

# 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}{2020-05-05}{ }
6 {L3 backend support: dvipdfmx}
7 </dvipdfmx>
8 <*dvips>
9 {l3backend-dvips.def}{2020-05-05}{ }
10 {L3 backend support: dvips}
11 </dvips>
12 <*dvisvgm>
13 {l3backend-dvisvgm.def}{2020-05-05}{ }
14 {L3 backend support: dvisvgm}
15 </dvisvgm>
16 <*pdfmode>
17 {l3backend-pdfmode.def}{2020-05-05}{ }
18 {L3 backend support: PDF mode}
19 </pdfmode>
20 <*xdvipdfmx>
21 {l3backend-xdvipdfmx.def}{2020-05-05}{ }
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 (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` keyword everything is kept in sync: the transformation matrix is set to the current point automatically. Note that this is still inside the text (BT ...ET block).

`\_kernel_backend_literal_pdf:x`

```

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.

`\__kernel_backend_literal_pdf:x`

```

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<sub>E</sub>X 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 T<sub>E</sub>X 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 T<sub>E</sub>X box is inserted to get things back on track. The clip path needs to come between those two such that it lines up with the current point, as does the T<sub>E</sub>X 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>
```

`\_color_backend_pickup:N` Allow for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 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` 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 {
```

```
\_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
```

```

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<sub>E</sub>X 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:cm: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  $\text{\TeX}$  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_but:</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_but:</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 | xdvipdfmx>

```

(End definition for `\_draw_backend_cm:nnnn` and `\_draw_backend_cm_aux:nnnn`.)

```

\_draw_backend_cm_decompose:nnnnN
\_draw_backend_cm_decompose_auxi:nnnnN
\_draw_backend_cm_decompose_auxii:nnnnN
\_draw_backend_cm_decompose_auxiii:nnnnN

```

Internally, transformations for drawing are tracked as a matrix. Not all engines provide a way of dealing with this: if we use a raw matrix, the engine loses track of positions (for example for hyperlinks), and this is not desirable. They do, however, allow us to track rotations and scalings. Luckily, we can decompose any (two-dimensional) matrix into two rotations and a single scaling:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} \cos \beta & \sin \beta \\ -\sin \beta & \cos \beta \end{bmatrix} \begin{bmatrix} w_1 & 0 \\ 0 & w_2 \end{bmatrix} \begin{bmatrix} \cos \gamma & \sin \gamma \\ -\sin \gamma & \cos \gamma \end{bmatrix}$$

The parent matrix can be converted to

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} E & H \\ -H & E \end{bmatrix} + \begin{bmatrix} F & G \\ G & -F \end{bmatrix}$$

From these, we can find that

$$\begin{aligned} \frac{w_1 + w_2}{2} &= \sqrt{E^2 + H^2} \\ \frac{w_1 - w_2}{2} &= \sqrt{F^2 + G^2} \\ \gamma - \beta &= \tan^{-1}(G/F) \\ \gamma + \beta &= \tan^{-1}(H/E) \end{aligned}$$

at which point we just have to do various pieces of re-arrangement to get all of the values. (See J. Blinn, *IEEE Comput. Graph. Appl.*, 1996, **16**, 82–88.) There is one wrinkle: the PostScript (and PDF) way of specifying a transformation matrix exchanges where one would normally expect  $B$  and  $C$  to be.

```

958 <*dviptdmtx | xdvipdfmx>
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 \</dvi>

```

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

`\_draw_backend_box_use:Nnnnn`

Inserting a  $\text{\TeX}$  box transformed to the requested position and using the current matrix is done using a mixture of  $\text{\TeX}$  and low-level manipulation. The offset can be handled by  $\text{\TeX}$ , 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   \<dvi>
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   \</dvi>
1017   \hbox_overlap_right:n { \box_use:N #1 }
1018   \<dvi>
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
\__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 }

```

Measuring the graphic is done by boxing up: for PDF graphics we could use `\tex_pdfximagebbox:D`, but if doesn't work for other types. As the box always starts at (0,0) there is no need to worry about the lower-left position.

```

1365 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1366 {
1367   \tex_immediate:D \tex_pdfximage:D
1368   \bool_lazy_or:nnT
1369   { \l__graphics_interpolate_bool }
1370   { ! \tl_if_empty_p:N \l__graphics_decodearray_tl }
1371   {
1372     attr ~
1373     {
1374       \tl_if_empty:NF \l__graphics_decodearray_tl
1375       { /Decode~[ \l__graphics_decodearray_tl ] }
1376       \bool_if:NT \l__graphics_interpolate_bool
1377       { /Interpolate~true }
1378     }
1379   }
1380   \int_compare:nNnT \l__graphics_page_int > 0
1381   { page ~ \int_use:N \l__graphics_page_int }
1382   \tl_if_empty:NF \l__graphics_pagebox_tl
1383   { \l__graphics_pagebox_tl }
1384   {#1}
1385   \hbox_set:Nn \l__graphics_internal_box
1386   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1387   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1388   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1389   \int_const:cn { c__graphics_graphics_#1 \l__graphics_graphics_attr_tl _int }
1390   { \tex_the:D \tex_pdflastximage:D }
1391   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1392 }

```

(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.

```

1393 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1394 {
1395   \tex_pdfrefximage:D
1396   \int_use:c { c__graphics_graphics_#1 \l__graphics_graphics_attr_tl _int }
1397 }
1398 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1399 \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 1400 \sys_if_shell:T
\l__graphics_backend_name_str 1401 {
\l__graphics_backend_ext_str 1402 \str_new:N \l__graphics_backend_dir_str
1403 \str_new:N \l__graphics_backend_name_str
1404 \str_new:N \l__graphics_backend_ext_str
1405 \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1406 {
1407 \file_parse_full_name:nNNN {#1}
1408 \l__graphics_backend_dir_str
1409 \l__graphics_backend_name_str
1410 \l__graphics_backend_ext_str
1411 \exp_args:Nx \__graphics_backend_getbb_eps:nn
1412 {
1413 \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1414 -converted-to.pdf
1415 }
1416 {#1}
1417 }
1418 \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1419 {
1420 \file_compare_timestamp:nNnT {#2} > {#1}
1421 {
1422 \sys_shell_now:n
1423 { repstopdf ~ #2 ~ #1 }
1424 }
1425 \tl_set:Nn \l_graphics_name_tl {#1}
1426 \__graphics_backend_getbb_pdf:n {#1}
1427 }
1428 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1429 {
1430 \file_parse_full_name:nNNN {#1}
1431 \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1432 \exp_args:Nx \__graphics_backend_include_pdf:n
1433 {
1434 \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1435 -converted-to.pdf
1436 }
1437 }
1438 }

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

1439 </pdfmode>

```

### 5.3 dvipdfmx backend

```

1440 <*dvipdfmx | xdvipdfmx>

\__graphics_backend_getbb_eps:n Simply use the generic functions: only for dvipdfmx in the extraction cases.
\__graphics_backend_getbb_jpg:n 1441 <*initex>
\__graphics_backend_getbb_pdf:n 1442 \use:n
\__graphics_backend_getbb_png:n 1443 </initex>

```

```

1444 <*package>
1445 \AtBeginDocument
1446 </package>
1447 { \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n }
1448 <*dvipdfmx>
1449 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1450 {
1451   \int_zero:N \l_graphics_page_int
1452   \tl_clear:N \l_graphics_pagebox_tl
1453   \graphics_extract_bb:n {#1}
1454 }
1455 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1456 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1457 {
1458   \tl_clear:N \l_graphics_decodearray_tl
1459   \bool_set_false:N \l_graphics_interpolate_bool
1460   \graphics_extract_bb:n {#1}
1461 }
1462 </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.

```

1463 \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> 1464 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1 1465 { 1466   \__kernel_backend_literal:x 1467   { 1468     PSfile = #1 \c_space_tl 1469     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl 1470     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl 1471     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl 1472     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim 1473   } 1474 } 1475 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1 1476 { \__graphics_backend_include_auxi:nn {#1} { image } } 1477 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n 1478 &lt;*dvipdfmx&gt; 1479 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1 1480 { \__graphics_backend_include_auxi:nn {#1} { epdf } } 1481 &lt;/dvipdfmx&gt; </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.

```

1482 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2

```

```

1483 {
1484   \_graphics_backend_include_auxii:xnn
1485   {
1486     \tl_if_empty:NF \l_graphics_pagebox_tl
1487     { : \l_graphics_pagebox_tl }
1488     \int_compare:nNnT \l_graphics_page_int > 1
1489     { :P \int_use:N \l_graphics_page_int }
1490     \tl_if_empty:NF \l_graphics_decodearray_tl
1491     { :D \l_graphics_decodearray_tl }
1492     \bool_if:NT \l_graphics_interpolate_bool
1493     { :I }
1494   }
1495   {#1} {#2}
1496 }
1497 \cs_new_protected:Npn \_graphics_backend_include_auxii:nnn #1#2#3
1498 {
1499   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1500   {
1501     \__kernel_backend_literal:x
1502     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1503   }
1504   { \_graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1505 }
1506 \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!

```

1507 \cs_new_protected:Npn \_graphics_backend_include_auxiii:nnn #1#2#3
1508 {
1509   \int_gincr:N \g__graphics_track_int
1510   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1511   \__kernel_backend_literal:x
1512   {
1513     pdf:#3~
1514     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1515     \int_compare:nNnT \l_graphics_page_int > 1
1516     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1517     \tl_if_empty:NF \l_graphics_pagebox_tl
1518     {
1519       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1520       bbox ~
1521       \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1522       \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1523       \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1524       \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1525     }
1526     (#1)
1527     \bool_lazy_or:nnT
1528     { \l_graphics_interpolate_bool }
1529     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1530     {
1531       <<
1532       \tl_if_empty:NF \l_graphics_decodearray_tl

```

```

1533             { /Decode~[ \l_graphics_decodearray_tl ] }
1534             \bool_if:NT \l_graphics_interpolate_bool
1535             { /Interpolate~true> }
1536         >>
1537     }
1538 }
1539 }

```

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

```

1540 </dvipdfmx | xdvipdfmx>

```

## 5.4 xdvipdfmx backend

```

1541 < *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  $\text{X}\text{\TeX}$  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

```

```

1542 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1543 {
1544     \int_zero:N \l_graphics_page_int
1545     \tl_clear:N \l_graphics_pagebox_tl
1546     \_graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1547 }
1548 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1549 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1550 {
1551     \tl_clear:N \l_graphics_decodearray_tl
1552     \bool_set_false:N \l_graphics_interpolate_bool
1553     \_graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1554 }
1555 \cs_new_protected:Npn \_graphics_backend_getbb_auxi:nN #1#2
1556 {
1557     \int_compare:nNnTF \l_graphics_page_int > 1
1558     { \_graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
1559     { \_graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1560 }
1561 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:nN #1#2#3
1562 { \_graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1563 \cs_generate_variant:Nn \_graphics_backend_getbb_auxii:nN { V }
1564 \cs_new_protected:Npn \_graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1565 {
1566     \tl_if_empty:NTF \l_graphics_pagebox_tl
1567     { \_graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1568     { \_graphics_backend_getbb_auxv:nNnn }
1569     {#1} #2 {#3} {#4}
1570 }
1571 \cs_new_protected:Npn \_graphics_backend_getbb_auxiv:nNnn #1#2#3#4#5
1572 {
1573     \use:x
1574     {

```



```

1575         \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1576         { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1577     }
1578 }
1579 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1580 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1581 {
1582     \graphics_bb_restore:nF {#1#3}
1583     { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1584 }
1585 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1586 {
1587     \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1588     \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1589     \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1590     \graphics_bb_save:n {#1#3}
1591 }
1592 \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<sub>Y</sub>TeX 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.

```

1593 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1594 {
1595     \tex_XeTeXpdffile:D
1596     \__graphics_backend_include_pdf_quote:w #1 "#1" \q_stop \c_space_tl
1597     \int_compare:nNnT \l_graphics_page_int > 0
1598     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1599     \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1600 }
1601 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \q_stop
1602 { " #2 " }

```

(End definition for \\_\_graphics\_backend\_include\_pdf:n and \\_\_graphics\_backend\_include\_bitmap\_quote:w.)

```
1603 </xdvipdfmx>
```

## 5.5 dvisvgm backend

```
1604 <*dvisvgm>
```

\\_\_graphics\_backend\_getbb\_eps:n Simply use the generic function.

