

File I

Implementation

1 l3backend-basics Implementation

```
1 <*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 \ProvidesExplFile
3 <*dvipdfmx>
4   {l3backend-dvipdfmx.def}{2022-09-28}{}
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2022-09-28}{}
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2022-09-28}{}
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2022-09-28}{}
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2022-09-28}{}
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2022-09-28}{}
25   {L3 backend support: XeTeX}
26 </xetex>
```

Check if the loaded kernel is at least enough to load this file. The kernel date has to be at least equal to `\ExplBackendFileDate` or later. If `__kernel_dependency_version_check:Nn` doesn't exist we're loading in an older kernel, so it's an error anyway. With time, this test should vanish and only the dependency check should remain.

```
27 \cs_if_exist:NTF \__kernel_dependency_version_check:nn
28   {
29     \__kernel_dependency_version_check:nn {2021-02-18}
30 <dvipdfmx>   {l3backend-dvipdfmx.def}
31 <dvips>     {l3backend-dvips.def}
32 <dvisvgm>   {l3backend-dvisvgm.def}
33 <luatex>    {l3backend-luatex.def}
34 <pdftex>    {l3backend-pdftex.def}
35 <xetex>     {l3backend-xetex.def}
```

```

36 }
37 {
38   \cs_if_exist_use:cF { @latex@error } { \errmessage }
39   {
40     Mismatched-LaTeX-support-files~detected. \MessageBreak
41     Loading~aborted!
42   }
43   { \use:c { @ehd } }
44   \tex_endinput:D
45 }

```

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 LuaTeX/pdfTeX-like.
- LuaTeX/pdfTeX and dvipdfmx/X_YTeX share drawing routines.
- X_YTeX is the same as dvipdfmx other than image size extraction so takes most of the same code.

`__kernel_backend_literal:e` 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.

```

__kernel_backend_literal:n
__kernel_backend_literal:x
46 \cs_new_eq:NN __kernel_backend_literal:e \tex_special:D
47 \cs_new_protected:Npn __kernel_backend_literal:n #1
48   { __kernel_backend_literal:e { \exp_not:n {#1} } }
49 \cs_generate_variant:Nn __kernel_backend_literal:n { x }

```

(End definition for `__kernel_backend_literal:e`.)

`__kernel_backend_first_shipout:n` We need to write at first shipout in a few places. As we want to use the most up-to-date method,

```

50 \cs_if_exist:NTF \@ifl@t@r
51   {
52     \@ifl@t@r \fmtversion { 2020-10-01 }
53     {
54       \cs_new_protected:Npn __kernel_backend_first_shipout:n #1
55         { \hook_gput_code:nnn { shipout / firstpage } { l3backend } {#1} }
56     }
57     { \cs_new_eq:NN __kernel_backend_first_shipout:n \AtBeginDvi }
58   }
59   { \cs_new_eq:NN __kernel_backend_first_shipout:n \use:n }

```

(End definition for `__kernel_backend_first_shipout:n`.)

1.1 dvips backend

```

60 <*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.

```

61 \cs_new_protected:Npn __kernel_backend_literal_postscript:n #1
62   { __kernel_backend_literal:n { ps:: #1 } }
63 \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 `"`).

```
64 \cs_new_protected:Npn \_kernel_backend_postscript:n #1
65   { \_kernel_backend_literal:n { ps: SDict ~ begin ~ #1 ~ end } }
66 \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.

```
67 \bool_if:NT \g__kernel_backend_header_bool
68   {
69     \_kernel_backend_first_shipout:n
70     { \_kernel_backend_literal:n { header = l3backend-dvips.pro } }
71   }
```

`_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.

```
72 \cs_new_protected:Npn \_kernel_backend_align_begin:
73   {
74     \_kernel_backend_literal:n { ps::[begin] }
75     \_kernel_backend_literal_postscript:n { currentpoint }
76     \_kernel_backend_literal_postscript:n { currentpoint~translate }
77   }
78 \cs_new_protected:Npn \_kernel_backend_align_end:
79   {
80     \_kernel_backend_literal_postscript:n { neg-exch~neg-exch~translate }
81     \_kernel_backend_literal:n { ps::[end] }
82   }
```

(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.

```
83 \cs_new_protected:Npn \_kernel_backend_scope_begin:
84   { \_kernel_backend_literal:n { ps:gsave } }
85 \cs_new_protected:Npn \_kernel_backend_scope_end:
86   { \_kernel_backend_literal:n { ps:grestore } }
```

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

```
87 </dvips>
```

1.2 LuaTeX and pdfTeX backends

88 `<*luatex | pdftex>`

Both LuaTeX and pdfTeX write PDFs directly rather than via an intermediate file. Although there are similarities, the move of LuaTeX to have more code in Lua means we create two independent files using shared DocStrip code.

`_kernel_backend_literal_pdf:n`
`_kernel_backend_literal_pdf:x`

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).

```
89 \cs_new_protected:Npn \_kernel_backend_literal_pdf:n #1
90 {
91 <*luatex>
92   \tex_pdfextension:D literal
93 </luatex>
94 <*pdftex>
95   \tex_pdfliteral:D
96 </pdftex>
97   { \exp_not:n {#1} }
98 }
99 \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.

```
100 \cs_new_protected:Npn \_kernel_backend_literal_page:n #1
101 {
102 <*luatex>
103   \tex_pdfextension:D literal ~
104 </luatex>
105 <*pdftex>
106   \tex_pdfliteral:D
107 </pdftex>
108   page { \exp_not:n {#1} }
109 }
```

(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:`

```
110 \cs_new_protected:Npn \_kernel_backend_scope_begin:
111 {
112 <*luatex>
113   \tex_pdfextension:D save \scan_stop:
114 </luatex>
115 <*pdftex>
116   \tex_pdfsave:D
117 </pdftex>
118 }
119 \cs_new_protected:Npn \_kernel_backend_scope_end:
120 {
121 <*luatex>
122   \tex_pdfextension:D restore \scan_stop:
123 </luatex>
124 <*pdftex>
125   \tex_pdfrestore:D
```

```

126 </pdftex>
127 }

```

(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.

```

128 \cs_new_protected:Npn \_kernel_backend_matrix:n #1
129 {
130 <*luatex>
131 \tex_pdfextension:D setmatrix
132 </luatex>
133 <*pdftex>
134 \tex_pdfsetmatrix:D
135 </pdftex>
136 { \exp_not:n {#1} }
137 }
138 \cs_generate_variant:Nn \_kernel_backend_matrix:n { x }

```

(End definition for `_kernel_backend_matrix:n.`)

```

139 </luatex | pdftex>

```

1.3 dvipdfmx backend

```

140 <*dvipdfmx | xetex>

```

The `dvipdfmx` shares code with the PDF mode one (using the common section to this file) but also with `XYTeX`. 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 `XYTeX` as required.

`_kernel_backend_literal_pdf:n` Undocumented but equivalent to `pdfTeX`'s `literal` keyword. It's similar to be not the same as the documented `contents` keyword as that adds a `q/Q` pair.

```

141 \cs_new_protected:Npn \_kernel_backend_literal_pdf:n #1
142 { \_kernel_backend_literal:n { pdf:literal~ #1 } }
143 \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!

```

144 \cs_new_protected:Npn \_kernel_backend_literal_page:n #1
145 { \_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. We use the versions originally from `xdvipfmx (x:)` as these are well-tested “in the wild”.

```

146 \cs_new_protected:Npn \_kernel_backend_scope_begin:
147 { \_kernel_backend_literal:n { x:gsave } }
148 \cs_new_protected:Npn \_kernel_backend_scope_end:
149 { \_kernel_backend_literal:n { x:grestore } }

```

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

```

150 </dvipdfmx | xetex>

```

1.4 dvisvgm backend

151 `*dvisvgm)`

`_kernel_backend_literal_svg:n`
`_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.

152 `\cs_new_protected:Npn _kernel_backend_literal_svg:n #1`
 153 `{ _kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }`
 154 `\cs_generate_variant:Nn _kernel_backend_literal_svg:n { x }`

(End definition for `_kernel_backend_literal_svg:n`.)

