <u>2196</u> , <u>2326</u> , <u>2369</u> , <u>2413</u> , <u>2432</u> ,
.	<u>2504</u> , <u>2505</u> , <u>2515</u> , <u>2520</u> , <u>2525</u> , <u>2526</u>
<code>\cs_new:Npx</code>	<u>2052</u> , <u>2078</u> , <u>2226</u> , <u>2282</u> , <u>2296</u>

`\cs_new_eq:NN`
 . 25, 522, 769, 775, 776, 916, 1025,
 1318, 1347, 1404, 1405, 1453, 1461,
 1483, 1554, 1617, 1624, 1656, 1801
`\cs_new_protected:Npn`
 ... 26, 30, 33, 45, 51, 56, 58, 100,
 103, 105, 107, 111, 114, 116, 118,
 126, 148, 150, 165, 181, 196, 198,
 224, 238, 253, 255, 268, 282, 335,
 348, 373, 393, 398, 407, 409, 414,
 416, 425, 430, 440, 467, 478, 483,
 485, 487, 497, 499, 524, 530, 535,
 537, 539, 547, 555, 564, 574, 576,
 579, 581, 595, 600, 621, 643, 646,
 659, 672, 677, 679, 681, 683, 685,
 687, 689, 691, 700, 709, 711, 713,
 718, 723, 729, 735, 747, 771, 773,
 777, 782, 787, 797, 806, 808, 811,
 813, 815, 817, 822, 827, 832, 834,
 847, 852, 854, 856, 858, 860, 862,
 864, 866, 875, 884, 886, 888, 893,
 919, 934, 959, 971, 983, 995, 1002,
 1027, 1032, 1034, 1042, 1052, 1060,
 1065, 1070, 1081, 1091, 1101, 1103,
 1105, 1107, 1138, 1140, 1145, 1147,
 1149, 1152, 1173, 1184, 1197, 1199,
 1201, 1203, 1205, 1207, 1209, 1211,
 1213, 1223, 1232, 1240, 1242, 1244,
 1254, 1269, 1274, 1289, 1319, 1333,
 1348, 1360, 1371, 1399, 1411, 1424,
 1434, 1455, 1462, 1470, 1481, 1485,
 1488, 1503, 1513, 1548, 1555, 1561,
 1567, 1570, 1577, 1586, 1591, 1599,
 1618, 1625, 1631, 1633, 1635, 1646,
 1665, 1668, 1670, 1674, 1684, 1705,
 1710, 1715, 1720, 1729, 1755, 1769,
 1800, 1802, 1804, 1806, 1811, 1826,
 1831, 1873, 1902, 1926, 1935, 1974,
 1981, 2006, 2030, 2031, 2032, 2033,
 2036, 2038, 2060, 2062, 2167, 2200,
 2242, 2303, 2305, 2312, 2314, 2318,
 2328, 2334, 2339, 2344, 2349, 2351,
 2353, 2361, 2385, 2400, 2415, 2417,
 2419, 2430, 2433, 2435, 2456, 2487,
 2489, 2494, 2499, 2506, 2508, 2512,
 2513, 2514, 2516, 2517, 2518, 2519,
 2521, 2522, 2523, 2524, 2527, 2528
`\cs_new_protected:Npx` 62, 70, 78, 84,
 90, 501, 510, 899, 907, 2042, 2064,
 2072, 2086, 2093, 2112, 2127, 2134,
 2145, 2234, 2248, 2256, 2274, 2309
`\cs_set_eq:NN` 1967, 1968
`\cs_set_protected:Npn`
 378, 435, 2143, 2165, 2198

D

dim commands:

`\dim_eval:n` 1778, 2047, 2048,
 2049, 2091, 2407, 2408, 2409, 2434
`\dim_max:nn` 1881, 1892
`\dim_set:Nn` ... 1393, 1394, 1594, 1595
`\dim_to_decimal:n` 293, 294, 295, 296,
 297, 299, 1063, 1068, 1074, 1075,
 1076, 1077, 1086, 1087, 1088, 1179,
 1198, 1651, 1652, 1879, 1890, 1908,
 1909, 1911, 1914, 1916, 1920, 1978
`\dim_to_decimal_in_bp:n` . 137, 138,
 139, 187, 188, 189, 244, 245, 246,
 543, 544, 551, 552, 559, 560, 568,
 569, 570, 667, 671, 675, 780, 785,
 791, 792, 793, 801, 802, 842, 846,
 850, 1183, 1324, 1325, 1326, 1327,
 1475, 1476, 1477, 1478, 1527, 1528,
 1529, 1530, 1640, 1641, 1642, 1643

draw internal commands:

`__draw_align_currentpoint...` .. 21
`__draw_backend_add_to_path:n` ...
 1060, 1106
`__draw_backend_begin:` 524, 771, 1027
`__draw_backend_box_use:Nnnnn` ...
 16, 747, 1002, 1289
`__draw_backend_cap_but:`
 659, 834, 1173
`__draw_backend_cap_rectangle:` ..
 659, 834, 1173
`__draw_backend_cap_round:`
 659, 834, 1173
`__draw_backend_clip:` 579, 811, 1105
`__draw_backend_closepath:`
 579, 811, 1105
`__draw_backend_closestroke:` ...
 579, 811, 1105
`__draw_backend_cm:nnnn` 735,
 755, 756, 757, 919, 1006, 1274, 1292
`__draw_backend_cm_aux:nnnn` ... 919
`__draw_backend_cm_decompose:nnnnN`
 929, 958
`__draw_backend_cm_decompose_-`
 auxi:nnnnN 958
`__draw_backend_cm_decompose_-`
 auxii:nnnnN 958
`__draw_backend_cm_decompose_-`
 auxiii:nnnnN 958
`__draw_backend_color_fill:n` .. 691
`__draw_backend_color_fill:nnn` 1213
`__draw_backend_color_fill_-`
 cmyk:nnnn 691, 866, 1213
`__draw_backend_color_fill_-`
 gray:n 691, 866, 1213