```

1605 <*initex>
1606 \use:n
1607 </initex>
1608 <*package>
1609 \AtBeginDocument
1610 </package>
1611 { \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`

```

1612 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1613 {
1614   \int_zero:N \l_graphics_page_int
1615   \tl_clear:N \l_graphics_pagebox_tl
1616   \graphics_extract_bb:n {#1}
1617 }
1618 \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

```

1619 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1620 {
1621   \tl_clear:N \l_graphics_decodearray_tl
1622   \bool_set_false:N \l_graphics_interpolate_bool
1623   \graphics_extract_bb:n {#1}
1624 }

```

(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`

```

1625 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1626 { \_graphics_backend_include:nn { PSfile } {#1} }
1627 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1628 { \_graphics_backend_include:nn { pdffile } {#1} }
1629 \cs_new_protected:Npn \_graphics_backend_include:nn #1#2
1630 {
1631   \__kernel_backend_literal:x
1632   {
1633     #1 = #2 \c_space_tl
1634     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1635     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1636     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1637     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1638   }
1639 }

```

(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.

```

1640 \cs_new_protected:Npn \_graphics_backend_include_png:n #1
1641 {
1642   \__kernel_backend_literal:x
1643   {
1644     dvisvgm:img~
1645     \dim_to_decimal:n { \l_graphics_ury_dim } ~
1646     \dim_to_decimal:n { \l_graphics_ury_dim } ~

```

```

1647         \_graphics_backend_include_bitmap_quote:w #1 " #1 " \q_stop
1648     }
1649 }
1650 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_png:n
1651 \cs_new:Npn \_graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \q_stop
1652 { " #2 " }

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

1653 </divsvgm>
1654 </initex | package>

```

## 6 l3backend-pdf Implementation

```

1655 <*initex | package>
1656 <@@=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

1657 \box_new:N \l\_pdf\_internal\_box

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

```

### 6.2 dvips backend

```

1658 <*dvips>

\_pdf_backend_pdfmark:n Used often enough it should be a separate function.
\_pdf_backend_pdfmark:x
1659 \cs_new_protected:Npn \_pdf_backend_pdfmark:n #1
1660 { \_kernel_backend_postscript:n { mark #1 ~ pdfmark } }
1661 \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
1662 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
1663 { \_pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
1664 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
1665 { \_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.

```
1666 \int_new:N \g__pdf_backend_object_int
1667 \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.

```
1668 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
1669 {
1670   \int_gincr:N \g__pdf_backend_object_int
1671   \int_const:cn
1672     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
1673     { \g__pdf_backend_object_int }
1674   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
1675 }
1676 \cs_new:Npn \__pdf_backend_object_ref:n #1
1677 { { 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.

```
1678 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
1679 {
1680   \__pdf_backend_pdfmark:x
1681   {
1682     /_objdef ~ \__pdf_backend_object_ref:n {#1}
1683     /type
1684     \str_case_e:nn
1685       { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
1686       {
1687         { array } { /array }
1688         { dict } { /dict }
1689         { fstream } { /stream }
1690         { stream } { /stream }
1691       }
1692     /OBJ
1693   }
1694   \use:c
1695     { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
1696     { \__pdf_backend_object_ref:n {#1} } {#2}
1697 }
1698 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
1699 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
1700 {
1701   \__pdf_backend_pdfmark:x
1702     { #1 [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
1703 }
1704 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
1705 {
1706   \__pdf_backend_pdfmark:x
1707     { #1 << \exp_not:n {#2} >> /PUT }
1708 }
1709 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
```

```

1710 {
1711   \exp_args:Nx
1712   \__pdf_backend_object_write_stream:nnn {#1} #2
1713 }
1714 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
1715 {
1716   \__kernel_backend_postscript:n
1717   {
1718     [nobreak]
1719     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
1720     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
1721   }
1722 }

```

(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 1723 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
1724 {
1725   \int_gincr:N \g__pdf_backend_object_int
1726   \__pdf_backend_pdfmark:x
1727   {
1728     /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
1729     /type
1730     \str_case:nn
1731       {#1}
1732       {
1733         { array } { /array }
1734         { dict } { /dict }
1735         { fstream } { /stream }
1736         { stream } { /stream }
1737       }
1738     /OBJ
1739   }
1740   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
1741   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
1742 }
1743 \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.

```

1744 \cs_new:Npn \__pdf_backend_object_last:
1745 { { pdf.obj \int_use:N \g__pdf_backend_object_int } }

```

(End definition for \\_\_pdf\_backend\_object\_last:.)

\\_\_pdf\_backend\_pageobject\_ref:n Page references are easy in dvips.

```

1746 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
1747 { { Page #1 } }

```

(End definition for \\_\_pdf\_backend\_pageobject\_ref:n.)

### 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.
1748 \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.
1749 \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.
1750 \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
object data lists. Here, to get the co-ordinates of the annotation, we need to have the
data collected at the PostScript level. That requires a bit of box trickery (effectively a
LATEX 2ε picture of zero size). Once the data is collected, use it to set up the annotation
border.
1751 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
1752 {
1753   \exp_args:Nf \__pdf_backend_annotation_aux:nnnn
1754   { \dim_eval:n {#1} } {#2} {#3} {#4}
1755 }
1756 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
1757 {
1758   \box_move_down:nn {#3}
1759   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
1760   \box_move_up:nn {#2}
1761   {
1762     \hbox:n
1763     {
1764       \tex_kern:D #1 \scan_stop:
1765       \__kernel_backend_postscript:n { pdf.save.ur }
1766       \tex_kern:D -#1 \scan_stop:
1767     }
1768   }
1769   \int_gincr:N \g__pdf_backend_object_int
1770   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
1771   \__pdf_backend_pdfmark:x
1772   {
1773     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
1774     pdf.rect
1775     #4 ~
1776     /ANN
1777   }
1778 }
(End definition for \__pdf_backend_annotation:nnnn.)

```

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

```

1779 \cs_new:Npn \_pdf_backend_annotation_last:
1780 { { 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.

```

1781 \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.

```

1782 \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.

```

1783 \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.

```

1784 \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.

```

1785 \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.

```

1786 \tl_new:N \l__pdf_breaklink_pdfmark_tl
1787 \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.

```

1788 \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.

```

1789 \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.

```

1790 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
1791 { \__pdf_backend_link_begin:nw { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
1792 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
1793 { \__pdf_backend_link_begin:nw {#1#2} }
1794 \cs_new_protected:Npn \__pdf_backend_link_begin:nw #1
1795 {
1796   \bool_if:NF \g__pdf_backend_link_bool
1797   { \__pdf_backend_link_begin_aux:nw {#1} }
1798 }
1799 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
1800 {
1801   \bool_gset_true:N \g__pdf_backend_link_bool
1802   \__kernel_backend_postscript:n
1803   { /pdf.link.dict ( #1 ) def }
1804   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
1805   \__pdf_backend_link_sf_save:
1806   \mode_if_math:TF
1807   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
1808   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
1809   \hbox_set:Nw \l__pdf_backend_content_box
1810   \__pdf_backend_link_sf_restore:
1811   \bool_if:NT \g__pdf_backend_link_math_bool
1812   { \c_math_toggle_token }
1813 }
1814 \cs_new_protected:Npn \__pdf_backend_link_end:
1815 {
1816   \bool_if:NT \g__pdf_backend_link_bool
1817   { \__pdf_backend_link_end_aux: }
1818 }
1819 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
1820 {
1821   \bool_if:NT \g__pdf_backend_link_math_bool
1822   { \c_math_toggle_token }
1823   \__pdf_backend_link_sf_save:
1824   \hbox_set_end:
1825   \__pdf_backend_link_minima:
1826   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
1827   \exp_args:Nx \__pdf_backend_link_outerbox:n
1828   {

```



```

1829 <*initex>
1830     \l_galley_total_left_margin_dim
1831 </initex>
1832 <*package>
1833     \int_if_odd:nTF { \value { page } }
1834         { \oddsidemargin }
1835         { \evensidemargin }
1836 </package>
1837 }
1838 \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
1839 { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
1840 \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
1841 \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
1842 \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
1843 \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
1844 {
1845     \hbox:n
1846     { \__kernel_backend_postscript:n { pdf.save.linkur } }
1847 }
1848 \int_gincr:N \g__pdf_backend_object_int
1849 \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
1850 \__kernel_backend_postscript:x
1851 {
1852     mark
1853     /objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
1854     \g__pdf_backend_link_dict_tl \c_space_tl
1855     pdf.rect
1856     /ANN ~ \l__pdf_breaklink_pdfmark_tl
1857 }
1858 \__pdf_backend_link_sf_restore:
1859 \bool_gset_false:N \g__pdf_backend_link_bool
1860 }
1861 \cs_new_protected:Npn \__pdf_backend_link_minima:
1862 {
1863     \hbox_set:Nn \l__pdf_backend_model_box { Gg }
1864     \__kernel_backend_postscript:x
1865     {
1866         /pdf.linkdp.pad ~
1867         \dim_to_decimal:n
1868         {
1869             \dim_max:nn
1870             {
1871                 \box_dp:N \l__pdf_backend_model_box
1872                 - \box_dp:N \l__pdf_backend_content_box
1873             }
1874             { Opt }
1875         } ~
1876         pdf.pt.dvi ~ def
1877         /pdf.linkht.pad ~
1878         \dim_to_decimal:n
1879         {
1880             \dim_max:nn
1881             {
1882                 \box_ht:N \l__pdf_backend_model_box

```

```

1883         - \box_ht:N \l__pdf_backend_content_box
1884     }
1885     { Opt }
1886 } ~
1887 pdf.pt.dvi ~ def
1888 }
1889 }
1890 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
1891 {
1892     \__kernel_backend_postscript:x
1893     {
1894         /pdf.outerbox
1895         [
1896             \dim_to_decimal:n {#1} ~
1897             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
1898         (*initex)
1899             \dim_to_decimal:n { #1 + \l_galley_text_width_dim } ~
1900         (/initex)
1901         (*package)
1902             \dim_to_decimal:n { #1 + \textwidth } ~
1903         (/package)
1904             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
1905         ]
1906         [ exch { pdf.pt.dvi } forall ] def
1907         /pdf.baselineskip ~
1908         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
1909         { pdf.pt.dvi ~ def }
1910         { pop ~ pop }
1911         ifelse
1912     }
1913 }
1914 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
1915 {
1916     \int_gset:Nn \g__pdf_backend_link_sf_int
1917     {
1918         \mode_if_horizontal:TF
1919         { \tex_spacefactor:D }
1920         { 0 }
1921     }
1922 }
1923 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
1924 {
1925     \mode_if_horizontal:T
1926     {
1927         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
1928         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
1929     }
1930 }

```

(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<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> end.

```

1931 <*package>
1932 \use_none:n
1933 {
1934   \cs_if_exist:NT \@makecol@hook
1935   {
1936     \tl_put_right:Nn \@makecol@hook
1937     {
1938       \box_if_empty:NF \@cclv
1939       {
1940         \vbox_set:Nn \@cclv
1941         {
1942           \__kernel_backend_postscript:n
1943           {
1944             pdf.globaldict /pdf.brokenlink.rect ~ known
1945             { pdf.bordertracking.continue }
1946             if
1947             }
1948             \vbox_unpack_drop:N \@cclv
1949             \__kernel_backend_postscript:n
1950             { pdf.bordertracking.endpage }
1951           }
1952         }
1953       }
1954       \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
1955       \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
1956       \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
1957     }
1958   }
1959 </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.

```

1960 \cs_new:Npn \__pdf_backend_link_last:
1961 { { 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.

```

1962 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
1963 {
1964   \__kernel_backend_postscript:x
1965   {
1966     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
1967   }
1968 }

```

(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.

```

1969 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2

```

```

1970 {
1971   \_kernel_backend_postscript:n { pdf.dest.anchor }
1972   \_pdf_backend_pdfmark:x
1973   {
1974     /View
1975     [
1976       \str_case:nnF {#2}
1977       {
1978         { xyz } { /XYZ ~ pdf.dest.point ~ null }
1979         { fit } { /Fit }
1980         { fitb } { /FitB }
1981         { fitbh } { /FitBH ~ pdf.dest.y }
1982         { fitbv } { /FitBV ~ pdf.dest.x }
1983         { fith } { /FitH ~ pdf.dest.y }
1984         { fitv } { /FitV ~ pdf.dest.x }
1985       }
1986       {
1987         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
1988       }
1989     ]
1990     /Dest ( \exp_not:n {#1} ) cvn
1991     /DEST
1992   }
1993 }
1994 \cs_new_protected:Npn \_pdf_backend_destination_box:nn #1#2
1995 {
1996   \group_begin:
1997   \hbox_set:Nn \l__pdf_internal_box {#2}
1998   \box_move_down:nn
1999   { \box_dp:N \l__pdf_internal_box }
2000   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2001   \box_use:N \l__pdf_internal_box
2002   \box_move_up:nn
2003   { \box_ht:N \l__pdf_internal_box }
2004   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ur } } }
2005   \_pdf_backend_pdfmark:n
2006   {
2007     /View
2008     [
2009       /FitR ~
2010       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2011       pdf.urx ~ pdf.ury ~ pdf.dest2device
2012     ]
2013     /Dest ( #1 ) cvn
2014     /DEST
2015   }
2016   \group_end:
2017 }

```

(End definition for \\_pdf\_backend\_destination:nn and \\_pdf\_backend\_destination\_box:nn.)