`\g_kernel_backend_scope_int`
`\l_kernel_backend_scope_int`

In SVG, we need to track scope nesting as properties attach to scopes; that requires a pair of `int` registers.

155 `\int_new:N \g_kernel_backend_scope_int`
 156 `\int_new:N \l_kernel_backend_scope_int`

(End definition for `\g_kernel_backend_scope_int` and `\l_kernel_backend_scope_int`.)

`_kernel_backend_scope_begin:`
`_kernel_backend_scope_end:`
`_kernel_backend_scope_begin:n`
`_kernel_backend_scope_begin:x`
`_kernel_backend_scope:n`
`_kernel_backend_scope:x`

In SVG, the need to attach concepts to a scope means we need to be sure we will close all of the open scopes. That is easiest done if we only need an outer “wrapper” `begin/end` pair, and within that we apply operations as a simple scoped statements. To keep down the non-productive groups, we also have a `begin` version that does take an argument.

157 `\cs_new_protected:Npn _kernel_backend_scope_begin:`
 158 `{`
 159 `_kernel_backend_literal_svg:n { <g> }`
 160 `\int_set_eq:NN`
 161 `\l_kernel_backend_scope_int`
 162 `\g_kernel_backend_scope_int`
 163 `\group_begin:`
 164 `\int_gset:Nn \g_kernel_backend_scope_int { 1 }`
 165 `}`
 166 `\cs_new_protected:Npn _kernel_backend_scope_end:`
 167 `{`
 168 `\prg_replicate:nn`
 169 `{ \g_kernel_backend_scope_int }`
 170 `{ _kernel_backend_literal_svg:n { </g> } }`
 171 `\group_end:`
 172 `\int_gset_eq:NN`
 173 `\g_kernel_backend_scope_int`
 174 `\l_kernel_backend_scope_int`
 175 `}`
 176 `\cs_new_protected:Npn _kernel_backend_scope_begin:n #1`
 177 `{`
 178 `_kernel_backend_literal_svg:n { <g ~ #1 > }`
 179 `\int_set_eq:NN`
 180 `\l_kernel_backend_scope_int`
 181 `\g_kernel_backend_scope_int`
 182 `\group_begin:`
 183 `\int_gset:Nn \g_kernel_backend_scope_int { 1 }`
 184 `}`
 185 `\cs_generate_variant:Nn _kernel_backend_scope_begin:n { x }`

```

186 \cs_new_protected:Npn \__kernel_backend_scope:n #1
187 {
188   \__kernel_backend_literal_svg:n { <g ~ #1 > }
189   \int_gincr:N \g__kernel_backend_scope_int
190 }
191 \cs_generate_variant:Nn \__kernel_backend_scope:n { x }

```

(End definition for __kernel_backend_scope_begin: and others.)

```

192 </dvisvgm>
193 </package>

```

2 I3backend-box Implementation

```

194 <*package>
195 <@@=box>

```

2.1 dvips backend

```

196 <*dvips>

```

__box_backend_clip:N The `dvips` backend scales all absolute dimensions based on the output resolution selected and any `TeX` 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.

```

197 \cs_new_protected:Npn \__box_backend_clip:N #1
198 {
199   \__kernel_backend_scope_begin:
200   \__kernel_backend_align_begin:
201   \__kernel_backend_literal_postscript:n { matrix-currentmatrix }
202   \__kernel_backend_literal_postscript:n
203     { Resolution~72~div~VResolution~72~div~scale }
204   \__kernel_backend_literal_postscript:n { DVImag-dup~scale }
205   \__kernel_backend_literal_postscript:x
206     {
207       0 ~
208       \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
209       \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
210       \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
211       rectclip
212     }
213   \__kernel_backend_literal_postscript:n { setmatrix }
214   \__kernel_backend_align_end:
215   \hbox_overlap_right:n { \box_use:N #1 }
216   \__kernel_backend_scope_end:
217   \skip_horizontal:n { \box_wd:N #1 }
218 }

```

(End definition for __box_backend_clip:N.)

__box_backend_rotate: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.

```

219 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
220 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
221 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
222 {
223   \__kernel_backend_scope_begin:
224   \__kernel_backend_align_begin:
225   \__kernel_backend_literal_postscript:x
226   {
227     \fp_compare:nNnTF {#2} = \c_zero_fp
228     { 0 }
229     { \fp_eval:n { round ( -(#2) , 5 ) } } } ~
230   rotate
231   }
232   \__kernel_backend_align_end:
233   \box_use:N #1
234   \__kernel_backend_scope_end:
235 }

```

(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.

```

236 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
237 {
238   \__kernel_backend_scope_begin:
239   \__kernel_backend_align_begin:
240   \__kernel_backend_literal_postscript:x
241   {
242     \fp_eval:n { round ( #2 , 5 ) } ~
243     \fp_eval:n { round ( #3 , 5 ) } ~
244     scale
245   }
246   \__kernel_backend_align_end:
247   \hbox_overlap_right:n { \box_use:N #1 }
248   \__kernel_backend_scope_end:
249 }

```

(End definition for `__box_backend_scale:Nnn`.)

250 `</dvips>`

2.2 LuaTeX and pdfTeX backends

251 `<*luatex | pdftex>`

`__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.

```

252 \cs_new_protected:Npn \__box_backend_clip:N #1
253 {
254   \__kernel_backend_scope_begin:
255   \__kernel_backend_literal_pdf:x
256   {

```

```

257     0~
258     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
259     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
260     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
261     re~W~n
262   }
263   \hbox_overlap_right:n { \box_use:N #1 }
264   \__kernel_backend_scope_end:
265   \skip_horizontal:n { \box_wd:N #1 }
266 }

```

(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.

```

267 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
268 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
269 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
270 {
271   \__kernel_backend_scope_begin:
272   \box_set_wd:Nn #1 { Opt }
273   \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
274   \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
275     { \fp_zero:N \l__box_backend_cos_fp }
276   \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
277   \__kernel_backend_matrix:x
278   {
279     \fp_use:N \l__box_backend_cos_fp \c_space_tl
280     \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp
281       { 0~0 }
282     {
283       \fp_use:N \l__box_backend_sin_fp
284       \c_space_tl
285       \fp_eval:n { -\l__box_backend_sin_fp }
286     }
287     \c_space_tl
288     \fp_use:N \l__box_backend_cos_fp
289   }
290   \box_use:N #1
291   \__kernel_backend_scope_end:
292 }
293 \fp_new:N \l__box_backend_cos_fp
294 \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.

```

295 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
296 {
297   \__kernel_backend_scope_begin:
298   \__kernel_backend_matrix:x

```

```

299     {
300       \fp_eval:n { round ( #2 , 5 ) } ~
301       0~0~
302       \fp_eval:n { round ( #3 , 5 ) }
303     }
304     \hbox_overlap_right:n { \box_use:N #1 }
305     \__kernel_backend_scope_end:
306   }

```

(End definition for __box_backend_scale:Nnn.)

```
307 </luatex | pdftex>
```

2.3 dvipdfmx/X_YTeX backend

```
308 <*dvipdfmx | xetex>
```

__box_backend_clip:N The code here is identical to that for LuaTeX/pdfTeX: unlike rotation and scaling, there is no higher-level support in the backend for clipping.

```

309 \cs_new_protected:Npn \__box_backend_clip:N #1
310 {
311   \__kernel_backend_scope_begin:
312   \__kernel_backend_literal_pdf:x
313   {
314     0~
315     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
316     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
317     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
318     re~W~n
319   }
320   \hbox_overlap_right:n { \box_use:N #1 }
321   \__kernel_backend_scope_end:
322   \skip_horizontal:n { \box_wd:N #1 }
323 }

```

(End definition for __box_backend_clip:N.)

__box_backend_rotate:Nn
 __box_backend_rotate_aux:Nn

Rotating in dvipdfmx/X_YTeX 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.

```

324 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
325 { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
326 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
327 {
328   \__kernel_backend_scope_begin:
329   \__kernel_backend_literal:x
330   {
331     x:rotate~
332     \fp_compare:nNnTF {#2} = \c_zero_fp
333     { 0 }
334     { \fp_eval:n { round ( #2 , 5 ) } } }
335   }

```

```

336     \box_use:N #1
337     \__kernel_backend_scope_end:
338 }

```

(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.