_draw_backend_color_fill_- rgb:nnn	691, 866, 1213
_draw_backend_color_gray_aux:n	1236, 1240
_draw_backend_color_reset: . .	866
_draw_backend_color_select:n .	866
_draw_backend_color_stroke:n .	691
_draw_backend_color_stroke_- cmyk:nnnn	691, 866, 1213
_draw_backend_color_stroke_- gray:n	691, 866, 1213
_draw_backend_color_stroke_- rgb:nnn	691, 866, 1213
_draw_backend_curveto:nnnnnn . .	539, 777, 1060
_draw_backend_dash:n 659, 834, 1173	
_draw_backend_dash_aux:nn . .	1173
_draw_backend_dash_pattern:nn .	659, 834, 1173
_draw_backend_discardpath: . . .	579, 811, 1105
_draw_backend_end: . 524, 771, 1027	
_draw_backend_evenodd_rule: . . .	574, 806, 1101
_draw_backend_fill: 579, 811, 1105	
_draw_backend_fillstroke:	579, 811, 1105
_draw_backend_join_bevel:	659, 834, 1173
_draw_backend_join_miter:	659, 834, 1173
_draw_backend_join_round:	659, 834, 1173
_draw_backend_lineto:nn	539, 777, 1060
_draw_backend_linewidth:n	659, 834, 1173
_draw_backend_literal:n 522, 527,	528, 532, 536, 538, 541, 549, 557,
	566, 580, 583, 586, 592, 602, 603,
	604, 609, 612, 618, 623, 624, 625,
	630, 631, 634, 640, 650, 656, 661,
	674, 678, 680, 682, 684, 686, 688,
	690, 737, 749, 750, 751, 752, 753,
	754, 758, 759, 761, 762, 763, 764,
	765, 769, 779, 784, 789, 799, 812,
	814, 816, 819, 824, 829, 833, 836,
	849, 853, 855, 857, 859, 861, 863,
	865, 1025, 1046, 1054, 1112, 1131, 1157
_draw_backend_miterlimit:n . . .	659, 834, 1173
_draw_backend_moveto:nn	539, 777, 1060
_draw_backend_nonzero_rule: . . .	574, 806, 1101
_draw_backend_path:n	1105
_draw_backend_rectangle:nnnn . .	539, 777, 1060
_draw_backend_scope:n	1030, 1034, 1102, 1104, 1124,
	1164, 1186, 1198, 1200, 1202, 1204,
	1206, 1208, 1210, 1212, 1256, 1276
_draw_backend_scope_begin: . . .	535, 772, 775, 1029, 1034
_draw_backend_scope_end:	535, 774, 775, 1033, 1034
_draw_backend_select:n	1225, 1243, 1271
_draw_backend_stroke: 579, 811, 1105	
\g_draw_clip_path_int	1111, 1114, 1127, 1156, 1159, 1167
_draw_color_reset:	732
\g_draw_draw_clip_bool . . .	579, 1105
\g_draw_draw_eor_bool	574, 588, 606,
	614, 627, 636, 652, 806, 820, 825, 830
\g_draw_draw_path_int	1105
\g_draw_draw_path_tl	1060, 1116, 1132, 1134, 1161, 1170
\g_draw_draw_scope_int	1034
\l_draw_draw_scope_int	1034
\g_draw_path_int	1120, 1137
E	
\endlandscape	2379
\evensidemargin	1847
exp commands:	
\exp_after:wN	385, 1605
\exp_args:Nf	664, 839
\exp_args:NNf	149, 197, 254
\exp_args:Nnx	1746, 2364
\exp_args:NV	380
\exp_args:Nx	484, 1417, 1438, 1717, 1839, 2330
\exp_last_unbraced:Nx	389, 437
\exp_not:N	67,
	76, 95, 506, 507, 515, 904, 905,
	912, 2047, 2048, 2049, 2054, 2056,
	2057, 2080, 2082, 2083, 2089, 2090,
	2091, 2096, 2097, 2099, 2109, 2117,
	2118, 2121, 2122, 2123, 2149, 2151,
	2158, 2228, 2230, 2231, 2236, 2240,
	2250, 2262, 2263, 2269, 2270, 2276,
	2278, 2280, 2287, 2292, 2298, 2300
\exp_not:n	27, 67, 76, 95, 1708,
	1713, 2002, 2175, 2176, 2195, 2196,
	2206, 2207, 2342, 2347, 2358, 2439

F

file commands:

\file_compare_timestamp:nNnTF . 1426
 \file_parse_full_name:nNNN 1413, 1436

fp commands:

\fp_compare:nNnTF
 156, 203, 209, 261, 939, 952, 997
 \fp_eval:n 149, 158, 171,
 172, 197, 214, 229, 231, 254, 263,
 274, 275, 342, 357, 358, 403, 404,
 408, 412, 472, 473, 474, 475, 484,
 492, 493, 494, 678, 695, 696, 705,
 706, 710, 712, 716, 721, 740, 741,
 853, 870, 871, 879, 880, 885, 887,
 891, 896, 924, 925, 941, 946, 947,
 954, 964, 965, 966, 967, 976, 977,
 978, 979, 988, 989, 990, 991, 1012,
 1013, 1200, 1218, 1219, 1220, 1228,
 1229, 1237, 1243, 1249, 1250, 1251,
 1272, 1282, 1283, 1999, 2109, 2452
 \fp_new:N 222, 223
 \fp_set:Nn 202, 205
 \fp_use:N 208, 212, 217
 \fp_zero:N 204
 \c_zero_fp . 156, 203, 209, 261, 939, 952

G

galley commands:

\l_galley_text_width_dim 1911
 \l_galley_total_left_margin_dim 1842

graphics commands:

\graphics_bb_restore:nTF . 1362, 1588
 \graphics_bb_save:n 1397, 1596
 \l_graphics_decodearray_tl
 1339, 1340,
 1350, 1376, 1380, 1381, 1464, 1496,
 1497, 1535, 1538, 1539, 1557, 1627
 \graphics_extract_bb:n
 1459, 1466, 1622, 1629
 \l_graphics_interpolate_bool ...
 1341, 1351, 1375, 1382,
 1465, 1498, 1534, 1540, 1558, 1628
 \l_graphics_llx_dim
 1324, 1475, 1527, 1640
 \l_graphics_lly_dim
 1325, 1476, 1528, 1641
 \l_graphics_name_tl 1431
 \l_graphics_page_int
 1335, 1355, 1356, 1386,
 1387, 1457, 1494, 1495, 1521, 1522,
 1550, 1563, 1564, 1603, 1604, 1620
 \l_graphics_pagebox_tl
 41, 1336, 1354,

1388, 1389, 1458, 1492, 1493, 1523,
 1525, 1551, 1572, 1573, 1605, 1621
 \graphics_read_bb:n . 1318, 1453, 1617
 \l_graphics_urx_dim
 .. 1326, 1393, 1477, 1529, 1594, 1642
 \l_graphics_ury_dim .. 1327, 1394,
 1478, 1530, 1595, 1643, 1651, 1652

graphics internal commands:

\l__graphics_backend_dir_str . 1406
 \l__graphics_backend_ext_str . 1406
 __graphics_backend_getbb_auxi:n
 1333
 __graphics_backend_getbb_-
 auxi:nN 1548
 __graphics_backend_getbb_-
 auxii:n 1333
 __graphics_backend_getbb_-
 auxii:nnN 1548
 __graphics_backend_getbb_-
 auxiii:nNnn 1548
 __graphics_backend_getbb_-
 auxiv:nnNnn 1548
 __graphics_backend_getbb_-
 auxv:nNnn 1548
 __graphics_backend_getbb_-
 auxvi:nNnn 1589, 1591
 __graphics_backend_getbb_eps:n .
 1312, 1406, 1447, 1611
 __graphics_backend_getbb_eps:nm
 1406
 __graphics_backend_getbb_eps:nn
 1417, 1424
 __graphics_backend_getbb_jpg:n .
 1333, 1447, 1548, 1618
 __graphics_backend_getbb_-
 pagebox:w 1548, 1605
 __graphics_backend_getbb_pdf:n .
 1333, 1432, 1447, 1548, 1625
 __graphics_backend_getbb_png:n .
 1333, 1447, 1548, 1618
 __graphics_backend_include:nn 1631
 __graphics_backend_include_-
 auxi:nn 1470
 __graphics_backend_include_-
 auxii:nnn 1470
 __graphics_backend_include_-
 auxiii:nnn 1470
 __graphics_backend_include_-
 bitmap_quote:w 1599, 1646
 __graphics_backend_include_-
 eps:n 1319, 1406, 1470, 1631
 __graphics_backend_include_-
 jpg:n 1399, 1470, 1646