#### 6.2.4 Structure

\\_pdf\_backend\_compresslevel:n Doable for the usual ps2pdf method.  
\\_pdf\_backend\_compress\_objects:n

```

2018 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2019 {
2020   \int_compare:nNnT {#1} = 0
2021   {
2022     \__kernel_backend_literal_postscript:n
2023     {
2024       /setdistillerparams ~ where
2025       { pop << /CompressPages ~ false >> setdistillerparams }
2026       if
2027     }
2028   }
2029 }
2030 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2031 {
2032   \bool_if:nF {#1}
2033   {
2034     \__kernel_backend_literal_postscript:n
2035     {
2036       /setdistillerparams ~ where
2037       { pop << /CompressStreams ~ false >> setdistillerparams }
2038       if
2039     }
2040   }
2041 }

```

(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

```

2042 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2043 \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:

```

2044 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2045 \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:

```

2046 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2047 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2048 \cs_new_protected:Npn \__pdf_backend_emc:
2049 { \__pdf_backend_pdfmark:n { /EMC } }

```

(End definition for \\_\_pdf\_backend\_bdc:nn and \\_\_pdf\_backend\_emc:.)

2050 </dvips>

## 6.3 pdfmode backend

2051 `\*pdfmode\`

### 6.3.1 Annotations

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

```
2052 \cs_new_protected:Npx \_pdf_backend_annotation:nnnn #1#2#3#4
2053 {
2054   \cs_if_exist:NTF \tex_pdfextension:D
2055   { \tex_pdfextension:D annot ~ }
2056   { \tex_pdfannot:D }
2057   width ~ \exp_not:N \dim_eval:n {#1} ~
2058   height ~ \exp_not:N \dim_eval:n {#2} ~
2059   depth ~ \exp_not:N \dim_eval:n {#3} ~
2060   {#4}
2061 }
```

(End definition for `\_pdf_backend_annotation:nnnn`.)

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

```
2062 \cs_new:Npx \_pdf_backend_annotation_last:
2063 {
2064   \exp_not:N \int_value:w
2065   \cs_if_exist:NTF \tex_pdffeedback:D
2066   { \exp_not:N \tex_pdffeedback:D lastannot ~ }
2067   { \exp_not:N \tex_pdflastannot:D }
2068   \c_space_tl 0 ~ R
2069 }
```

(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:`

```
2070 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2071 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2072 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2073 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2074 \cs_new_protected:Npx \_pdf_backend_link_begin:nnnw #1#2#3
2075 {
2076   \cs_if_exist:NTF \tex_pdfextension:D
2077   { \tex_pdfextension:D startlink ~ }
2078   { \tex_pdfstartlink:D }
2079   attr {#1}
2080   #2 {#3}
2081 }
2082 \cs_new_protected:Npx \_pdf_backend_link_end:
2083 {
2084   \cs_if_exist:NTF \tex_pdfextension:D
2085   { \tex_pdfextension:D endlink \scan_stop: }
2086   { \tex_pdfendlink:D }
2087 }
```

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

`\_pdf_backend_link_last:` Formatted for direct use.

```

2088 \cs_new:Npx \_pdf_backend_link_last:
2089 {
2090   \exp_not:N \int_value:w
2091   \cs_if_exist:NTF \tex_pdffeedback:D
2092     { \exp_not:N \tex_pdffeedback:D lastlink ~ }
2093     { \exp_not:N \tex_pdflastlink:D }
2094   \c_space_tl 0 ~ R
2095 }

```

(End definition for `\_pdf_backend_link_last:`.)

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

```

2096 \cs_new_protected:Npx \_pdf_backend_link_margin:n #1
2097 {
2098   \cs_if_exist:NTF \tex_pdfvariable:D
2099     { \exp_not:N \tex_pdfvariable:D linkmargin }
2100     { \exp_not:N \tex_pdflinkmargin:D }
2101     \exp_not:N \dim_eval:n {#1} \scan_stop:
2102 }

```

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

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

`\_pdf_backend_destination_box:nn`

```

2103 \cs_new_protected:Npx \_pdf_backend_destination:nn #1#2
2104 {
2105   \cs_if_exist:NTF \tex_pdfextension:D
2106     { \exp_not:N \tex_pdfextension:D dest ~ }
2107     { \exp_not:N \tex_pdfdest:D }
2108     name {#1}
2109     \exp_not:N \str_case:nnF {#2}
2110     {
2111       { xyz } { xyz }
2112       { fit } { fit }
2113       { fitb } { fitb }
2114       { fitbh } { fitbh }
2115       { fitbv } { fitbv }
2116       { fith } { fith }
2117       { fitv } { fitv }
2118     }
2119     { xyz ~ zoom \exp_not:N \fp_eval:n { #2 * 10 } }
2120     \scan_stop:
2121 }
2122 \cs_new_protected:Npx \_pdf_backend_destination_box:nn #1#2
2123 {
2124   \group_begin:
2125     \hbox_set:Nn \l__pdf_internal_box {#2}
2126     \cs_if_exist:NTF \tex_pdfextension:D
2127       { \exp_not:N \tex_pdfextension:D dest ~ }
2128       { \exp_not:N \tex_pdfdest:D }
2129     name {#1}
2130     fitr ~

```

```

2131         width \exp_not:N \box_wd:N \l__pdf_internal_box
2132         height \exp_not:N \box_ht:N \l__pdf_internal_box
2133         depth \exp_not:N \box_dp:N \l__pdf_internal_box
2134         \box_use:N \l__pdf_internal_box
2135     \group_end:
2136 }

```

(End definition for `\__pdf_backend_destination:nn` and `\__pdf_backend_destination_box:nn`.)

### 6.3.2 Catalogue entries

```

\__pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
2137 \cs_new_protected:Npx \__pdf_backend_catalog_gput:nn #1#2
2138 {
2139     \cs_if_exist:NTF \tex_pdfextension:D
2140     { \tex_pdfextension:D catalog }
2141     { \tex_pdfcatalog:D }
2142     { / #1 ~ #2 }
2143 }
2144 \cs_new_protected:Npx \__pdf_backend_info_gput:nn #1#2
2145 {
2146     \cs_if_exist:NTF \tex_pdfextension:D
2147     { \tex_pdfextension:D info }
2148     { \tex_pdfinfo:D }
2149     { / #1 ~ #2 }
2150 }

```

(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.

```

2151 \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
2152 \group_begin:
2153 \cs_set_protected:Npn \__pdf_tmp:w #1#2
2154 {
2155     \cs_new_protected:Npx \__pdf_backend_object_new:nn ##1##2
2156     {
2157         #1 reserveobjnum ~
2158         \int_const:cn
2159         { c__pdf_backend_object_ \exp_not:N \tl_to_str:n {##1} _int }
2160         {#2}
2161         \prop_gput:Nnn \exp_not:N \g__pdf_backend_object_prop {##1} {##2}
2162     }
2163 }
2164 \cs_if_exist:NTF \tex_pdfextension:D
2165 {
2166     \__pdf_tmp:w
2167     { \tex_pdfextension:D obj ~ }
2168     { \exp_not:N \tex_pdffeedback:D lastobj }

```



```

2169     }
2170     { \_pdf_tmp:w { \tex_pdfobj:D } { \tex_pdflastobj:D } }
2171 \group_end:
2172 \cs_new:Npn \_pdf_backend_object_ref:n #1
2173 { \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 2174 \group_begin:
\_pdf_exp_not_i:nn 2175 \cs_set_protected:Npn \_pdf_tmp:w #1
\_pdf_exp_not_ii:nn 2176 {
2177     \cs_new_protected:Npn \_pdf_backend_object_write:nn ##1##2
2178     {
2179         \tex_immediate:D #1 useobjnum ~
2180         \int_use:c
2181         { c\_pdf_backend_object_ \tl_to_str:n {##1} _int }
2182         \str_case_e:nn
2183         { \prop_item:Nn \g\_pdf_backend_object_prop {##1} }
2184         {
2185             { array } { { [ ~ \exp_not:n {##2} ~ ] } }
2186             { dict } { { << ~ \exp_not:n {##2} ~ >> } }
2187             { fstream }
2188             {
2189                 stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2190                 file ~ { \_pdf_exp_not_ii:nn ##2 }
2191             }
2192             { stream }
2193             {
2194                 stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2195                 { \_pdf_exp_not_ii:nn ##2 }
2196             }
2197         }
2198     }
2199 }
2200 \cs_if_exist:NTF \tex_pdfextension:D
2201 { \_pdf_tmp:w { \tex_pdfextension:D obj ~ } }
2202 { \_pdf_tmp:w { \tex_pdfobj:D } }
2203 \group_end:
2204 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
2205 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2206 \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 2207 \group_begin:
2208 \cs_set_protected:Npn \_pdf_tmp:w #1
2209 {
2210     \cs_new_protected:Npn \_pdf_backend_object_now:nn ##1##2
2211     {
2212         \tex_immediate:D #1
2213         \str_case:nn
2214         {##1}

```

```

2215         {
2216             { array } { { [ ~ \exp_not:n {##2} ~ ] } }
2217             { dict } { { << ~ \exp_not:n {##2} ~ >> } }
2218             { fstream }
2219             {
2220                 stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2221                 file ~ { \_pdf_exp_not_ii:nn ##2 }
2222             }
2223             { stream }
2224             {
2225                 stream ~ attr ~ { \_pdf_exp_not_i:nn ##2 } ~
2226                 { \_pdf_exp_not_ii:nn ##2 }
2227             }
2228         }
2229     }
2230 }
2231 \cs_if_exist:NTF \tex_pdfextension:D
2232 { \_pdf_tmp:w { \tex_pdfextension:D obj ~ } }
2233 { \_pdf_tmp:w { \tex_pdfobj:D } }
2234 \group_end:
2235 \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.

```

2236 \cs_new:Npx \_pdf_backend_object_last:
2237 {
2238     \exp_not:N \int_value:w
2239     \cs_if_exist:NTF \tex_pdffeedback:D
2240     { \exp_not:N \tex_pdffeedback:D lastobj ~ }
2241     { \exp_not:N \tex_pdflastobj:D }
2242     \c_space_tl 0 ~ R
2243 }

```

(End definition for \\_pdf\_backend\_object\_last:.)

\\_pdf\_backend\_pageobject\_ref:n The usual wrapper situation.

```

2244 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2245 {
2246     \exp_not:N \int_value:w
2247     \cs_if_exist:NTF \tex_pdffeedback:D
2248     { \exp_not:N \tex_pdffeedback:D pageref }
2249     { \exp_not:N \tex_pdfpageref:D }
2250     \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2251 }

```

(End definition for \\_pdf\_backend\_pageobject\_ref:n.)

### 6.3.4 Structure

\\_pdf\_backend\_compresslevel:n Simply pass data to the engine.

```

\_pdf_backend_compress_objects:n 2252 \cs_new_protected:Npx \_pdf_backend_compresslevel:n #1
\_pdf_backend_objcompresslevel:n 2253 {
2254     \exp_not:N \tex_global:D
2255     \cs_if_exist:NTF \tex_pdfcompresslevel:D

```

```

2256     { \tex_pdfcompresslevel:D }
2257     { \tex_pdfvariable:D compresslevel }
2258     \exp_not:N \int_value:w \exp_not:N \int_eval:n {#1} \scan_stop:
2259   }
2260   \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2261   {
2262     \bool_if:nTF {#1}
2263     { \__pdf_backend_objcompresslevel:n { 2 } }
2264     { \__pdf_backend_objcompresslevel:n { 0 } }
2265   }
2266   \cs_new_protected:Npx \__pdf_backend_objcompresslevel:n #1
2267   {
2268     \exp_not:N \tex_global:D
2269     \cs_if_exist:NTF \tex_pdfobjcompresslevel:D
2270     { \tex_pdfobjcompresslevel:D }
2271     { \tex_pdfvariable:D objcompresslevel }
2272     #1 \scan_stop:
2273   }

(End definition for \__pdf_backend_compresslevel:n, \__pdf_backend_compress_objects:n, and \__-
pdf_backend_objcompresslevel:n.)

```

\\_pdf\_backend\_version\_major\_gset:n At present, we don't have a primitive for the major version in pdfTeX, but we anticipate  
\\_pdf\_backend\_version\_minor\_gset:n one ...