```

339 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
340 {
341     \__kernel_backend_scope_begin:
342     \__kernel_backend_literal:x
343     {
344         x:scale~
345         \fp_eval:n { round ( #2 , 5 ) } ~
346         \fp_eval:n { round ( #3 , 5 ) }
347     }
348     \hbox_overlap_right:n { \box_use:N #1 }
349     \__kernel_backend_scope_end:
350 }

```

(End definition for `__box_backend_scale:Nnn`.)

```

351 </dviPDFmx | xetex>

```

2.4 dvisvgm backend

```

352 <*dvisvgm>

```

`__box_backend_clip:N`
`\g__kernel_clip_path_int`

Clipping in SVG is more involved than with other backends. The first issue is that the clipping path must be defined separately from where it is used, so we need to track how many paths have applied. The naming here uses `l3cp` as the namespace with a number following. Rather than use a rectangular operation, we define the path manually as this allows it to have a depth: easier than the alternative approach of shifting content up and down using scopes to allow for the depth of the `TEX` box and keep the reference point the same!

```

353 \cs_new_protected:Npn \__box_backend_clip:N #1
354 {
355     \int_gincr:N \g__kernel_clip_path_int
356     \__kernel_backend_literal_svg:x
357     { < clipPath-id = " l3cp \int_use:N \g__kernel_clip_path_int " > }
358     \__kernel_backend_literal_svg:x
359     {
360         <
361         path ~ d =
362         "
363             M ~ 0 ~
364             \dim_to_decimal:n { -\box_dp:N #1 } ~
365             L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
366             \dim_to_decimal:n { -\box_dp:N #1 } ~
367             L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
368             \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
369             L ~ 0 ~
370             \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
371             Z

```

```

372         "
373         />
374     }
375     \__kernel_backend_literal_svg:n
376     { < /clipPath > }

```

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

```

377     \__kernel_backend_scope_begin:n
378     {
379         transform =
380         "
381             translate ( { ?x } , { ?y } ) ~
382             scale ( 1 , -1 )
383         "
384     }
385     \__kernel_backend_scope:x
386     {
387         clip-path =
388         "url ( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int ) "
389     }
390     \__kernel_backend_scope:n
391     {
392         transform =
393         "
394             scale ( -1 , 1 ) ~
395             translate ( { ?x } , { ?y } ) ~
396             scale ( -1 , -1 )
397         "
398     }
399     \box_use:N #1
400     \__kernel_backend_scope_end:
401 }
402 \int_new:N \g__kernel_clip_path_int

```

(End definition for $\backslash_box_backend_clip:N$ and $\backslash_g_kernel_clip_path_int$.)

$\backslash_box_backend_rotate:Nn$ 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.

```

403 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
404 {
405     \__kernel_backend_scope_begin:x
406     {
407         transform =
408         "
409             rotate
410             ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
411         "
412     }
413     \box_use:N #1

```

```

414     \__kernel_backend_scope_end:
415 }

```

(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).

```

416 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
417 {
418   \__kernel_backend_scope_begin:x
419   {
420     transform =
421     "
422       translate ( { ?x } , { ?y } ) ~
423       scale
424       (
425         \fp_eval:n { round ( -#2 , 5 ) } ,
426         \fp_eval:n { round ( -#3 , 5 ) }
427       ) ~
428       translate ( { ?x } , { ?y } ) ~
429       scale ( -1 )
430     "
431   }
432   \hbox_overlap_right:n { \box_use:N #1 }
433   \__kernel_backend_scope_end:
434 }

```

(End definition for __box_backend_scale:Nnn.)

```

435 </dvisvgm>

```

```

436 </package>

```

3 13backend-color Implementation

```

437 <*package>

```

```

438 <@@=color>

```

Color support is split into parts: collecting data from L^AT_EX 2_ε, the color stack, general color, separations, and color for drawings. We have different approaches in each backend, and have some choices to make about dvipdfmx/X_YL_TE_X in particular. Whilst it is in some ways convenient to use the same approach in multiple backends, the fact that dvipdfmx/X_YL_TE_X is PDF-based means it (largely) sticks closer to direct PDF output.

3.1 Collecting information from L^AT_EX 2_ε

3.1.1 dvips-style

```

439 <*dvisvgm | dvipdfmx | dvips | xetex>

```

__color_backend_pickup:N Allow for L^AT_EX 2_ε color. Here, the possible input values are limited: dvips-style colors can be taken as-is. The x-type expansion is there to cover the case where xcolor is in use.

```

440 \cs_new_protected:Npn \__color_backend_pickup:N #1
441 {
442   \exp_args:NV \tl_if_head_is_space:nTF \current@color

```

```

443     {
444       \tl_set:Nn #1 { { gray } { 0 } }
445       \msg_warning:nx { color } { unhandled }
446       { \current@color }
447     }
448     {
449       \exp_last_unbraced:Nx \_color_backend_pickup:w
450       { \current@color } \s__color_stop #1
451     }
452   }
453   \cs_new_protected:Npn \_color_backend_pickup:w #1 ~ #2 \s__color_stop #3
454   { \tl_set:Nn #3 { {#1} {#2} } }
(End definition for \_color_backend_pickup:N and \_color_backend_pickup:w.)
455 </dvisvgm | dvipdfmx | dvips | xetex>

```

3.1.2 LuaTeX and pdfTeX

```

456 <*luatex | pdftex>

```

Same ideas, but with a different backend-dependent format.

```

\_color_backend_pickup:N
\_color_backend_pickup:w

```

```

457 \cs_new_protected:Npn \_color_backend_pickup:N #1
458 {
459   \exp_last_unbraced:Nx \_color_backend_pickup:w
460   { \current@color } ~ 0 ~ 0 ~ 0 \s__color_stop #1
461 }
462 \cs_new_protected:Npn \_color_backend_pickup:w
463 #1 ~ #2 ~ #3 ~ #4 ~ #5 ~ #6 \s__color_stop #7
464 {
465   \str_if_eq:nnTF {#2} { g }
466   { \tl_set:Nn #7 { { gray } {#1} } }
467   {
468     \str_if_eq:nnTF {#4} { rg }
469     { \tl_set:Nn #7 { { rgb } { #1 ~ #2 ~ #3 } } }
470     {
471       \str_if_eq:nnTF {#5} { k }
472       { \tl_set:Nn #7 { { cmyk } { #1 ~ #2 ~ #3 ~ #4 } } }
473       {
474         \tl_set:Nn #1 { { gray } { 0 } }
475         \msg_warning:nx { color } { unhandled }
476         { \current@color }
477       }
478     }
479   }
480 }
(End definition for \_color_backend_pickup:N and \_color_backend_pickup:w.)
481 </luatex | pdftex>

```

3.2 The color stack

For PDF-based engines, we have a color stack available inside the specials. This is used for concepts beyond color itself: it is needed to manage the graphics state generally. Although dvipdfmx/X_YTeX have multiple color stacks in recent releases, the way these

interact with the original single stack and with other graphic state operations means that currently it is not feasible to use the multiple stacks.

3.2.1 Common code

```
482 <*luatex | pdftex>
```

`\l__color_backend_stack_int` For tracking which stack is in use where multiple stacks are used: currently just pdfTeX/LuaTeX but at some future stage may also cover dvipdfmx/X_YTeX.

```
483 \int_new:N \l__color_backend_stack_int
```

(End definition for `\l__color_backend_stack_int`.)