<code>__graphics_backend_include-</code>	<code>\int_set_eq:NN</code>	1036, 1940
<code>pdf:n</code>	<code>\int_use:N</code>	286,
<code>__graphics_backend_include_pdf-</code>		317, 1114, 1120, 1127, 1159, 1167,
<code>quote:w</code>		1356, 1387, 1402, 1495, 1508, 1520,
<code>__graphics_backend_include-</code>		1522, 1604, 1683, 1734, 1747, 1751,
<code>png:n</code>		1763, 1786, 1792, 1865, 1973, 2163,
<code>\l__graphics_backend_name_str</code>		2170, 2327, 2365, 2370, 2406, 2414
<code>\l__graphics_graphics_attr_tl</code>	<code>\int_value:w</code>	2054, 2080, 2228, 2240
	<code>\int_zero:N</code>	1335, 1457, 1550, 1620
<code>\l__graphics_internal_box</code>		K
		kernel internal commands:
<code>\g__graphics_track_int</code>	<code>__kernel_backend_align_begin:</code>	45, 129, 153, 168
	<code>__kernel_backend_align_end:</code>	45, 143, 161, 175
group commands:	<code>\g__kernel_backend_header_bool</code>	41, 42
<code>\group_begin:</code>	<code>__kernel_backend_literal:n</code>	25, 31,
2008, 2114, 2142, 2164, 2197, 2458		34, 43, 47, 54, 57, 59, 101, 104, 106,
<code>\group_end:</code>		108, 112, 258, 271, 418, 426, 526,
2028, 2125, 2161, 2193, 2224, 2485		533, 731, 936, 943, 949, 1009, 1019,
<code>\group_insert_after:N</code>		1321, 1472, 1507, 1517, 1637, 1648,
		2310, 2434, 2488, 2492, 2497, 2502
		<code>__kernel_backend_literal_page:n</code>
		70, 103, 2304, 2306, 2507, 2509
		<code>__kernel_backend_literal_pdf:n</code>
		62, 100, 184, 241, 769, 916
		<code>__kernel_backend_literal-</code>
		<code>postscript:n</code>
		130, 131, 133, 134, 142, 154, 169, 522
		<code>__kernel_backend_literal_svg:n</code>
		111, 115, 117,
		119, 285, 287, 304, 1025, 1293, 1304
		<code>__kernel_backend_matrix:n</code>
		90, 206, 227, 922
		<code>__kernel_backend_postscript:n</code>
		33, 420,
		725, 1666, 1722, 1772, 1779, 1814,
		1851, 1858, 1862, 1876, 1904, 1954,
		1961, 1967, 1976, 1983, 2012, 2016
		<code>__kernel_backend_scope_begin:</code>
		56, 78, 105, 114, 128, 152, 167, 183,
		200, 226, 240, 257, 270, 775, 1004, 1291
		<code>__kernel_backend_scope_begin:n</code>
		118, 306, 314, 319, 337, 350
		<code>__kernel_backend_scope_end:</code>
		56, 78, 105, 114, 145, 163,
		177, 193, 220, 234, 250, 266, 278,
		329, 330, 331, 346, 365, 776, 1021, 1305
		<code>\l__kernel_color_stack_int</code>
		466, 506, 515, 904, 912

H	
hbox commands:	
<code>\hbox:n</code>	1772, 1773, 1776,
	1851, 1857, 2012, 2016, 2462, 2471
<code>\hbox_overlap_right:n</code>	144,
	176, 192, 233, 249, 277, 364, 760, 1017
<code>\hbox_set:Nn</code>	1391,
	1593, 1838, 1875, 2009, 2115, 2459
<code>\hbox_set:Nw</code>	1821
<code>\hbox_set_end:</code>	1836
<code>\hbox_unpack:N</code>	1968

I	
int commands:	
<code>\int_compare:nNnTF</code>	1355, 1386, 1494,
	1521, 1563, 1603, 1939, 2260, 2286
<code>\int_const:Nn</code>	
	1395, 1516, 1677, 2148, 2321
<code>\int_eval:n</code>	2240,
	2263, 2270, 2280, 2488, 2496, 2501
<code>\int_gincr:N</code>	284,
	1055, 1111, 1156, 1515, 1676, 1731,
	1758, 1782, 1860, 2320, 2363, 2402
<code>\int_gset:Nn</code>	1928
<code>\int_gset_eq:NN</code>	
	1048, 1759, 1783, 1861, 2403
<code>\int_gzero:N</code>	1040
<code>\int_if_exist:NTF</code>	1505
<code>\int_if_odd:nTF</code>	1845
<code>\int_new:N</code>	
	334, 466, 1058, 1059, 1137, 1469,
	1672, 1754, 1793, 1795, 2316, 2384

L
`\landscape` 2375, 2377

M
math commands:
`\c_math_toggle_token` 1824, 1834
mode commands:
`\mode_if_horizontal:TF` ... 1930, 1937
`\mode_if_math:TF` 1818