```

2274   \cs_new_protected:Npx \__pdf_backend_version_major_gset:n #1
2275   {
2276     \cs_if_exist:NTF \tex_pdfvariable:D
2277     {
2278       \int_compare:nNnT \tex_luatexversion:D > { 106 }
2279       {
2280         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2281         \exp_not:N \int_eval:n {#1} \scan_stop:
2282       }
2283     }
2284     {
2285       \cs_if_exist:NT \tex_pdfmajorversion:D
2286       {
2287         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2288         \exp_not:N \int_eval:n {#1} \scan_stop:
2289       }
2290     }
2291   }
2292   \cs_new_protected:Npx \__pdf_backend_version_minor_gset:n #1
2293   {
2294     \exp_not:N \tex_global:D
2295     \cs_if_exist:NTF \tex_pdfminorversion:D
2296     { \exp_not:N \tex_pdfminorversion:D }
2297     { \tex_pdfvariable:D minorversion }
2298     \exp_not:N \int_eval:n {#1} \scan_stop:
2299   }

```

(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:

```

2300 \cs_new:Npx \__pdf_backend_version_major:
2301 {
2302   \cs_if_exist:NTF \tex_pdfvariable:D
2303   {
2304     \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2305     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2306     { 1 }
2307   }
2308   {
2309     \cs_if_exist:NTF \tex_pdfmajorversion:D
2310     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2311     { 1 }
2312   }
2313 }
2314 \cs_new:Npx \__pdf_backend_version_minor:
2315 {
2316   \exp_not:N \tex_the:D
2317   \cs_if_exist:NTF \tex_pdfminorversion:D
2318   { \exp_not:N \tex_pdfminorversion:D }
2319   { \tex_pdfvariable:D minorversion }
2320 }

```

(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>.

```

2321 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2322 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2323 \cs_new_protected:Npn \__pdf_backend_emc:
2324 { \__kernel_backend_literal_page:n { EMC } }

```

(End definition for \\_\_pdf\_backend\_bdc:nn and \\_\_pdf\_backend\_emc:.)

2325 </pdfmode>

## 6.4 dvipdfmx backend

2326 <\*dvipdfmx | xdvipdfmx>

\\_\_pdf\_backend:n A generic function for the backend PDF specials: used where we can.

```

2327 \cs_new_protected:Npx \__pdf_backend:n #1
2328 { \__kernel_backend_literal:n { pdf: #1 } }
2329 \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
2330 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2331 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2332 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2333 { \__pdf_backend:n { docinfo << /#1 ~ #2 >> } }

```

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

## 6.4.2 Objects

<code>\g__pdf_backend_object_int</code>	For tracking objects to allow finalisation.
<code>\g__pdf_backend_object_prop</code>	<pre> 2334 \int_new:N \g__pdf_backend_object_int 2335 \prop_new:N \g__pdf_backend_object_prop (End definition for \g__pdf_backend_object_int and \g__pdf_backend_object_prop.) </pre>
<code>\__pdf_backend_object_new:nn</code>	Objects are tracked at the macro level, but we don't have to do anything at this stage.
<code>\__pdf_backend_object_ref:n</code>	<pre> 2336 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2 2337 { 2338   \int_gincr:N \g__pdf_backend_object_int 2339   \int_const:cn 2340     { c__pdf_backend_object_ \tl_to_str:n {#1} _int } 2341     { \g__pdf_backend_object_int } 2342   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2} 2343 } 2344 \cs_new:Npn \__pdf_backend_object_ref:n #1 2345 { @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.) </pre>
<code>\__pdf_backend_object_write:nn</code>	This is where we choose the actual type.
<code>\__pdf_backend_object_write:nx</code>	
<code>\__pdf_backend_object_write:nnn</code>	
<code>\__pdf_backend_object_write_array:nn</code>	
<code>\__pdf_backend_object_write_dict:nn</code>	
<code>\__pdf_backend_object_write_fstream:nn</code>	
<code>\__pdf_backend_object_write_stream:nn</code>	
<code>\__pdf_backend_object_write_stream:nnnn</code>	
	<pre> 2346 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2 2347 { 2348   \exp_args:Nx \__pdf_backend_object_write:nnn 2349     { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2} 2350 } 2351 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx } 2352 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3 2353 { 2354   \use:c { __pdf_backend_object_write_ #1 :nn } 2355   { \__pdf_backend_object_ref:n {#2} } {#3} 2356 } 2357 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2 2358 { 2359   \__pdf_backend:x 2360     { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] } 2361 } 2362 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2 2363 { 2364   \__pdf_backend:x 2365     { obj ~ #1 ~ &lt;&lt; ~ \exp_not:n {#2} ~ &gt;&gt; } 2366 } 2367 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2 2368 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 } 2369 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2 2370 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 } 2371 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4 2372 { 2373   \__pdf_backend:x 2374   { 2375     #1 stream ~ #2 ~ 2376     ( \exp_not:n {#4} ) ~ &lt;&lt; \exp_not:n {#3} &gt;&gt; 2377   } </pre>

```
2378 }
```

(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 2379 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2380 {
2381   \int_gincr:N \g__pdf_backend_object_int
2382   \exp_args:Nnx \use:c { \_pdf_backend_object_write_ #1 :nn }
2383   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2384   {#2}
2385 }
2386 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }
```

(End definition for `\_pdf_backend_object_now:nn`.)

`\_pdf_backend_object_last:`

```
2387 \cs_new:Npn \_pdf_backend_object_last:
2388 { @pdf.obj \int_use:N \g__pdf_backend_object_int }
```

(End definition for `\_pdf_backend_object_last:.`)

`\_pdf_backend_pageobject_ref:n` Page references are easy in (x)dvipdfmx.

```
2389 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2390 { @page #1 }
```

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

### 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.

```
2391 \bool_new:N \g__pdf_landscape_bool
2392 <*package>
2393 \AtBeginDocument
2394 {
2395   \cs_if_exist:NT \landscape
2396   {
2397     \tl_put_right:Nn \landscape
2398     { \bool_gset_true:N \g__pdf_landscape_bool }
2399     \tl_put_left:Nn \endlandscape
2400     { \bool_gset_false:N \g__pdf_landscape_bool }
2401   }
2402 }
2403 </package>
```

(End definition for `\g__pdf_landscape_bool`.)

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

```
2404 \int_new:N \g__pdf_backend_annotation_int
```

(End definition for `\g__pdf_backend_annotation_int`.)

\\_pdf\_backend\_annotation:nnnn  
\\_pdf\_backend\_annotation\_aux:nnnn

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

```

2405 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2406 {
2407   \bool_if:NTF \g__pdf_landscape_bool
2408   {
2409     \box_move_up:nn {#2}
2410     {
2411       \vbox:n
2412       {
2413         \_pdf_backend_annotation_aux:nnnn
2414         { #2 + #3 } {#1} { Opt } {#4}
2415       }
2416     }
2417   }
2418   { \_pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4} }
2419 }
2420 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2421 {
2422   \int_gincr:N \g__pdf_backend_object_int
2423   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2424   \_pdf_backend:x
2425   {
2426     ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
2427     width ~ \dim_eval:n {#1} ~
2428     height ~ \dim_eval:n {#2} ~
2429     depth ~ \dim_eval:n {#3} ~
2430     <</Type/Annot #4 >>
2431   }
2432 }

```

(End definition for \\_pdf\_backend\_annotation:nnnn and \\_pdf\_backend\_annotation\_aux:nnnn.)

\\_pdf\_backend\_annotation\_last:

```

2433 \cs_new:Npn \_pdf_backend_annotation_last:
2434 { @pdf.obj \int_use:N \g__pdf_backend_annotation_int }

```

(End definition for \\_pdf\_backend\_annotation\_last:.)

\\_pdf\_backend\_link\_begin\_goto:nnw  
\\_pdf\_backend\_link\_begin\_user:nnw  
\\_pdf\_backend\_link\_begin:n  
\\_pdf\_backend\_link\_end:

All created using the same internals.

```

2435 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2436 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2437 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2438 { \_pdf_backend_link_begin:n {#1#2} }
2439 \cs_new_protected:Npn \_pdf_backend_link_begin:n #1
2440 {
2441   \_pdf_backend:n
2442   {
2443     bann
2444     <<
2445     /Type /Annot
2446     #1
2447     >>
2448   }

```

```

2449 }
2450 \cs_new_protected:Npn \__pdf_backend_link_end:
2451 { \__pdf_backend:n { eann } }

(End definition for \__pdf_backend_link_begin_goto:nw and others.)

```

\\_\_pdf\_backend\_link\_last: Data not available.

```

2452 \cs_new:Npn \__pdf_backend_link_last: { }

(End definition for \__pdf_backend_link_last:.)

```

\\_\_pdf\_backend\_link\_margin:n Pass to dvipdfmx.

```

2453 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2454 { \__kernel_backend_literal:x { dvipdfmx:config-g~ \dim_eval:n {#1} } }

(End definition for \__pdf_backend_link_margin:n.)

```

\\_\_pdf\_backend\_destination:nn Here, we need to turn the zoom into a scale. The method for FitR is from Alexander Grahn: the idea is to avoid needing to do any calculations in T<sub>E</sub>X by using the backend data for @xpos and @ypos.

```

2455 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2456 {
2457   \__pdf_backend:x
2458   {
2459     dest ~ ( \exp_not:n {#1} )
2460     [
2461       @thispage
2462       \str_case:nnF {#2}
2463       {
2464         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2465         { fit } { /Fit }
2466         { fitb } { /FitB }
2467         { fitbh } { /FitBH }
2468         { fitbv } { /FitBV ~ @xpos }
2469         { fith } { /FitH ~ @ypos }
2470         { fitv } { /FitV ~ @xpos }
2471       }
2472       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2473     ]
2474   }
2475 }
2476 \cs_new_protected:Npn \__pdf_backend_destination_box:nn #1#2
2477 {
2478   \group_begin:
2479   \hbox_set:Nn \l__pdf_internal_box {#2}
2480   \box_move_down:nn { \box_dp:N \l__pdf_internal_box }
2481   {
2482     \hbox:n
2483     {
2484       \__pdf_backend:n { obj ~ @pdf_ #1 _llx ~ @xpos }
2485       \__pdf_backend:n { obj ~ @pdf_ #1 _lly ~ @ypos }
2486     }
2487   }
2488   \box_use:N \l__pdf_internal_box
2489   \box_move_up:nn { \box_ht:N \l__pdf_internal_box }

```



```

2490     {
2491         \hbox:n
2492         {
2493             \_pdf_backend:n
2494             {
2495                 dest ~ (#1)
2496                 [
2497                     @thispage
2498                     /FitR ~
2499                     @pdf_ #1 _llx ~ @pdf_ #1 _lly ~
2500                     @xpos ~ @ypos
2501                 ]
2502             }
2503         }
2504     }
2505 \group_end:
2506 }

```

(End definition for \\_pdf\_backend\_destination:nn and \\_pdf\_backend\_destination\_box:nn.)

#### 6.4.4 Structure

\\_pdf\_backend\_compresslevel:n Pass data to the backend: these are a one-shot.  
\\_pdf\_backend\_compress\_objects:n

```

2507 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2508 { \_kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2509 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2510 {
2511     \bool_if:nF {#1}
2512     { \_kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2513 }

```

(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

```

2514 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2515 {
2516     \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2517     \_kernel_backend_literal:x { pdf:majorversion~ \_pdf_backend_version_major: }
2518 }
2519 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2520 {
2521     \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2522     \_kernel_backend_literal:x { pdf:minorversion~ \_pdf_backend_version_minor: }
2523 }

```

(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:

```

2524 \cs_new:Npn \_pdf_backend_version_major: { 1 }
2525 \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:`

```
2526 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2527 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2528 \cs_new_protected:Npn \_pdf_backend_emc:
2529 { \__kernel_backend_literal_page:n { EMC } }
```

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

```
2530 </dvipdfmx | xdvipdfmx>
```

## 6.5 dvisvgm backend

```
2531 <*dvisvgm>
```

### 6.5.1 Catalogue entries

`\_pdf_backend_catalog_gput:nn`  
`\_pdf_backend_info_gput:nn`

No-op.

```
2532 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2 { }
2533 \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`  
`\_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:`  
`\_pdf_backend_pageobject_ref:n`

All no-ops here.

```
2534 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2 { }
2535 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
2536 \cs_new_protected:Npn \_pdf_backend_object_write:nn #1#2 { }
2537 \cs_new_protected:Npn \_pdf_backend_object_write:nx #1#2 { }
2538 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
2539 \cs_new_protected:Npn \_pdf_backend_object_now:nx #1#2 { }
2540 \cs_new:Npn \_pdf_backend_object_last: { }
2541 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1 { }
```

(End definition for `\_pdf_backend_object_new:nn` and others.)