```
484 </luatex | pdftex>
```

3.2.2 LuaTeX and pdfTeX

```
485 <*luatex | pdftex>
```

`__kernel_color_backend_stack_init:Nnn`

```
486 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
487 {
488   \int_const:Nn #1
489   {
490     <*luatex>
491     \tex_pdffeedback:D colorstackinit ~
492     </luatex>
493     <*pdftex>
494     \tex_pdfcolorstackinit:D
495     </pdftex>
496     \tl_if_blank:nF {#2} { #2 ~ }
497     {#3}
498   }
499 }
```

(End definition for `__kernel_color_backend_stack_init:Nnn`.)

`__kernel_color_backend_stack_push:nn`

`__kernel_color_backend_stack_pop:n`

```
500 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
501 {
502   <*luatex>
503   \tex_pdfextension:D colorstack ~
504   </luatex>
505   <*pdftex>
506   \tex_pdfcolorstack:D
507   </pdftex>
508   \int_eval:n {#1} ~ push ~ {#2}
509 }
510 \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
511 {
512   <*luatex>
513   \tex_pdfextension:D colorstack ~
514   </luatex>
515   <*pdftex>
516   \tex_pdfcolorstack:D
517   </pdftex>
```

```

518     \int_eval:n {#1} ~ pop \scan_stop:
519   }

```

(End definition for `__kernel_color_backend_stack_push:nn` and `__kernel_color_backend_stack_pop:n`.)

```

520 </luatex | pdftex>

```

3.3 General color

3.3.1 dvips-style

```

521 < *dvips | dvisvgm>

```

Push the data to the stack. In the case of dvips also saves the drawing color in raw PostScript. The spot model is for handling data in classical format.

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_named:n
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc
522 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
523   { \__color_backend_select:n { cmyk ~ #1 } }
524 \cs_new_protected:Npn \__color_backend_select_gray:n #1
525   { \__color_backend_select:n { gray ~ #1 } }
526 \cs_new_protected:Npn \__color_backend_select_named:n #1
527   { \__color_backend_select:n { ~ #1 } }
528 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
529   { \__color_backend_select:n { rgb ~ #1 } }
530 \cs_new_protected:Npn \__color_backend_select:n #1
531   {
532     \__kernel_backend_literal:n { color~push~ #1 }
533   < *dvips>
534     \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
535   < /dvips>
536   }
537 \cs_new_protected:Npn \__color_backend_reset:
538   { \__kernel_backend_literal:n { color~pop } }

```

(End definition for `__color_backend_select_cmyk:n` and others. This function is documented on page ??.)

```

539 < /dvips | dvisvgm>

```

3.3.2 LuaTeX and pdfTeX

```

540 < *luatex | pdftex>

```

```

\l__color_backend_fill_tl
\l__color_backend_stroke_tl
541 \tl_new:N \l__color_backend_fill_tl
542 \tl_new:N \l__color_backend_stroke_tl

```

(End definition for `\l__color_backend_fill_tl` and `\l__color_backend_stroke_tl`.)

Store the values then pass to the stack.

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_select:nn
\__color_backend_reset:
543 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
544   { \__color_backend_select:nn { #1 ~ k } { #1 ~ K } }
545 \cs_new_protected:Npn \__color_backend_select_gray:n #1
546   { \__color_backend_select:nn { #1 ~ g } { #1 ~ G } }
547 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
548   { \__color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
549 \cs_new_protected:Npn \__color_backend_select:nn #1#2

```

```

550 {
551   \tl_set:Nn \l__color_backend_fill_tl {#1}
552   \tl_set:Nn \l__color_backend_stroke_tl {#2}
553   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int { #1 ~ #2 }
554 }
555 \cs_new_protected:Npn \__color_backend_reset:
556 { \__kernel_color_backend_stack_pop:n \l__color_backend_stack_int }

```

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

```
557 </luatex | pdftex>
```

3.3.3 dvipdfmx/X_qT_EX

These backends have the most possible approaches: it recognises both `dvips`-based color specials and its own format, plus one can include PDF statements directly. Recent releases also have a color stack approach similar to `pdfTEX`. Of the stack methods, the dedicated the most versatile is the latter as it can cover all of the use cases we have. However, at present this interacts problematically with any color on the original stack. We therefore stick to a single-stack approach here.

```
558 <*dvipdfmx | xetex>
```

```

\__color_backend_select:n Using the single stack is relatively easy as there is only one route.
  \__color_backend_select_cmyk:n 559 \cs_new_protected:Npn \__color_backend_select:n #1
  \__color_backend_select_gray:n 560 { \__kernel_backend_literal:n { pdf : bc ~ [ #1 ] } }
  \__color_backend_select_rgb:n 561 \cs_new_eq:NN \__color_backend_select_cmyk:n \__color_backend_select:n
\__color_backend_reset: 562 \cs_new_eq:NN \__color_backend_select_gray:n \__color_backend_select:n
563 \cs_new_eq:NN \__color_backend_select_rgb:n \__color_backend_select:n
564 \cs_new_protected:Npn \__color_backend_reset:
565 { \__kernel_backend_literal:n { pdf : ec } }

```

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

```
\__color_backend_select_named:n For classical named colors, the only value we should get is Black.
```

```

566 \cs_new_protected:Npn \__color_backend_select_named:n #1
567 {
568   \str_if_eq:nnTF {#1} { Black }
569     { \__color_backend_select_gray:n { 0 } }
570     { \msg_error:nnn { color } { unknown-named-color } {#1} }
571 }
572 \msg_new:nnn { color } { unknown-named-color }
573 { Named-color~'#1'~is~not~known. }

```

(End definition for `__color_backend_select_named:n`.)

```
574 </dvipdfmx | xetex>
```

3.4 Separations

Here, life gets interesting and we need essentially one approach per backend.

```
575 <*dvipdfmx | luatex | pdftex | xetex | dvips>
```

But we start with some functionality needed for both PostScript and PDF based backends.

`\g_color_backend_colorant_prop`

```
576 \prop_new:N \g_color_backend_colorant_prop
```

(End definition for `\g_color_backend_colorant_prop`.)

`_color_backend_devicen_colorants:n`

`_color_backend_devicen_colorants:w`

```
577 \cs_new:Npx \_color_backend_devicen_colorants:n #1
```

```
578 {
```

```
579   \exp_not:N \tl_if_blank:nF {#1}
```

```
580   {
```

```
581     \c_space_tl
```

```
582     << ~
```

```
583     /Colorants ~
```

```
584     << ~
```

```
585     \exp_not:N \_color_backend_devicen_colorants:w #1 ~
```

```
586     \exp_not:N \q_recursion_tail \c_space_tl
```

```
587     \exp_not:N \q_recursion_stop
```

```
588     >> ~
```

```
589     >>
```

```
590   }
```

```
591 }
```

```
592 \cs_new:Npn \_color_backend_devicen_colorants:w #1 ~
```

```
593 {
```

```
594   \quark_if_recursion_tail_stop:n {#1}
```

```
595   \prop_if_in:NnT \g_color_backend_colorant_prop {#1}
```

```
596   {
```

```
597     #1 ~
```

```
598     \prop_item:Nn \g_color_backend_colorant_prop {#1} ~
```

```
599   }
```

```
600   \_color_backend_devicen_colorants:w
```

```
601 }
```

(End definition for `_color_backend_devicen_colorants:n` and `_color_backend_devicen_colorants:w`.)

```
602 </dvipdfmx | luatex | pdftex | xetex | dvips>
```

```
603 <*dvips>
```

`_color_backend_select_separation:nn`

`_color_backend_select_devicen:nn`

```
604 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
```

```
605 { \_color_backend_select:n { separation ~ #1 ~ #2 } }
```

```
606 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
```

(End definition for `_color_backend_select_separation:nn` and `_color_backend_select_devicen:nn`.)

`_color_backend_select_iccbased:nn`

No support.