O
`\oddsidemargin` 1846

P
pdf internal commands:
`__pdf_backend:n` 2309,
2313, 2315, 2341, 2346, 2355, 2404,
2421, 2431, 2437, 2464, 2465, 2473
`__pdf_backend_annotation:nnnn` ..
..... 1755, 2042, 2385
`__pdf_backend_annotation_-`
`aux:nnnn` 1755, 2385
`\g__pdf_backend_annotation_int` ..
..... 1754,
1759, 1783, 1792, 2384, 2403, 2414
`__pdf_backend_annotation_last:` ..
..... 1791, 2052, 2413
`__pdf_backend_bdc:nn`
..... 2036, 2303, 2506, 2527
`__pdf_backend_catalog_gput:nn` ..
..... 1668, 2127, 2312, 2512
`__pdf_backend_compress_objects:n`
..... 2030, 2234, 2487, 2521
`__pdf_backend_compresslevel:n` ..
..... 2030, 2234, 2487, 2521
`\l__pdf_backend_content_box` 1752,
1821, 1850, 1853, 1855, 1884, 1895
`__pdf_backend_destination:nn` ...
..... 1981, 2093, 2435
`__pdf_backend_destination_-`
`rectangle:nn` 1981, 2093, 2435
`__pdf_backend_emc:`
..... 2036, 2303, 2506, 2527
`__pdf_backend_info_gput:nn`
..... 1668, 2127, 2312, 2512
`__pdf_backend_link:nw` 1802
`__pdf_backend_link_aux:nw` ... 1802
`__pdf_backend_link_begin:n` .. 2415
`__pdf_backend_link_begin:nnnw` 2060
`__pdf_backend_link_begin:nw` ...
..... 1803, 1805, 1806
`__pdf_backend_link_begin_aux:nw`
..... 1809, 1811
`__pdf_backend_link_begin_-`
`goto:nnw` 1802, 2060, 2415

`__pdf_backend_link_begin_-`
`user:nnw` 1802, 2060, 2415
`\g__pdf_backend_link_bool`
..... 1797, 1808, 1813, 1828, 1871
`\g__pdf_backend_link_dict_tl` ...
..... 1794, 1816, 1866
`__pdf_backend_link_end:`
..... 1802, 2060, 2415
`__pdf_backend_link_end_aux:` . 1802
`\g__pdf_backend_link_int`
..... 1793, 1861, 1865, 1973
`__pdf_backend_link_last:`
..... 1972, 2078, 2432
`__pdf_backend_link_margin:n` ...
..... 1974, 2086, 2433
`\g__pdf_backend_link_math_bool` ..
..... 1796, 1819, 1820, 1823, 1833
`__pdf_backend_link_minima:` .. 1802
`__pdf_backend_link_outerbox:n` 1802
`\g__pdf_backend_link_sf_int`
..... 1795, 1928, 1939, 1940
`__pdf_backend_link_sf_restore:` 1802
`__pdf_backend_link_sf_save:` . 1802
`\l__pdf_backend_model_box` . 1753,
1838, 1875, 1883, 1894, 1909, 1916
`__pdf_backend_objcompresslevel:n`
..... 2234
`\g__pdf_backend_object_int`
1672, 1676, 1679, 1731, 1734, 1747,
1751, 1758, 1759, 1763, 1782, 1783,
1786, 1860, 1861, 2316, 2320, 2323,
2363, 2365, 2370, 2402, 2403, 2406
`__pdf_backend_object_last:`
..... 1750, 2226, 2369, 2514
`__pdf_backend_object_new:nn` ...
..... 1674, 2142, 2318, 2514
`__pdf_backend_object_now:nn` ...
..... 1729, 2197, 2361, 2514
`\g__pdf_backend_object_prop`
..... 1672, 1680, 1691, 1701,
2141, 2151, 2173, 2316, 2324, 2331
`__pdf_backend_object_ref:n` 1674,
1688, 1702, 2142, 2318, 2337, 2514
`__pdf_backend_object_write:nn` ..
..... 1684, 2164, 2328, 2514
`__pdf_backend_object_write:nnn` 2328
`__pdf_backend_object_write_-`
`array:nn` 1684, 2328
`__pdf_backend_object_write_-`
`dict:nn` 1684, 2328
`__pdf_backend_object_write_-`
`fstream:nn` 2328
`__pdf_backend_object_write_-`
`stream:nn` 1684, 2328