### 6.5.3 Structure

`\_pdf_backend_compresslevel:n`  
`\_pdf_backend_compress_objects:n`

These are all no-ops.

```
2542 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1 { }
2543 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1 { }
```

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

`\_pdf_backend_version_major_gset:n`  
`\_pdf_backend_version_minor_gset:n`

Data not available!

```
2544 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
2545 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1 { }
```

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

`\_pdf_backend_version_major:`  
`\_pdf_backend_version_minor:`

Data not available!

```
2546 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2547 \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: 2548 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2 { }
2549 \cs_new_protected:Npn \_pdf_backend_emc: { }

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

2550 </dvisvgm>
2551 </initex | package>

```

## 7 l3backend-header Implementation

```

2552 <*dvips & header>

pdf.globaldict A small global dictionary for backend use.

2553 true setglobal
2554 /pdf.globaldict 4 dict def
2555 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

2556
2557 /pdf.cvs { 65534 string cvs } def
2558 /pdf.dvi.pt { 72.27 mul Resolution div } def
2559 /pdf.pt.dvi { 72.27 div Resolution mul } def
2560 /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 2561 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad 2562 /pdf.linkdp.pad { 0 } def
2563 /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 2564 /pdf.rect
pdf.save.linkur 2565 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.llx 2566 /pdf.save.ll
pdf.lly 2567 {
pdf.urx 2568 currentpoint
pdf.ury 2569 /pdf.lly exch def
2570 /pdf.llx exch def
2571 }
2572 def
2573 /pdf.save.ur

```

```

2574 {
2575     currentpoint
2576     /pdf.ury exch def
2577     /pdf.urx exch def
2578 }
2579 def
2580 /pdf.save.linkll
2581 {
2582     currentpoint
2583     pdf.linkmargin add
2584     pdf.linkdp.pad add
2585     /pdf.lly exch def
2586     pdf.linkmargin sub
2587     /pdf.llx exch def
2588 }
2589 def
2590 /pdf.save.linkur
2591 {
2592     currentpoint
2593     pdf.linkmargin sub
2594     pdf.linkht.pad sub
2595     /pdf.ury exch def
2596     pdf.linkmargin add
2597     /pdf.urx exch def
2598 }
2599 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 2600 /pdf.dest.anchor
pdf.dev.y 2601 {
pdf.tmpa 2602     currentpoint exch
pdf.tmpb 2603     pdf.dvi.pt 72 add
pdf.tmpc 2604     /pdf.dest.x exch def
pdf.tmpd 2605     pdf.dvi.pt
2606     vsize 72 sub exch sub
2607     /pdf.dest.y exch def
2608 }
2609 def
2610 /pdf.dest.point
2611 { pdf.dest.x pdf.dest.y } def
2612 /pdf.dest2device
2613 {
2614     /pdf.dest.y exch def
2615     /pdf.dest.x exch def
2616     matrix currentmatrix
2617     matrix defaultmatrix
2618     matrix invertmatrix
2619     matrix concatmatrix

```

```

2620     cvx exec
2621     /pdf.dev.y exch def
2622     /pdf.dev.x exch def
2623     /pdf.tmpd exch def
2624     /pdf.tmpc exch def
2625     /pdf.tmpb exch def
2626     /pdf.tmpa exch def
2627     pdf.dest.x pdf.tmpa mul
2628     pdf.dest.y pdf.tmpc mul add
2629     pdf.dev.x add
2630     pdf.dest.x pdf.tmpb mul
2631     pdf.dest.y pdf.tmpd mul add
2632     pdf.dev.y add
2633   }
2634   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	2635 /pdf.bordertracking false def
pdf.brokenlink.skip	2636 /pdf.bordertracking.begin
pdf.brokenlink.dict	2637 {
pdf.bordertracking.endpage	2638 SDict /pdf.bordertracking true put
pdf.bordertracking.continue	2639 SDict /pdf.leftboundary undef
pdf.originx	2640 SDict /pdf.rightboundary undef
pdf.originy	2641 /a where
	2642 {
	2643 /a
	2644 {
	2645 currentpoint pop
	2646 SDict /pdf.rightboundary known dup
	2647 {
	2648 SDict /pdf.rightboundary get 2 index lt
	2649 { not }
	2650 if
	2651 }
	2652 if
	2653 { pop }
	2654 { SDict exch /pdf.rightboundary exch put }
	2655 ifelse
	2656 moveto
	2657 currentpoint pop
	2658 SDict /pdf.leftboundary known dup
	2659 {
	2660 SDict /pdf.leftboundary get 2 index gt
	2661 { not }
	2662 if
	2663 }
	2664 if
	2665 { pop }
	2666 { SDict exch /pdf.leftboundary exch put }

```

2667         ifelse
2668     }
2669     put
2670 }
2671 if
2672 }
2673 def
2674 /pdf.bordertracking.end
2675 {
2676     /a where { /a { moveto } put } if
2677     /x where { /x { 0 exch rmoveto } put } if
2678     SDict /pdf.leftboundary known
2679     { pdf.outerbox 0 pdf.leftboundary put }
2680     if
2681     SDict /pdf.rightboundary known
2682     { pdf.outerbox 2 pdf.rightboundary put }
2683     if
2684     SDict /pdf.bordertracking false put
2685 }
2686 def
2687 /pdf.bordertracking.endpage
2688 {
2689     pdf.bordertracking
2690     {
2691         pdf.bordertracking.end
2692         true setglobal
2693         pdf.globaldict
2694         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
2695         pdf.globaldict
2696         /pdf.brokenlink.skip pdf.baselineskip put
2697         pdf.globaldict
2698         /pdf.brokenlink.dict
2699         pdf.link.dict pdf.cvs put
2700         false setglobal
2701         mark pdf.link.dict cvx exec /Rect
2702         [
2703             pdf.llx
2704             pdf.lly
2705             pdf.outerbox 2 get pdf.linkmargin add
2706             currentpoint exch pop
2707             pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
2708         ]
2709         /ANN pdf.pdfmark
2710     }
2711     if
2712 }
2713 def
2714 /pdf.bordertracking.continue
2715 {
2716     /pdf.link.dict pdf.globaldict
2717     /pdf.brokenlink.dict get def
2718     /pdf.outerbox pdf.globaldict
2719     /pdf.brokenlink.rect get def
2720     /pdf.baselineskip pdf.globaldict

```

```

2721     /pdf.brokenlink.skip get def
2722 pdf.globaldict dup dup
2723 /pdf.brokenlink.dict undef
2724 /pdf.brokenlink.skip undef
2725 /pdf.brokenlink.rect undef
2726 currentpoint
2727 /pdf.originy exch def
2728 /pdf.originx exch def
2729 /a where
2730 {
2731     /a
2732     {
2733         moveto
2734         SDict
2735         begin
2736             currentpoint pdf.originy ne exch
2737             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     /x where
2751     {
2752         /x
2753         {
2754             0 exch rmoveto
2755             SDict~
2756             begin
2757                 currentpoint
2758                 pdf.originy ne exch pdf.originx ne or
2759                 {
2760                     pdf.save.linkll
2761                     /pdf.lly
2762                     pdf.lly pdf.outerbox 1 get sub def
2763                     pdf.bordertracking.begin
2764                 }
2765                 if
2766                 end
2767             }
2768             put
2769         }
2770         if
2771     }
2772     def

```

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

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

```

2773 /pdf.breaklink
2774 {
2775     pop
2776     counttomark 2 mod 0 eq
2777     {
2778         counttomark /pdf.count exch def
2779         {
2780             pdf.count 0 eq { exit } if
2781             counttomark 2 roll
2782             1 index /Rect eq
2783             {
2784                 dup 4 array copy
2785                 dup dup
2786                 1 get
2787                 pdf.outerbox pdf.rect.ht
2788                 pdf.linkmargin 2 mul add sub
2789                 3 exch put
2790                 dup
2791                 pdf.outerbox 2 get
2792                 pdf.linkmargin add
2793                 2 exch put
2794                 dup dup
2795                 3 get
2796                 pdf.outerbox pdf.rect.ht
2797                 pdf.linkmargin 2 mul add add
2798                 1 exch put
2799                 /pdf.currentrect exch def
2800                 pdf.breaklink.write
2801                 {
2802                     pdf.currentrect
2803                     dup
2804                     pdf.outerbox 0 get
2805                     pdf.linkmargin sub
2806                     0 exch put
2807                     dup
2808                     pdf.outerbox 2 get
2809                     pdf.linkmargin add
2810                     2 exch put
2811                     dup dup
2812                     1 get
2813                     pdf.baselineskip add
2814                     1 exch put
2815                     dup dup
2816                     3 get
2817                     pdf.baselineskip add
2818                     3 exch put
2819                     /pdf.currentrect exch def
2820                     pdf.breaklink.write

```



```

2821         }
2822         1 index 3 get
2823         pdf.linkmargin 2 mul add
2824         pdf.outerbox pdf.rect.ht add
2825         2 index 1 get sub
2826         pdf.baselineskip div round cvi 1 sub
2827         exch
2828         repeat
2829         pdf.currentrect
2830         dup
2831         pdf.outerbox 0 get
2832         pdf.linkmargin sub
2833         0 exch put
2834         dup dup
2835         1 get
2836         pdf.baselineskip add
2837         1 exch put
2838         dup dup
2839         3 get
2840         pdf.baselineskip add
2841         3 exch put
2842         dup 2 index 2 get 2 exch put
2843         /pdf.currentrect exch def
2844         pdf.breaklink.write
2845         SDict /pdf.pdfmark.good false put
2846         exit
2847     }
2848     { pdf.count 2 sub /pdf.count exch def }
2849     ifelse
2850 }
2851 loop
2852 }
2853 if
2854 /ANN
2855 }
2856 def
2857 /pdf.breaklink.write
2858 {
2859     counttomark 1 sub
2860     index /_objdef eq
2861     {
2862         counttomark -2 roll
2863         dup wcheck
2864         {
2865             readonly
2866             counttomark 2 roll
2867         }
2868         { pop pop }
2869     } ifelse
2870 }
2871 if
2872 counttomark 1 add copy
2873 pop pdf.currentrect
2874 /ANN pdfmark

```

```

2875 }
2876 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 function. Only mark types which are known are altered. At present, this is purely ANN pdf.outerbox marks, which are measured relative to the size of the baseline skip. If they are more than pdf.baselineskip one apparent line high, breaking is applied. pdf.pdfmark.dict

```

2877 /pdf.pdfmark
2878 {
2879   SDict /pdf.pdfmark.good true put
2880   dup /ANN eq
2881   {
2882     pdf.pdfmark.store
2883     pdf.pdfmark.dict
2884     begin
2885       Subtype /Link eq
2886       currentdict /Rect known and
2887       SDict /pdf.outerbox known and
2888       SDict /pdf.baselineskip known and
2889       {
2890         Rect 3 get
2891         pdf.linkmargin 2 mul add
2892         pdf.outerbox pdf.rect.ht add
2893         Rect 1 get sub
2894         pdf.baselineskip div round cvi 0 gt
2895         { pdf.breaklink }
2896         if
2897       }
2898       if
2899       end
2900       SDict /pdf.outerbox undef
2901       SDict /pdf.baselineskip undef
2902       currentdict /pdf.pdfmark.dict undef
2903     }
2904     if
2905     pdf.pdfmark.good
2906     { pdfmark }
2907     { cleartomark }
2908     ifelse
2909   }
2910   def
2911 /pdf.pdfmark.store
2912 {
2913   /pdf.pdfmark.dict 65534 dict def
2914   counttomark 1 add copy
2915   pop
2916   {
2917     dup mark eq
2918     {
2919       pop
2920       exit

```

```

2921     }
2922     {
2923         pdf.pdfmark.dict
2924         begin def end
2925     }
2926     ifelse
2927 }
2928 loop
2929 }
2930 def

```

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

```

2931 </dvips & header>