```
607 \cs_new_protected:Npn \_color_backend_select_iccbased:nn #1#2 { }
```

(End definition for `_color_backend_select_iccbased:nn`.)

`_color_backend_separation_init:nmnn`

`_color_backend_separation_init:nxxnn`

`_color_backend_separation_init_aux:nmnnnn`

`_color_backend_separation_init/DeviceCMYK:nnn`

`_color_backend_separation_init/DeviceGray:nnn`

`_color_backend_separation_init/DeviceRGB:nnn`

```
608 \cs_new_protected:Npx \_color_backend_separation_init:nmnnn #1#2#3#4#5
```

```
609 {
```

`_color_backend_separation_init_Device:Mn`

`_color_backend_separation_init:nnm`

`_color_backend_separation_init_count:n`

`_color_backend_separation_init_count:w`

`_color_backend_separation_init:nmnn`

`_color_backend_separation_init:w`

`_color_backend_separation_init:n`

`_color_backend_separation_init:nw`

`_color_backend_separation_init_CIELAB:nnm`

```

610 \bool_if:NT \g__kernel_backend_header_bool
611 {
612   \exp_args:Nx \__kernel_backend_first_shipout:n
613   {
614     \exp_not:N \__color_backend_separation_init_aux:nmnnnn
615     { \exp_not:N \int_use:N \g__color_model_int }
616     {#1} {#2} {#3} {#4} {#5}
617   }
618   \prop_gput:Nxx \exp_not:N \g__color_backend_colorant_prop
619   { / \exp_not:N \str_convert_pdfname:n {#1} }
620   {
621     << ~
622     /setcolorspace ~ {} ~
623     >> ~ begin ~
624     color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
625     end
626   }
627 }
628 }
629 \cs_generate_variant:Nn \__color_backend_separation_init:nmnnnn { nxx }
630 \cs_new_protected:Npn \__color_backend_separation_init_aux:nmnnnn #1#2#3#4#5#6
631 {
632   \__kernel_backend_literal:e
633   {
634     !
635     TeXDict ~ begin ~
636     /color #1
637     {
638       [ ~
639       /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
640       [ ~ #3 ~ ] ~
641       {
642         \cs_if_exist_use:cF { __color_backend_separation_init_ #3 :nmm }
643         { \__color_backend_separation_init:nmm }
644         {#4} {#5} {#6}
645       }
646       ] ~ setcolorspace
647     } ~ def ~
648     end
649   }
650 }
651 \cs_new:cpn { __color_backend_separation_init_ /DeviceCMYK :nmm } #1#2#3
652 { \__color_backend_separation_init_Device:Nn 4 {#3} }
653 \cs_new:cpn { __color_backend_separation_init_ /DeviceGray :nmm } #1#2#3
654 { \__color_backend_separation_init_Device:Nn 1 {#3} }
655 \cs_new:cpn { __color_backend_separation_init_ /DeviceRGB :nmm } #1#2#3
656 { \__color_backend_separation_init_Device:Nn 2 {#3} }
657 \cs_new:Npn \__color_backend_separation_init_Device:Nn #1#2
658 {
659   #2 ~
660   \prg_replicate:nn {#1}
661   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
662   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
663 }

```

For the generic case, we cannot use `/FunctionType 2` unfortunately, so we have to code that idea up in PostScript. Here, we will therefore assume that a range is *always* given. First, we count values in each argument: at the backend level, we can assume there are always well-behaved with spaces present.

```

664 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3
665 {
666   \exp_args:Ne \__color_backend_separation_init:nnnn
667   { \__color_backend_separation_init_count:n {#2} }
668   {#1} {#2} {#3}
669 }
670 \cs_new:Npn \__color_backend_separation_init_count:n #1
671 { \int_eval:n { 0 \__color_backend_separation_init_count:w #1 ~ \s__color_stop } }
672 \cs_new:Npn \__color_backend_separation_init_count:w #1 ~ #2 \s__color_stop
673 {
674   +1
675   \tl_if_blank:nF {#2}
676   { \__color_backend_separation_init_count:w #2 \s__color_stop }
677 }

```

Now we implement the algorithm. In the terms in the PostScript manual, we have $\mathbf{N} = 1$ and $\mathbf{Domain} = [0 \ 1]$, with \mathbf{Range} as #2, $\mathbf{C0}$ as #3 and $\mathbf{C1}$ as #4, with the number of output components in #1. So all we have to do is implement $y_i = \mathbf{C0}_i + x(\mathbf{C1}_i - \mathbf{C0}_i)$ with lots of stack manipulation, then check the ranges. That's done by adding everything to the stack first, then using the fact we know all of the offsets. As manipulating the stack is tricky, we start by re-formatting the $\mathbf{C0}$ and $\mathbf{C1}$ arrays to be interleaved, and add a 0 to each pair: this is used to keep the stack of constant length while we are doing the first pass of mathematics. We then working through that list, calculating from the last to the first value before tidying up by removing all of the input values. We do that by first copying all of the final y values to the end of the stack, then rolling everything so we can pop the now-unneeded material.

```

678 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
679 {
680   \__color_backend_separation_init:w #3 ~ \s__color_stop #4 ~ \s__color_stop
681   \prg_replicate:nn {#1}
682   {
683     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
684     \int_eval:n { 3 * #1 } ~ index ~ mul ~
685     2 ~ index ~ add ~
686     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
687   }
688   \int_step_function:nnnN {#1} { -1 } { 1 }
689   \__color_backend_separation_init:n
690   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
691   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
692   \tl_if_blank:nF {#2}
693   { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
694 }
695 \cs_new:Npn \__color_backend_separation_init:w
696 #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
697 {
698   #1 ~ #3 ~ 0 ~
699   \tl_if_blank:nF {#2}
700   { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }

```

```

701 }
702 \cs_new:Npn \__color_backend_separation_init:n #1
703 { \int_eval:n { #1 * 2 } ~ index ~ }

```

Finally, we deal with the range limit if required. This is handled by splitting the range into pairs. It's then just a question of doing the comparisons, this time dropping everything except the desired result.

```

704 \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s__color_stop
705 {
706   #2 ~ #3 ~
707   2 ~ index ~ 2 ~ index ~ lt ~
708   { ~ pop ~ exch ~ pop ~ } ~
709   { ~
710     2 ~ index ~ 1 ~ index ~ gt ~
711     { ~ exch ~ pop ~ exch ~ pop ~ } ~
712     { ~ pop ~ pop ~ } ~
713     ifelse ~
714   }
715   ifelse ~
716   #1 ~ 1 ~ roll ~
717   \tl_if_blank:nF {#4}
718   { \__color_backend_separation_init:nw {#1} #4 \s__color_stop }
719 }

```

CIELAB support uses the detail from the PostScript reference, page 227; other than that block of PostScript, this is the same as for PDF-based routes.

```

720 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nmn #1#2#3
721 {
722   \__color_backend_separation_init:nxxxn
723   {#2}
724   {
725     /CIEBasedABC ~
726     << ~
727     /RangeABC ~ [ ~ \c__color_model_range_CIELAB_tl \c_space_tl ] ~
728     /DecodeABC ~
729     [ ~
730     { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
731     { ~ 500 ~ div ~ } ~ bind ~
732     { ~ 200 ~ div ~ } ~ bind ~
733     ] ~
734     /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
735     /DecodeLMN ~
736     [ ~
737     { ~
738     dup ~ 6 ~ 29 ~ div ~ ge ~
739     { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
740     { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
741     ifelse ~
742     0.9505 ~ mul ~
743     } ~ bind ~
744     { ~
745     dup ~ 6 ~ 29 ~ div ~ ge ~
746     { ~ dup ~ dup ~ mul ~ mul ~ } ~
747     { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
748     ifelse ~

```

```

749         } ~ bind ~
750         { ~
751           dup ~ 6 ~ 29 ~ div ~ ge ~
752           { ~ dup ~ dup ~ mul ~ mul ~ } ~
753           { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
754           ifelse ~
755           1.0890 ~ mul ~
756         } ~ bind
757       ] ~
758       /WhitePoint ~
759       [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
760       >>
761     }
762     { \c__color_model_range_CIELAB_tl }
763     { 100 ~ 0 ~ 0 }
764     {#3}
765   }

```

(End definition for `__color_backend_separation_init:nnnnn` and others.)

`__color_backend_devicen_init:nmn` Trivial as almost all of the work occurs in the shared code.

```

766 \cs_new_protected:Npn \__color_backend_devicen_init:nmn #1#2#3
767 {
768   \__kernel_backend_literal:e
769   {
770     !
771     TeXDict ~ begin ~
772     /color \int_use:N \g__color_model_int
773     {
774       [ ~
775         /DeviceN ~
776         [ ~ #1 ~ ] ~
777         #2 ~
778         { ~ #3 ~ } ~
779         \__color_backend_devicen_colorants:n {#1}
780       ] ~ setcolorspace
781     } ~ def ~
782   end
783 }
784 }

```

(End definition for `__color_backend_devicen_init:nmn`.)

`__color_backend_iccbased_init:nmn` No support at present.

```

785 \cs_new_protected:Npn \__color_backend_iccbased_init:nmn #1#2#3 { }

```

(End definition for `__color_backend_iccbased_init:nmn`.)

```

786 </dvips>
787 <*dvisvgm>

```

`__color_backend_select_separation:nn` No support at present.

```

\__color_backend_select_devicen:nn
788 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2 { }
789 \cs_new_eq:NN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn

```

(End definition for `_color_backend_select_separation:nn` and `_color_backend_select_devicen:nn`.)

No support at present.

`_color_backend_separation_init:nmmn`
`_color_backend_separation_init_CIELAB:nnn`

```
790 \cs_new_protected:Npn \_color_backend_separation_init:nmmn #1#2#3#4#5 { }
791 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3 { }
```

(End definition for `_color_backend_separation_init:nmmn` and `_color_backend_separation_init_CIELAB:nnn`.)

`_color_backend_select_iccbased:nn`

As detailed in <https://www.w3.org/TR/css-color-4/#at-profile>, we can apply a color profile using CSS. As we have a local file, we use a relative URL.

```
792 \cs_new_protected:Npn \_color_backend_select_iccbased:nn #1#2
793 {
794   \_kernel_backend_literal_svg:x
795   {
796     <style>
797       @color-profile ~
798       \str_if_eq:nnTF {#2} { cmyk }
799       { device-cmyk }
800       { --color \int_use:N \g_color_model_int }
801       \c_space_tl
802       {
803         src:("#1")
804       }
805     </style>
806   }
807 }
```

(End definition for `_color_backend_select_iccbased:nn`.)

```
808 </dvisvgm>
809 <*dviPDFmx | luatex | pdftex | xetex>
```

`_color_backend_select_separation:nn`
`_color_backend_select_devicen:nn`
`_color_backend_select_iccbased:nn`

```
810 <*dviPDFmx | xetex>
811 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
812 { \_kernel_backend_literal:x { pdf : bc ~ \pdf_object_ref:n {#1} ~ [ #2 ] } }
813 </dviPDFmx | xetex>
814 <*luatex | pdftex>
815 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
816 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
817 </luatex | pdftex>
818 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
819 \cs_new_eq:NN \_color_backend_select_iccbased:nn \_color_backend_select_separation:nn
```

(End definition for `_color_backend_select_separation:nn`, `_color_backend_select_devicen:nn`, and `_color_backend_select_iccbased:nn`.)

`_color_backend_init_resource:n`

Resource initiation comes up a few times. For `dviPDFmx/XqTeX`, we skip this as at present it's handled by the backend.

```
820 \cs_new_protected:Npn \_color_backend_init_resource:n #1
821 {
822 <*luatex | pdftex>
823   \bool_lazy_and:nnT
824   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }

```

```

825     { \pdfmanagement_if_active_p: }
826     {
827         \use:x
828         {
829             \pdfmanagement_add:nnn
830             { Page / Resources / ColorSpace }
831             { #1 }
832             { \pdf_object_ref_last: }
833         }
834     }
835 </luatex | pdftex>
836 }

```

(End definition for `_color_backend_init_resource:n`.)

```

\_color_backend_separation_init:nmnm
\_color_backend_separation_init:nn
\_color_backend_separation_init_CIELAB:nnm

```

Initialising the PDF structures needs two parts: creating an object containing the “real” name of the Separation, then adding a reference to that to each page. We use a separate object for the tint transformation following the model in the PDF reference. The object here for the color needs to be named as that way it’s accessible to `dvipdfmx/X4TEX`.

```

837 \cs_new_protected:Npn \_color_backend_separation_init:nmnm #1#2#3#4#5
838 {
839     \pdf_object_unnamed_write:nx { dict }
840     {
841         /FunctionType ~ 2
842         /Domain ~ [0 ~ 1]
843         \t1_if_blank:nF {#3} { /Range ~ [#3] }
844         /CO ~ [#4] ~
845         /C1 ~ [#5] /N ~ 1
846     }
847     \exp_args:Nx \_color_backend_separation_init:nn
848     { \str_convert_pdfname:n {#1} } {#2}
849     \_color_backend_init_resource:n { color \int_use:N \g_color_model_int }
850 }
851 \cs_new_protected:Npn \_color_backend_separation_init:nn #1#2
852 {
853     \use:x
854     {
855         \pdf_object_new:n { color \int_use:N \g_color_model_int }
856         \pdf_object_write:nnm { color \int_use:N \g_color_model_int } { array }
857         { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
858     }
859     \prop_gput:Nnx \g_color_backend_colorant_prop { /#1 }
860     { \pdf_object_ref_last: }
861 }

```

For CIELAB colors, we need one object per document for the illuminant, plus initialisation of the color space referencing that object.

```

862 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnm #1#2#3
863 {
864     \pdf_object_if_exist:nF { \_color_illuminant_CIELAB_ #1 }
865     {
866         \pdf_object_new:n { \_color_illuminant_CIELAB_ #1 }
867         \pdf_object_write:nmx { \_color_illuminant_CIELAB_ #1 } { array }
868         {

```

```

869         /Lab ~
870         <<
871         /WhitePoint ~
872         [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _t1 } ]
873         /Range ~ [ \c__color_model_range_CIELAB_t1 ]
874         >>
875     }
876 }
877 \__color_backend_separation_init:nnnnn
878 {#2}
879 { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
880 { \c__color_model_range_CIELAB_t1 }
881 { 100 ~ 0 ~ 0 }
882 {#3}
883 }

```

(End definition for __color_backend_separation_init:nnnnn, __color_backend_separation_init:nn, and __color_backend_separation_init_CIELAB:nnn.)

__color_backend_devicen_init:nnn Similar to the Separations case, but with an arbitrary function for the alternative space
__color_backend_devicen_init:w work.

```

884 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
885 {
886   \pdf_object_unnamed_write:nx { stream }
887   {
888     {
889       /FunctionType ~ 4 ~
890       /Domain ~
891       [ ~
892         \prg_replicate:nn
893         { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
894         { 0 ~ 1 ~ }
895       ] ~
896       /Range ~
897       [ ~
898         \str_case:nn {#2}
899         {
900           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
901           { /DeviceGray } { 0 ~ 1 }
902           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
903         } ~
904       ]
905     }
906     { {#3} }
907   }
908   \use:x
909   {
910     \pdf_object_new:n { color \int_use:N \g__color_model_int }
911     \pdf_object_write:nnn { color \int_use:N \g__color_model_int } { array }
912     {
913       /DeviceN ~
914       [ ~ #1 ~ ] ~
915       #2 ~
916       \pdf_object_ref_last:

```

```

917         \_color_backend_devicen_colorants:n {#1}
918     }
919 }
920 \_color_backend_init_resource:n { color \int_use:N \g_color_model_int }
921 }
922 \cs_new:Npn \_color_backend_devicen_init:w #1 ~ #2 \s_color_stop
923 {
924     + 1
925     \tl_if_blank:nF {#2}
926     { \_color_backend_devicen_init:w #2 \s_color_stop }
927 }

```

(End definition for _color_backend_devicen_init:nnn and _color_backend_devicen_init:w.)

_color_backend_iccbased_init:nnn Lots of data to save here: we only want to do that once per file, so track it by name.

```

928 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3
929 {
930     \pdf_object_if_exist:nF { __color_icc_ #1 }
931     {
932         \pdf_object_new:n { __color_icc_ #1 }
933         \pdf_object_write:nxx { __color_icc_ #1 } { fstream }
934         {
935             {
936                 /N ~ \exp_not:n { #2 } ~
937                 \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
938             }
939             {#1}
940         }
941     }
942     \pdf_object_unnamed_write:nx { array }
943     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
944     \_color_backend_init_resource:n { color \int_use:N \g_color_model_int }
945 }

```

(End definition for _color_backend_iccbased_init:nnn.)

_color_backend_iccbased_device:nnn This is very similar to setting up a color space: the only part we add to the page resources differently.