```

# Index

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

- A**
- \AtBeginDocument ..... 374, 431, 1316, 1445, 1609, 2393
  - \AtBeginDvi ..... 36, 37
- B**
- bool commands:
- \bool\_gset\_false:N ..... 577, 593, 619, 641, 657, 809, 1135, 1171, 1808, 1859, 2400
  - \bool\_gset\_true:N ..... 575, 644, 807, 1150, 1801, 1807, 2398
  - \bool\_if:NTF 584, 588, 606, 610, 614, 627, 632, 636, 648, 652, 820, 825, 830, 1109, 1154, 1341, 1376, 1492, 1534, 1796, 1811, 1816, 1821, 2407
  - \bool\_if:NTF ..... 2032, 2262, 2511
  - \bool\_lazy\_and:nnTF ..... 40
  - \bool\_lazy\_or:nnTF ..... 1368, 1527
  - \bool\_new:N ..... 578, 645, 810, 1151, 1784, 1785, 2391
  - \bool\_set\_false:N ..... 1351, 1459, 1552, 1622
- box commands:
- \box\_dp:N ..... 137, 139, 187, 189, 244, 246, 293, 295, 297, 299, 1838, 1871, 1872, 1897, 1999, 2133, 2480
  - \box\_ht:N ..... 139, 189, 246, 297, 299, 1388, 1589, 1843, 1882, 1883, 1904, 2003, 2132, 2489
  - \box\_if\_empty:NTF ..... 1938
  - \box\_move\_down:nn ..... 1758, 1838, 1998, 2480
  - \box\_move\_up:nn ..... 1760, 1843, 2002, 2409, 2489
  - \box\_new:N ..... 1657, 1748, 1749
  - \box\_set\_dp:Nn ..... 1302
  - \box\_set\_ht:Nn ..... 1301
  - \box\_set\_wd:Nn ..... 201, 1300
  - \box\_use:N . 144, 162, 176, 192, 219, 233, 249, 265, 277, 328, 345, 364, 760, 1017, 1303, 1789, 2001, 2134, 2488
  - \box\_wd:N . 138, 146, 188, 194, 245, 251, 294, 296, 332, 1387, 1588, 2131
- box internal commands:
- \\_\_box\_backend\_clip:N ..... 126, 181, 238, 282
  - \l\_box\_backend\_cos\_fp ..... 196
  - \\_\_box\_backend\_rotate:Nn ..... 148, 196, 253, 335
  - \\_\_box\_backend\_rotate\_aux:Nn ... 148, 196, 253
  - \\_\_box\_backend\_scale:Nnn ..... 165, 224, 268, 348
  - \l\_box\_backend\_sin\_fp ..... 196
  - \g\_box\_clip\_path\_int ..... 282
- C**
- clist commands:
- \clist\_map\_function:nN ..... 665, 840
  - \clist\_map\_function:nn ..... 1178
- color internal commands:
- \\_\_color\_backend\_cmyk:nnnn . 398, 467
  - \\_\_color\_backend\_cmyk\_aux:nnnn . 467
  - \\_\_color\_backend\_gray:n .... 398, 467
  - \\_\_color\_backend\_gray\_aux:n ... 467
  - \\_\_color\_backend\_pickup:N .. 372, 429
  - \\_\_color\_backend\_pickup:w 13, 372, 429
  - \\_\_color\_backend\_reset: .... 398, 467
  - \\_\_color\_backend\_rgb:nnn ... 398, 467
  - \\_\_color\_backend\_rgb\_aux:nnn .. 467
  - \\_\_color\_backend\_select:n .. 398, 467
  - \\_\_color\_backend\_spot:nn ... 398, 467
- color.fc ..... 398, 524
- cs commands:
- \cs\_generate\_variant:Nn ..... 28, 32, 35, 69, 97, 102, 113, 120, 424, 509, 523, 728, 734, 770, 918, 1026, 1057, 1506, 1563, 1579, 1661, 1698, 1743, 2204, 2235, 2329, 2351, 2386
  - \cs\_gset:Npx ..... 2516, 2521
  - \cs\_if\_exist:NTF ..... 36, 64, 72, 80, 86, 92, 376, 433, 503, 512, 901, 909, 1934, 2054, 2065, 2076, 2084, 2091, 2098, 2105, 2126, 2139, 2146, 2164, 2200, 2231, 2239, 2247, 2255, 2269, 2276, 2285, 2295, 2302, 2309, 2317, 2395
  - \cs\_if\_exist\_p:N ..... 41
  - \cs\_new:Npn .. 670, 845, 1182, 1592, 1601, 1651, 1676, 1744, 1746, 1779, 1960, 2044, 2045, 2172, 2205, 2206, 2344, 2387, 2389, 2433, 2452, 2524, 2525, 2535, 2540, 2541, 2546, 2547
  - \cs\_new:Npx ..... 2062, 2088, 2236, 2244, 2300, 2314

`\cs_new_eq:NN` .....  
     . 25, 522, 769, 775, 776, 916, 1025,  
       1318, 1347, 1398, 1399, 1447, 1455,  
       1477, 1548, 1611, 1618, 1650, 1789  
`\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, 1365, 1393, 1405, 1418,  
       1428, 1449, 1456, 1464, 1475, 1479,  
       1482, 1497, 1507, 1542, 1549, 1555,  
       1561, 1564, 1571, 1580, 1585, 1593,  
       1612, 1619, 1625, 1627, 1629, 1640,  
       1659, 1662, 1664, 1668, 1678, 1699,  
       1704, 1709, 1714, 1723, 1751, 1756,  
       1788, 1790, 1792, 1794, 1799, 1814,  
       1819, 1861, 1890, 1914, 1923, 1962,  
       1969, 1994, 2018, 2030, 2042, 2043,  
       2046, 2048, 2070, 2072, 2177, 2210,  
       2260, 2321, 2323, 2330, 2332, 2336,  
       2346, 2352, 2357, 2362, 2367, 2369,  
       2371, 2379, 2405, 2420, 2435, 2437,  
       2439, 2450, 2453, 2455, 2476, 2507,  
       2509, 2514, 2519, 2526, 2528, 2532,  
       2533, 2534, 2536, 2537, 2538, 2539,  
       2542, 2543, 2544, 2545, 2548, 2549  
`\cs_new_protected:Npx` 62, 70, 78, 84,  
     90, 501, 510, 899, 907, 2052, 2074,  
     2082, 2096, 2103, 2122, 2137, 2144,  
     2155, 2252, 2266, 2274, 2292, 2327  
`\cs_set_eq:NN` ..... 1955, 1956  
`\cs_set_protected:Npn` .....  
     ..... 378, 435, 2153, 2175, 2208

## D

dim commands:

`\dim_eval:n` ..... 1754, 2057, 2058,  
     2059, 2101, 2427, 2428, 2429, 2454  
`\dim_max:nn` ..... 1869, 1880  
`\dim_set:Nn` ... 1387, 1388, 1588, 1589  
`\dim_to_decimal:n` 293, 294, 295, 296,  
     297, 299, 1063, 1068, 1074, 1075,  
     1076, 1077, 1086, 1087, 1088, 1179,  
     1198, 1645, 1646, 1867, 1878, 1896,  
     1897, 1899, 1902, 1904, 1908, 1966  
`\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,  
     1469, 1470, 1471, 1472, 1521, 1522,  
     1523, 1524, 1634, 1635, 1636, 1637

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

<code>\__draw_backend_color_fill_-</code>	<code>\__draw_backend_nonzero_rule: ...</code>
<code>rgb:nnn</code> ..... <a href="#">691</a> , <a href="#">866</a> , <a href="#">1213</a>	..... <a href="#">574</a> , <a href="#">806</a> , <a href="#">1101</a>
<code>\__draw_backend_color_gray_aux:n</code>	<code>\__draw_backend_path:n</code> ..... <a href="#">1105</a>
..... <a href="#">1236</a> , <a href="#">1240</a>	<code>\__draw_backend_rectangle:nnnn</code> ..
<code>\__draw_backend_color_reset: ..</code> <a href="#">866</a>	..... <a href="#">539</a> , <a href="#">777</a> , <a href="#">1060</a>
<code>\__draw_backend_color_select:n</code> . <a href="#">866</a>	<code>\__draw_backend_scope:n</code> .....
<code>\__draw_backend_color_stroke:n</code> . <a href="#">691</a>	... <a href="#">1030</a> , <a href="#">1034</a> , <a href="#">1102</a> , <a href="#">1104</a> , <a href="#">1124</a> ,
<code>\__draw_backend_color_stroke_-</code>	<a href="#">1164</a> , <a href="#">1186</a> , <a href="#">1198</a> , <a href="#">1200</a> , <a href="#">1202</a> , <a href="#">1204</a> ,
<code>cmyk:nnnn</code> ..... <a href="#">691</a> , <a href="#">866</a> , <a href="#">1213</a>	<a href="#">1206</a> , <a href="#">1208</a> , <a href="#">1210</a> , <a href="#">1212</a> , <a href="#">1256</a> , <a href="#">1276</a>
<code>\__draw_backend_color_stroke_-</code>	<code>\__draw_backend_scope_begin: ...</code>
<code>gray:n</code> ..... <a href="#">691</a> , <a href="#">866</a> , <a href="#">1213</a>	..... <a href="#">535</a> , <a href="#">772</a> , <a href="#">775</a> , <a href="#">1029</a> , <a href="#">1034</a>
<code>\__draw_backend_color_stroke_-</code>	<code>\__draw_backend_scope_end: .....</code>
<code>rgb:nnn</code> ..... <a href="#">691</a> , <a href="#">866</a> , <a href="#">1213</a>	..... <a href="#">535</a> , <a href="#">774</a> , <a href="#">775</a> , <a href="#">1033</a> , <a href="#">1034</a>
<code>\__draw_backend_curveto:nnnnnn</code> ..	<code>\__draw_backend_select:n</code> .....
..... <a href="#">539</a> , <a href="#">777</a> , <a href="#">1060</a>	..... <a href="#">1225</a> , <a href="#">1243</a> , <a href="#">1271</a>
<code>\__draw_backend_dash:n</code> <a href="#">659</a> , <a href="#">834</a> , <a href="#">1173</a>	<code>\__draw_backend_stroke: 579, 811, 1105</code>
<code>\__draw_backend_dash_aux:nn</code> .. <a href="#">1173</a>	<code>\g__draw_clip_path_int</code> .....
<code>\__draw_backend_dash_pattern:nn</code> .	... <a href="#">1111</a> , <a href="#">1114</a> , <a href="#">1127</a> , <a href="#">1156</a> , <a href="#">1159</a> , <a href="#">1167</a>
..... <a href="#">659</a> , <a href="#">834</a> , <a href="#">1173</a>	<code>\__draw_color_reset: .....</code> <a href="#">732</a>
<code>\__draw_backend_discardpath: ...</code>	<code>\g__draw_draw_clip_bool</code> ... <a href="#">579</a> , <a href="#">1105</a>
..... <a href="#">579</a> , <a href="#">811</a> , <a href="#">1105</a>	<code>\g__draw_draw_eor_bool</code> .....
<code>\__draw_backend_end: .</code> <a href="#">524</a> , <a href="#">771</a> , <a href="#">1027</a>	..... <a href="#">574</a> , <a href="#">588</a> , <a href="#">606</a> ,
<code>\__draw_backend_evenodd_rule: ...</code>	<a href="#">614</a> , <a href="#">627</a> , <a href="#">636</a> , <a href="#">652</a> , <a href="#">806</a> , <a href="#">820</a> , <a href="#">825</a> , <a href="#">830</a>
..... <a href="#">574</a> , <a href="#">806</a> , <a href="#">1101</a>	<code>\g__draw_draw_path_int</code> ..... <a href="#">1105</a>
<code>\__draw_backend_fill: 579, 811, 1105</code>	<code>\g__draw_draw_path_tl</code> .....
<code>\__draw_backend_fillstroke: ....</code>	... <a href="#">1060</a> , <a href="#">1116</a> , <a href="#">1132</a> , <a href="#">1134</a> , <a href="#">1161</a> , <a href="#">1170</a>
..... <a href="#">579</a> , <a href="#">811</a> , <a href="#">1105</a>	<code>\g__draw_draw_scope_int</code> ..... <a href="#">1034</a>
<code>\__draw_backend_join_bevel: ....</code>	<code>\l__draw_draw_scope_int</code> ..... <a href="#">1034</a>
..... <a href="#">659</a> , <a href="#">834</a> , <a href="#">1173</a>	<code>\g__draw_path_int</code> ..... <a href="#">1120</a> , <a href="#">1137</a>
<code>\__draw_backend_join_miter: ....</code>	
..... <a href="#">659</a> , <a href="#">834</a> , <a href="#">1173</a>	
<code>\__draw_backend_join_round: ....</code>	
..... <a href="#">659</a> , <a href="#">834</a> , <a href="#">1173</a>	
<code>\__draw_backend_lineto:nn</code> .....	
..... <a href="#">539</a> , <a href="#">777</a> , <a href="#">1060</a>	
<code>\__draw_backend_linewidth:n</code> ....	
..... <a href="#">659</a> , <a href="#">834</a> , <a href="#">1173</a>	
<code>\__draw_backend_literal:n</code> <a href="#">522</a> , <a href="#">527</a> ,	
<a href="#">528</a> , <a href="#">532</a> , <a href="#">536</a> , <a href="#">538</a> , <a href="#">541</a> , <a href="#">549</a> , <a href="#">557</a> ,	
<a href="#">566</a> , <a href="#">580</a> , <a href="#">583</a> , <a href="#">586</a> , <a href="#">592</a> , <a href="#">602</a> , <a href="#">603</a> ,	
<a href="#">604</a> , <a href="#">609</a> , <a href="#">612</a> , <a href="#">618</a> , <a href="#">623</a> , <a href="#">624</a> , <a href="#">625</a> ,	
<a href="#">630</a> , <a href="#">631</a> , <a href="#">634</a> , <a href="#">640</a> , <a href="#">650</a> , <a href="#">656</a> , <a href="#">661</a> ,	
<a href="#">674</a> , <a href="#">678</a> , <a href="#">680</a> , <a href="#">682</a> , <a href="#">684</a> , <a href="#">686</a> , <a href="#">688</a> ,	
<a href="#">690</a> , <a href="#">737</a> , <a href="#">749</a> , <a href="#">750</a> , <a href="#">751</a> , <a href="#">752</a> , <a href="#">753</a> ,	
<a href="#">754</a> , <a href="#">758</a> , <a href="#">759</a> , <a href="#">761</a> , <a href="#">762</a> , <a href="#">763</a> , <a href="#">764</a> ,	
<a href="#">765</a> , <a href="#">769</a> , <a href="#">779</a> , <a href="#">784</a> , <a href="#">789</a> , <a href="#">799</a> , <a href="#">812</a> ,	
<a href="#">814</a> , <a href="#">816</a> , <a href="#">819</a> , <a href="#">824</a> , <a href="#">829</a> , <a href="#">833</a> , <a href="#">836</a> ,	
<a href="#">849</a> , <a href="#">853</a> , <a href="#">855</a> , <a href="#">857</a> , <a href="#">859</a> , <a href="#">861</a> , <a href="#">863</a> ,	
<a href="#">865</a> , <a href="#">1025</a> , <a href="#">1046</a> , <a href="#">1054</a> , <a href="#">1112</a> , <a href="#">1131</a> , <a href="#">1157</a>	
<code>\__draw_backend_miterlimit:n</code> ...	
..... <a href="#">659</a> , <a href="#">834</a> , <a href="#">1173</a>	
<code>\__draw_backend_moveto:nn</code> .....	
..... <a href="#">539</a> , <a href="#">777</a> , <a href="#">1060</a>	
	<b>E</b>
	<code>\endlandscape</code> ..... <a href="#">2399</a>
	<code>\evensidemargin</code> ..... <a href="#">1835</a>
	exp commands:
	<code>\exp_after:wN</code> ..... <a href="#">385</a> , <a href="#">1599</a>
	<code>\exp_args:Nf</code> ..... <a href="#">664</a> , <a href="#">839</a> , <a href="#">1753</a>
	<code>\exp_args:NNf</code> ..... <a href="#">149</a> , <a href="#">197</a> , <a href="#">254</a>
	<code>\exp_args:Nnx</code> ..... <a href="#">1740</a> , <a href="#">2382</a>
	<code>\exp_args:NV</code> ..... <a href="#">380</a>
	<code>\exp_args:Nx</code> .....
	... <a href="#">484</a> , <a href="#">1411</a> , <a href="#">1432</a> , <a href="#">1711</a> , <a href="#">1827</a> , <a href="#">2348</a>
	<code>\exp_last_unbraced:Nx</code> ..... <a href="#">389</a> , <a href="#">437</a>
	<code>\exp_not:N</code> ..... <a href="#">67</a> , <a href="#">76</a> , <a href="#">95</a> , <a href="#">506</a> ,
	<a href="#">507</a> , <a href="#">515</a> , <a href="#">904</a> , <a href="#">905</a> , <a href="#">912</a> , <a href="#">2057</a> , <a href="#">2058</a> ,
	<a href="#">2059</a> , <a href="#">2064</a> , <a href="#">2066</a> , <a href="#">2067</a> , <a href="#">2090</a> , <a href="#">2092</a> ,
	<a href="#">2093</a> , <a href="#">2099</a> , <a href="#">2100</a> , <a href="#">2101</a> , <a href="#">2106</a> , <a href="#">2107</a> ,
	<a href="#">2109</a> , <a href="#">2119</a> , <a href="#">2127</a> , <a href="#">2128</a> , <a href="#">2131</a> , <a href="#">2132</a> ,
	<a href="#">2133</a> , <a href="#">2159</a> , <a href="#">2161</a> , <a href="#">2168</a> , <a href="#">2238</a> , <a href="#">2240</a> ,
	<a href="#">2241</a> , <a href="#">2246</a> , <a href="#">2248</a> , <a href="#">2249</a> , <a href="#">2254</a> , <a href="#">2258</a> ,
	<a href="#">2268</a> , <a href="#">2280</a> , <a href="#">2281</a> , <a href="#">2287</a> , <a href="#">2288</a> , <a href="#">2294</a> ,
	<a href="#">2296</a> , <a href="#">2298</a> , <a href="#">2305</a> , <a href="#">2310</a> , <a href="#">2316</a> , <a href="#">2318</a>
	<code>\exp_not:n</code> ... <a href="#">27</a> , <a href="#">67</a> , <a href="#">76</a> , <a href="#">95</a> , <a href="#">1702</a> ,
	<a href="#">1707</a> , <a href="#">1990</a> , <a href="#">2185</a> , <a href="#">2186</a> , <a href="#">2205</a> , <a href="#">2206</a> ,
	<a href="#">2216</a> , <a href="#">2217</a> , <a href="#">2360</a> , <a href="#">2365</a> , <a href="#">2376</a> , <a href="#">2459</a>