```

946 \cs_new_protected:Npn \_color_backend_iccbased_device:nnn #1#2#3
947 {
948     \pdf_object_if_exist:nF { __color_icc_ #1 }
949     {
950         \pdf_object_new:n { __color_icc_ #1 }
951         \pdf_object_write:nnn { __color_icc_ #1 } { fstream }
952         {
953             { /N ~ #3 }
954             {#1}
955         }
956     }
957     \pdf_object_unnamed_write:nx { array }
958     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
959     \_color_backend_init_resource:n { Default #2 }
960 }

```

(End definition for _color_backend_iccbased_device:nnn.)

```

961 </dviPDFmx | luatex | pdftex | xetex>

```

3.5 Fill and stroke color

Here, dvipdfmx/X_YTeX we write direct PDF specials for the fill, and only use the stack for the stroke color (see above for comments on why we cannot use multiple stacks with these backends). LuaTeX and pdfTeX have mutiple stacks that can deal with fill and stroke. For dvips we have to manage fill and stroke color ourselves. We also handle dvisvgm independently, as there we can create SVG directly.

962 <*dvipdfmx | xetex>

```

    \_color_backend_fill:n
\_color_backend_fill_cmyk:n 963 \cs_new_protected:Npn \_color_backend_fill:n #1
\_color_backend_fill_gray:n 964 { \_kernel_backend_literal:n { pdf : bc ~ fill ~ [ #1 ] } }
\_color_backend_fill_rgb:n 965 \cs_new_eq:NN \_color_backend_fill_cmyk:n \_color_backend_fill:n
\_color_backend_stroke:n 966 \cs_new_eq:NN \_color_backend_fill_gray:n \_color_backend_fill:n
    \_color_backend_stroke_cmyk:n 967 \cs_new_eq:NN \_color_backend_fill_rgb:n \_color_backend_fill:n
    \_color_backend_stroke_gray:n 968 \cs_new_protected:Npn \_color_backend_stroke:n #1
    \_color_backend_stroke_rgb:n 969 { \_kernel_backend_literal:n { pdf : bc ~ stroke ~ [ #1 ] } }
970 \cs_new_eq:NN \_color_backend_stroke_cmyk:n \_color_backend_stroke:n
971 \cs_new_eq:NN \_color_backend_stroke_gray:n \_color_backend_stroke:n
972 \cs_new_eq:NN \_color_backend_stroke_rgb:n \_color_backend_stroke:n

```

(End definition for _color_backend_fill:n and others.)

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn 973 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
    \_color_backend_fill_devicen:nn 974 {
    \_color_backend_stroke_devicen:nn 975 \_kernel_backend_literal:x
976 { pdf : bc ~ fill ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
977 }
978 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
979 {
980 \_kernel_backend_literal:x
981 { pdf : bc ~ stroke ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
982 }
983 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
984 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

(End definition for _color_backend_fill_separation:nn and others.)

```

\_color_backend_fill_reset:
    \_color_backend_stroke_reset: 985 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
986 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:

```

(End definition for _color_backend_fill_reset: and _color_backend_stroke_reset:.)

987 </dvipdfmx | xetex>

988 <*luatex | pdftex>

```

\_color_backend_fill_cmyk:n
\_color_backend_fill_gray:n
\_color_backend_fill_rgb:n
    \_color_backend_fill:n
    \_color_backend_stroke_cmyk:n
    \_color_backend_stroke_gray:n
    \_color_backend_stroke_rgb:n
\_color_backend_stroke:n

```

Drawing (fill/stroke) color is handled in dvipdfmx/X_YTeX in the same way as LuaTeX/pdfTeX. We use the same approach as earlier, except the color stack is not involved so the generic direct PDF operation is used. There is no worry about the nature of strokes: everything is handled automatically.

```

989 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
990 { \_color_backend_fill:n { #1 ~ k } }

```

```

991 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
992   { \__color_backend_fill:n { #1 ~ g } }
993 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
994   { \__color_backend_fill:n { #1 ~ rg } }
995 \cs_new_protected:Npn \__color_backend_fill:n #1
996   {
997     \tl_set:Nn \l__color_backend_fill_tl {#1}
998     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
999     { #1 ~ \l__color_backend_stroke_tl }
1000  }
1001 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1002   { \__color_backend_stroke:n { #1 ~ K } }
1003 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1004   { \__color_backend_stroke:n { #1 ~ G } }
1005 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1006   { \__color_backend_stroke:n { #1 ~ RG } }
1007 \cs_new_protected:Npn \__color_backend_stroke:n #1
1008   {
1009     \tl_set:Nn \l__color_backend_stroke_tl {#1}
1010     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1011     { \l__color_backend_fill_tl \c_space_tl #1 }
1012  }

```

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

```

\__color_backend_fill_separation:nn
\__color_backend_stroke_separation:nn
\__color_backend_fill_devicen:nn
\__color_backend_stroke_devicen:nn
1013 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
1014   { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
1015 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1016   { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
1017 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1018 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

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

```

\__color_backend_fill_reset:
\__color_backend_stroke_reset:
1019 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1020 \cs_new_eq:NN \__color_backend_stroke_reset: \__color_backend_reset:

```

(End definition for `__color_backend_fill_reset:` and `__color_backend_stroke_reset:.`)

```
1021 </luatex | pdftex>
```

```
1022 <*dvips>
```

`__color_backend_fill_cmyk:n` Fill color here is the same as general color *except* we skip the stroke part.

```

\__color_backend_fill_gray:n
\__color_backend_fill_rgb:n
\__color_backend_fill:n
\__color_backend_stroke_cmyk:n
\__color_backend_stroke_gray:n
\__color_backend_stroke_rgb:n
1023 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1024   { \__color_backend_fill:n { cmyk ~ #1 } }
1025 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1026   { \__color_backend_fill:n { gray ~ #1 } }
1027 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1028   { \__color_backend_fill:n { rgb ~ #1 } }
1029 \cs_new_protected:Npn \__color_backend_fill:n #1
1030   {
1031     \__kernel_backend_literal:n { color~push~ #1 }
1032  }

```

```

1033 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1034 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1035 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1036 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1037 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1038 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }

```

(End definition for __color_backend_fill_cmyk:n and others.)

```

\__color_backend_fill_separation:nn
\__color_backend_stroke_separation:nn
\__color_backend_fill_devicen:nn
\__color_backend_stroke_devicen:nn
1039 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
1040 { \__color_backend_fill:n { separation ~ #1 ~ #2 } }
1041 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1042 { \__kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1043 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1044 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

\__color_backend_fill_reset:
\__color_backend_stroke_reset:
1045 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1046 \cs_new_protected:Npn \__color_backend_stroke_reset: { }

```

(End definition for __color_backend_fill_reset: and __color_backend_stroke_reset:.)

```

1047 </dvips>
1048 <*dvisvgm>

```

__color_backend_fill_cmyk:n Fill color here is the same as general color *except* we skip the stroke part.

```

\__color_backend_fill_gray:n
\__color_backend_fill_rgb:n
\__color_backend_fill:n
1049 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1050 { \__color_backend_fill:n { cmyk ~ #1 } }
1051 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1052 { \__color_backend_fill:n { gray ~ #1 } }
1053 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1054 { \__color_backend_fill:n { rgb ~ #1 } }
1055 \cs_new_protected:Npn \__color_backend_fill:n #1
1056 {
1057   \__kernel_backend_literal:n { color~push~ #1 }
1058 }

```

(End definition for __color_backend_fill_cmyk:n and others.)