## F

file commands:

\file\_compare\_timestamp:nNnTF . 1420  
 \file\_parse\_full\_name:nNNN 1407, 1430

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, 1987, 2119, 2472  
 \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 ..... 1899  
 \l\_galley\_total\_left\_margin\_dim 1830

graphics commands:

\graphics\_bb\_restore:nTF . 1362, 1582  
 \graphics\_bb\_save:n ..... 1391, 1590  
 \l\_graphics\_decodearray\_tl .....  
 ..... 1339, 1340,  
 1350, 1370, 1374, 1375, 1458, 1490,  
 1491, 1529, 1532, 1533, 1551, 1621  
 \graphics\_extract\_bb:n .....  
 ..... 1453, 1460, 1616, 1623  
 \l\_graphics\_interpolate\_bool ...  
 ..... 1341, 1351, 1369, 1376,  
 1459, 1492, 1528, 1534, 1552, 1622  
 \l\_graphics\_llx\_dim .....  
 ..... 1324, 1469, 1521, 1634  
 \l\_graphics\_lly\_dim .....  
 ..... 1325, 1470, 1522, 1635  
 \l\_graphics\_name\_tl ..... 1425  
 \l\_graphics\_page\_int .....  
 ..... 1335, 1355, 1356, 1380,  
 1381, 1451, 1488, 1489, 1515, 1516,  
 1544, 1557, 1558, 1597, 1598, 1614  
 \l\_graphics\_pagebox\_tl .....  
 ..... 41, 1336, 1354,

1382, 1383, 1452, 1486, 1487, 1517,  
 1519, 1545, 1566, 1567, 1599, 1615  
 \graphics\_read\_bb:n . 1318, 1447, 1611  
 \l\_graphics\_urx\_dim .....  
 .. 1326, 1387, 1471, 1523, 1588, 1636  
 \l\_graphics\_ury\_dim .. 1327, 1388,  
 1472, 1524, 1589, 1637, 1645, 1646

graphics internal commands:

\l\_\_graphics\_backend\_dir\_str . 1400  
 \l\_\_graphics\_backend\_ext\_str . 1400  
 \\_\_graphics\_backend\_getbb\_auxi:n  
 ..... 1333  
 \\_\_graphics\_backend\_getbb\_-  
 auxi:nN ..... 1542  
 \\_\_graphics\_backend\_getbb\_-  
 auxii:n ..... 1333  
 \\_\_graphics\_backend\_getbb\_-  
 auxii:nnN ..... 1542  
 \\_\_graphics\_backend\_getbb\_-  
 auxiii:nNnn ..... 1542  
 \\_\_graphics\_backend\_getbb\_-  
 auxiv:nnNnn ..... 1542  
 \\_\_graphics\_backend\_getbb\_-  
 auxv:nNnn ..... 1542  
 \\_\_graphics\_backend\_getbb\_-  
 auxvi:nNnn ..... 1583, 1585  
 \\_\_graphics\_backend\_getbb\_eps:n .  
 ..... 1312, 1400, 1441, 1605  
 \\_\_graphics\_backend\_getbb\_eps:nm  
 ..... 1400  
 \\_\_graphics\_backend\_getbb\_eps:nn  
 ..... 1411, 1418  
 \\_\_graphics\_backend\_getbb\_jpg:n .  
 ..... 1333, 1441, 1542, 1612  
 \\_\_graphics\_backend\_getbb\_-  
 pagebox:w ..... 1542, 1599  
 \\_\_graphics\_backend\_getbb\_pdf:n .  
 ..... 1333, 1426, 1441, 1542, 1619  
 \\_\_graphics\_backend\_getbb\_png:n .  
 ..... 1333, 1441, 1542, 1612  
 \\_\_graphics\_backend\_include:nn 1625  
 \\_\_graphics\_backend\_include\_-  
 auxi:nn ..... 1464  
 \\_\_graphics\_backend\_include\_-  
 auxii:nnn ..... 1464  
 \\_\_graphics\_backend\_include\_-  
 auxiii:nnn ..... 1464  
 \\_\_graphics\_backend\_include\_-  
 bitmap\_quote:w ..... 1593, 1640  
 \\_\_graphics\_backend\_include\_-  
 eps:n ..... 1319, 1400, 1464, 1625  
 \\_\_graphics\_backend\_include\_-  
 jpg:n ..... 1393, 1464, 1640

`\__graphics_backend_include_-pdf:n` .. [1393](#), [1432](#), [1464](#), [1593](#), [1625](#)  
`\__graphics_backend_include_pdf_-quote:w` ..... [1596](#), [1601](#)  
`\__graphics_backend_include_-png:n` ..... [1393](#), [1464](#), [1640](#)  
`\l__graphics_backend_name_str` . [1400](#)  
`\l__graphics_graphics_attr_tl` ...  
     ..... [1332](#), [1337](#),  
     [1344](#), [1352](#), [1362](#), [1389](#), [1391](#), [1396](#)  
`\l__graphics_internal_box` .....  
     .. [1385](#), [1387](#), [1388](#), [1587](#), [1588](#), [1589](#)  
`\g__graphics_track_int` .....  
     ..... [1463](#), [1509](#), [1510](#)  
group commands:  
`\group_begin:` ..... [1039](#),  
     [1996](#), [2124](#), [2152](#), [2174](#), [2207](#), [2478](#)  
`\group_end:` ..... [1047](#),  
     [2016](#), [2135](#), [2171](#), [2203](#), [2234](#), [2505](#)  
`\group_insert_after:N` .....  
     ..... [422](#), [507](#), [732](#), [905](#)

## H

hbox commands:  
`\hbox:n` ..... [1759](#), [1762](#),  
     [1839](#), [1845](#), [2000](#), [2004](#), [2482](#), [2491](#)  
`\hbox_overlap_right:n` ..... [144](#),  
     [176](#), [192](#), [233](#), [249](#), [277](#), [364](#), [760](#), [1017](#)  
`\hbox_set:Nn` ..... [1385](#),  
     [1587](#), [1826](#), [1863](#), [1997](#), [2125](#), [2479](#)  
`\hbox_set:Nw` ..... [1809](#)  
`\hbox_set_end:` ..... [1824](#)  
`\hbox_unpack:N` ..... [1956](#)

## I

int commands:  
`\int_compare:nNnTF` .....  
     ..... [1355](#), [1380](#), [1488](#), [1515](#),  
     [1557](#), [1597](#), [1927](#), [2020](#), [2278](#), [2304](#)  
`\int_const:Nn` .....  
     ..... [1389](#), [1510](#), [1671](#), [2158](#), [2339](#)  
`\int_eval:n` ..... [2258](#),  
     [2281](#), [2288](#), [2298](#), [2508](#), [2516](#), [2521](#)  
`\int_gincr:N` .....  
     [284](#), [1055](#), [1111](#), [1156](#), [1509](#), [1670](#),  
     [1725](#), [1769](#), [1848](#), [2338](#), [2381](#), [2422](#)  
`\int_gset:Nn` ..... [1916](#)  
`\int_gset_eq:NN` [1048](#), [1770](#), [1849](#), [2423](#)  
`\int_gzero:N` ..... [1040](#)  
`\int_if_exist:NTF` ..... [1499](#)  
`\int_if_odd:nTF` ..... [1833](#)  
`\int_new:N` .....  
     . [334](#), [466](#), [1058](#), [1059](#), [1137](#), [1463](#),  
     [1666](#), [1750](#), [1781](#), [1783](#), [2334](#), [2404](#)

`\int_set_eq:NN` ..... [1036](#), [1928](#)  
`\int_use:N` .....  
     . [286](#), [317](#), [1114](#), [1120](#), [1127](#), [1159](#),  
     [1167](#), [1356](#), [1381](#), [1396](#), [1489](#), [1502](#),  
     [1514](#), [1516](#), [1598](#), [1677](#), [1728](#), [1741](#),  
     [1745](#), [1773](#), [1780](#), [1853](#), [1961](#), [2173](#),  
     [2180](#), [2345](#), [2383](#), [2388](#), [2426](#), [2434](#)  
`\int_value:w` .....  
     ..... [2064](#), [2090](#), [2238](#), [2246](#), [2258](#)  
`\int_zero:N` ... [1335](#), [1451](#), [1544](#), [1614](#)

## K

kernel internal commands:

`\__kernel_backend_align_begin:` ..  
     ..... [45](#), [129](#), [153](#), [168](#)  
`\__kernel_backend_align_end:` ...  
     ..... [45](#), [143](#), [161](#), [175](#)  
`\g__kernel_backend_header_bool` ..  
     ..... [41](#), [42](#)  
`\__kernel_backend_literal:n` ....  
     ..... [25](#), [31](#),  
     [34](#), [43](#), [47](#), [54](#), [57](#), [59](#), [101](#), [104](#), [106](#),  
     [108](#), [112](#), [258](#), [271](#), [418](#), [426](#), [526](#),  
     [533](#), [731](#), [936](#), [943](#), [949](#), [1009](#), [1019](#),  
     [1321](#), [1466](#), [1501](#), [1511](#), [1631](#), [1642](#),  
     [2328](#), [2454](#), [2508](#), [2512](#), [2517](#), [2522](#)  
`\__kernel_backend_literal_page:n`  
     ..... [70](#), [103](#), [2322](#), [2324](#), [2527](#), [2529](#)  
`\__kernel_backend_literal_pdf:n` .  
     ..... [62](#), [100](#), [184](#), [241](#), [769](#), [916](#)  
`\__kernel_backend_literal_-postscript:n` .....  
     ..... [30](#), [48](#), [49](#), [53](#), [130](#), [131](#),  
     [133](#), [134](#), [142](#), [154](#), [169](#), [522](#), [2022](#), [2034](#)  
`\__kernel_backend_literal_svg:n` .  
     ..... [111](#), [115](#), [117](#),  
     [119](#), [285](#), [287](#), [304](#), [1025](#), [1293](#), [1304](#)  
`\__kernel_backend_matrix:n` .....  
     ..... [90](#), [206](#), [227](#), [922](#)  
`\__kernel_backend_postscript:n` ..  
     ..... [33](#), [420](#),  
     [725](#), [1660](#), [1716](#), [1759](#), [1765](#), [1802](#),  
     [1839](#), [1846](#), [1850](#), [1864](#), [1892](#), [1942](#),  
     [1949](#), [1955](#), [1964](#), [1971](#), [2000](#), [2004](#)  
`\__kernel_backend_scope_begin:` [5](#),  
     [56](#), [78](#), [105](#), [114](#), [128](#), [152](#), [167](#), [183](#),  
     [200](#), [226](#), [240](#), [257](#), [270](#), [775](#), [1004](#), [1291](#)  
`\__kernel_backend_scope_begin:n` .  
     ..... [118](#), [306](#), [314](#), [319](#), [337](#), [350](#)  
`\__kernel_backend_scope_end:` ...  
     ..... [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> . . . . .	<code>\_pdf_backend_link_begin_-</code>
. . . . . <a href="#">466</a> , <a href="#">506</a> , <a href="#">515</a> , <a href="#">904</a> , <a href="#">912</a>	goto:nnw . . . . . <a href="#">1790</a> , <a href="#">2070</a> , <a href="#">2435</a>
<b>L</b>	<code>\_pdf_backend_link_begin_-</code>
<code>\landscape</code> . . . . . <a href="#">2395</a> , <a href="#">2397</a>	user:nnw . . . . . <a href="#">1790</a> , <a href="#">2070</a> , <a href="#">2435</a>
<b>M</b>	<code>\g_pdf_backend_link_bool</code> . . . . .
math commands:	. . . . . <a href="#">1785</a> , <a href="#">1796</a> , <a href="#">1801</a> , <a href="#">1816</a> , <a href="#">1859</a>
<code>\c_math_toggle_token</code> . . . . <a href="#">1812</a> , <a href="#">1822</a>	<code>\g_pdf_backend_link_dict_tl</code> . . . .
mode commands:	. . . . . <a href="#">1782</a> , <a href="#">1804</a> , <a href="#">1854</a>
<code>\mode_if_horizontal:TF</code> . . . <a href="#">1918</a> , <a href="#">1925</a>	<code>\_pdf_backend_link_end:</code> . . . . .
<code>\mode_if_math:TF</code> . . . . . <a href="#">1806</a>	. . . . . <a href="#">1790</a> , <a href="#">2070</a> , <a href="#">2435</a>
<b>O</b>	<code>\_pdf_backend_link_end_aux:</code> . . <a href="#">1790</a>
<code>\oddsidemargin</code> . . . . . <a href="#">1834</a>	<code>\g_pdf_backend_link_int</code> . . . . .
<b>P</b>	. . . . . <a href="#">1781</a> , <a href="#">1849</a> , <a href="#">1853</a> , <a href="#">1961</a>
pdf internal commands:	<code>\_pdf_backend_link_last:</code> . . . . .
<code>\_pdf_backend:n</code> . . . . . <a href="#">2327</a> ,	. . . . . <a href="#">1960</a> , <a href="#">2088</a> , <a href="#">2452</a>
<a href="#">2331</a> , <a href="#">2333</a> , <a href="#">2359</a> , <a href="#">2364</a> , <a href="#">2373</a> , <a href="#">2424</a> ,	<code>\_pdf_backend_link_margin:n</code> . . . .
<a href="#">2441</a> , <a href="#">2451</a> , <a href="#">2457</a> , <a href="#">2484</a> , <a href="#">2485</a> , <a href="#">2493</a>	. . . . . <a href="#">1962</a> , <a href="#">2096</a> , <a href="#">2453</a>
<code>\_pdf_backend_annotation:nnnn</code> . .	<code>\g_pdf_backend_link_math_bool</code> . .
. . . . . <a href="#">1751</a> , <a href="#">2052</a> , <a href="#">2405</a>	. . . . . <a href="#">1784</a> , <a href="#">1807</a> , <a href="#">1808</a> , <a href="#">1811</a> , <a href="#">1821</a>
<code>\_pdf_backend_annotation_-</code>	<code>\_pdf_backend_link_minima:</code> . . <a href="#">1790</a>
aux:nnnn . . . . . <a href="#">1753</a> , <a href="#">1756</a> , <a href="#">2405</a>	<code>\_pdf_backend_link_outterbox:n</code> <a href="#">1790</a>
<code>\g_pdf_backend_annotation_int</code> . .	<code>\g_pdf_backend_link_sf_int</code> . . . .
. . <a href="#">1750</a> , <a href="#">1770</a> , <a href="#">1780</a> , <a href="#">2404</a> , <a href="#">2423</a> , <a href="#">2434</a>	. . . . . <a href="#">1783</a> , <a href="#">1916</a> , <a href="#">1927</a> , <a href="#">1928</a>
<code>\_pdf_backend_annotation_last:</code> . .	<code>\_pdf_backend_link_sf_restore:</code> <a href="#">1790</a>
. . . . . <a href="#">1779</a> , <a href="#">2062</a> , <a href="#">2433</a>	<code>\_pdf_backend_link_sf_save:</code> . <a href="#">1790</a>
<code>\_pdf_backend_bdc:nn</code> . . . . .	<code>\l_pdf_backend_model_box</code> . <a href="#">1749</a> ,
. . . . . <a href="#">2046</a> , <a href="#">2321</a> , <a href="#">2526</a> , <a href="#">2548</a>	<a href="#">1826</a> , <a href="#">1863</a> , <a href="#">1871</a> , <a href="#">1882</a> , <a href="#">1897</a> , <a href="#">1904</a>
<code>\_pdf_backend_catalog_gput:nn</code> . .	<code>\_pdf_backend_objcompresslevel:n</code>
. . . . . <a href="#">1662</a> , <a href="#">2137</a> , <a href="#">2330</a> , <a href="#">2532</a>	. . . . . <a href="#">2252</a>
<code>\_pdf_backend_compress_objects:n</code>	<code>\g_pdf_backend_object_int</code> . . . .
. . . . . <a href="#">2018</a> , <a href="#">2252</a> , <a href="#">2507</a> , <a href="#">2542</a>	. . . . . <a href="#">1666</a> , <a href="#">1670</a> , <a href="#">1673</a> ,
<code>\_pdf_backend_compresslevel:n</code> . .	<a href="#">1725</a> , <a href="#">1728</a> , <a href="#">1741</a> , <a href="#">1745</a> , <a href="#">1769</a> , <a href="#">1770</a> ,
. . . . . <a href="#">2018</a> , <a href="#">2252</a> , <a href="#">2507</a> , <a href="#">2542</a>	<a href="#">1773</a> , <a href="#">1848</a> , <a href="#">1849</a> , <a href="#">2334</a> , <a href="#">2338</a> , <a href="#">2341</a> ,
<code>\l_pdf_backend_content_box</code> <a href="#">1748</a> ,	<a href="#">2381</a> , <a href="#">2383</a> , <a href="#">2388</a> , <a href="#">2422</a> , <a href="#">2423</a> , <a href="#">2426</a>
<a href="#">1809</a> , <a href="#">1838</a> , <a href="#">1841</a> , <a href="#">1843</a> , <a href="#">1872</a> , <a href="#">1883</a>	<code>\_pdf_backend_object_last:</code> . . . .
<code>\_pdf_backend_destination:nn</code> . . .	. . . . . <a href="#">1744</a> , <a href="#">2236</a> , <a href="#">2387</a> , <a href="#">2534</a>
. . . . . <a href="#">1969</a> , <a href="#">2103</a> , <a href="#">2455</a>	<code>\_pdf_backend_object_new:nn</code> . . .
<code>\_pdf_backend_destination_-</code>	. . . . . <a href="#">1668</a> , <a href="#">2152</a> , <a href="#">2336</a> , <a href="#">2534</a>
box:nn . . . . . <a href="#">1969</a> , <a href="#">2103</a> , <a href="#">2455</a>	<code>\_pdf_backend_object_now:nn</code> . . .
<code>\_pdf_backend_emc:</code> . . . . .	. . . . . <a href="#">1723</a> , <a href="#">2207</a> , <a href="#">2379</a> , <a href="#">2534</a>
. . . . . <a href="#">2046</a> , <a href="#">2321</a> , <a href="#">2526</a> , <a href="#">2548</a>	<code>\g_pdf_backend_object_prop</code> . . . .
<code>\_pdf_backend_info_gput:nn</code> . . . .	. . . . . <a href="#">1666</a> , <a href="#">1674</a> , <a href="#">1685</a> , <a href="#">1695</a> ,
. . . . . <a href="#">1662</a> , <a href="#">2137</a> , <a href="#">2330</a> , <a href="#">2532</a>	<a href="#">2151</a> , <a href="#">2161</a> , <a href="#">2183</a> , <a href="#">2334</a> , <a href="#">2342</a> , <a href="#">2349</a>
<code>\_pdf_backend_link:nw</code> . . . . . <a href="#">1790</a>	<code>\_pdf_backend_object_ref:n</code> <a href="#">1668</a> ,
<code>\_pdf_backend_link_aux:nw</code> . . . <a href="#">1790</a>	<a href="#">1682</a> , <a href="#">1696</a> , <a href="#">2152</a> , <a href="#">2336</a> , <a href="#">2355</a> , <a href="#">2534</a>
<code>\_pdf_backend_link_begin:n</code> . . <a href="#">2435</a>	<code>\_pdf_backend_object_write:nn</code> . .
<code>\_pdf_backend_link_begin:nnnw</code> <a href="#">2070</a>	. . . . . <a href="#">1678</a> , <a href="#">2174</a> , <a href="#">2346</a> , <a href="#">2534</a>
<code>\_pdf_backend_link_begin:nw</code> . . .	<code>\_pdf_backend_object_write:nnn</code> <a href="#">2346</a>
. . . . . <a href="#">1791</a> , <a href="#">1793</a> , <a href="#">1794</a>	<code>\_pdf_backend_object_write_-</code>
<code>\_pdf_backend_link_begin_aux:nw</code>	array:nn . . . . . <a href="#">1678</a> , <a href="#">2346</a>
. . . . . <a href="#">1797</a> , <a href="#">1799</a>	<code>\_pdf_backend_object_write_-</code>
	dict:nn . . . . . <a href="#">1678</a> , <a href="#">2346</a>
	<code>\_pdf_backend_object_write_-</code>
	fstream:nn . . . . . <a href="#">2346</a>