__color_backend_stroke_cmyk:n For drawings in SVG, we use scopes for all stroke colors. That requires using RGB values, which luckily are easy to convert here (cmyk to RGB is a fixed function).

```

\__color_backend_stroke_cmyk:w
\__color_backend_stroke_gray:n
\__color_backend_stroke_gray_aux:n
\__color_backend_stroke_rgb:n
\__color_backend_stroke_rgb:w
\__color_backend:nnn
1059 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1060 { \__color_backend_cmyk:w #1 \s__color_stop }
1061 \cs_new_protected:Npn \__color_backend_stroke_cmyk:w
1062 #1 ~ #2 ~ #3 ~ #4 \s__color_stop
1063 {
1064   \use:x
1065   {
1066     \__color_backend:nnn
1067     { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1068     { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1069     { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }

```

```

1070     }
1071   }
1072   \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1073   {
1074     \use:x
1075     {
1076       \__color_backend_stroke_gray_aux:n
1077       { \fp_eval:n { 100 * (#1) } }
1078     }
1079   }
1080   \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1081   { \__color_backend:nnn {#1} {#1} {#1} }
1082   \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1083   { \__color_backend_rgb:w #1 \s__color_stop }
1084   \cs_new_protected:Npn \__color_backend_stroke_rgb:w
1085   #1 ~ #2 ~ #3 \s__color_stop
1086   {
1087     \use:x
1088     {
1089       \__color_backend:nnn
1090       { \fp_eval:n { 100 * (#1) } }
1091       { \fp_eval:n { 100 * (#2) } }
1092       { \fp_eval:n { 100 * (#3) } }
1093     }
1094   }
1095   \cs_new_protected:Npx \__color_backend:nnn #1#2#3
1096   {
1097     \__kernel_backend_scope:n
1098     {
1099       stroke =
1100       "
1101         rgb
1102         (
1103           #1 \c_percent_str ,
1104           #2 \c_percent_str ,
1105           #3 \c_percent_str
1106         )
1107       "
1108     }
1109   }

```

(End definition for __color_backend_stroke_cmyk:n and others.)

At present, these are no-ops.

```

\__color_backend_fill_separation:nn
\__color_backend_stroke_separation:nn
\__color_backend_fill_devicen:nn
\__color_backend_stroke_devicen:nn
1110 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1111 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2 { }
1112 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1113 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

\__color_backend_fill_reset:
\__color_backend_stroke_reset:
1114 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1115 \cs_new_protected:Npn \__color_backend_stroke_reset: { }

```

(End definition for `_color_backend_fill_reset:` and `_color_backend_stroke_reset:`.)

`_color_backend_devicen_init:nnn` No support at present.

```
\_color_backend_iccbased_init:nnn 1116 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3 { }  
1117 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3 { }
```

(End definition for `_color_backend_devicen_init:nnn` and `_color_backend_iccbased_init:nnn`.)

```
1118 \end{dvipsvm}
```

```
1119 \end{package}
```

4 I3backend-draw Implementation

```
1120 \*package
```

```
1121 \@@=draw
```

4.1 dvips backend

```
1122 \*dvips
```

`_draw_backend_literal:n` The same as literal PostScript: same arguments about positioning apply her.

```
\_draw_backend_literal:x 1123 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n  
1124 \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 matching `ps::[end]`: contrast with `ps:`, which positions but where we can't split material between separate calls. The `@beginspecial/@endspecial` pair are from `special.pro` and correct the scale and y -axis direction. 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.

```
1125 \cs_new_protected:Npn \_draw_backend_begin:  
1126 {  
1127   \_kernel_backend_literal:n { ps::[begin] }  
1128   \_draw_backend_literal:n { @beginspecial }  
1129 }  
1130 \cs_new_protected:Npn \_draw_backend_end:  
1131 {  
1132   \_draw_backend_literal:n { @endspecial }  
1133   \_kernel_backend_literal:n { ps::[end] }  
1134 }
```

(End definition for `_draw_backend_begin:` and `_draw_backend_end:`.)

`_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.

```
\_draw_backend_scope_end: 1135 \cs_new_protected:Npn \_draw_backend_scope_begin:  
1136 { \_draw_backend_literal:n { save } }  
1137 \cs_new_protected:Npn \_draw_backend_scope_end:  
1138 { \_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
`_draw_backend_lineto:nn` the need to convert to bp. Notice that x-type expansion is included here to ensure that
`_draw_backend_rectangle:nmmn` any variable values are forced to literals before any possible caching. There is no native
`_draw_backend_curveto:nmmmmn` rectangular path command (without also clipping, filling or stroking), so that task is
done using a small amount of PostScript.

```

1139 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1140 {
1141   \_draw_backend_literal:x
1142   {
1143     \dim_to_decimal_in_bp:n {#1} ~
1144     \dim_to_decimal_in_bp:n {#2} ~ moveto
1145   }
1146 }
1147 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1148 {
1149   \_draw_backend_literal:x
1150   {
1151     \dim_to_decimal_in_bp:n {#1} ~
1152     \dim_to_decimal_in_bp:n {#2} ~ lineto
1153   }
1154 }
1155 \cs_new_protected:Npn \_draw_backend_rectangle:nmmn #1#2#3#4
1156 {
1157   \_draw_backend_literal:x
1158   {
1159     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1160     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1161     moveto-dup-0~rlineto-exch-0~exch~rlineto-neg-0~rlineto~closepath
1162   }
1163 }
1164 \cs_new_protected:Npn \_draw_backend_curveto:nmmmmn #1#2#3#4#5#6
1165 {
1166   \_draw_backend_literal:x
1167   {
1168     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1169     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1170     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1171     curveto
1172   }
1173 }

```

(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:` 1174 `\cs_new_protected:Npn _draw_backend_evenodd_rule:`
`\g__draw_draw_eor_bool` 1175 `{ \bool_gset_true:N \g__draw_draw_eor_bool }`
1176 `\cs_new_protected:Npn _draw_backend_nonzero_rule:`
1177 `{ \bool_gset_false:N \g__draw_draw_eor_bool }`
1178 `\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
`__draw_backend_stroke:` also desirable to have the `clip` keyword after a stroke or fill. To achieve those outcomes,
`__draw_backend_closestroke:` there is some work to do. For color, the stroke color is simple but the fill one has to be
`__draw_backend_fill:` inserted by hand. For clipping, the required ordering is achieved using a `TEX` switch.
`__draw_backend_fillstroke:` All of the operations end with a new path instruction as they do not terminate (again in
`__draw_backend_clip:` contrast to PDF).
`__draw_backend_discardpath:`

```

1179 \cs_new_protected:Npn \__draw_backend_closepath:
1180 { \__draw_backend_literal:n { closepath } }
1181 \cs_new_protected:Npn \__draw_backend_stroke:
1182 {
1183   \__draw_backend_literal:n { gsave }
1184   \__draw_backend_literal:n { color.sc }
1185   \__draw_backend_literal:n { stroke }
1186   \__draw_backend_literal:n { grestore }
1187   \bool_if:NT \g__draw_draw_clip_bool
1188     {
1189       \__draw_backend_literal:x
1190       {
1191         \bool_if:NT \g__draw_draw_eor_bool { eo }
1192         clip
1193       }
1194     }
1195   \__draw_backend_literal:n { newpath }
1196   \bool_gset_false:N \g__draw_draw_clip_bool
1197 }
1198 \cs_new_protected:Npn \__draw_backend_closestroke:
1199 {
1200   \__draw_backend_closepath:
1201   \__draw_backend_stroke:
1202 }
1203 \cs_new_protected:Npn \__draw_backend_fill:
1204 {
1205   \__draw_backend_literal:x
1206   {
1207     \bool_if:NT \g__draw_draw_eor_bool { eo }
1208     fill
1209   }
1210   \bool_if:NT \g__draw_draw_clip_bool
1211   {
1212     \__draw_backend_literal:x
1213     {
1214       \bool_if:NT \g__draw_draw_eor_bool { eo }
1215       clip
1216     }
1217   }
1218   \__draw_backend_literal:n { newpath }
1219   \bool_gset_false:N \g__draw_draw_clip_bool
1220 }
1221 \cs_new_protected:Npn \__draw_backend_fillstroke:
1222 {
1223   \__draw_backend_literal:x
1224   {
1225     \bool_if:NT \g__draw_draw_eor_bool { eo }

```

```

1226         fill
1227     }
1228     \_draw_backend_literal:n { gsave }
1229     \_draw_backend_literal:n { color.sc }
1230     \_draw_backend_literal:n { stroke }
1231     \_draw_backend_literal:n { grestore }
1232     \bool_if:NT \g__draw_draw_clip_bool
1233     {
1234         \_draw_backend_literal:x
1235         {
1236             \bool_if:NT \g__draw_draw_eor_bool { eo }
1237             clip
1238         }
1239     }
1240     \_draw_backend_literal:n { newpath }
1241     \bool_gset_false:N \g__draw_draw_clip_bool
1242 }
1243 \cs_new_protected:Npn \_draw_backend_clip:
1244 { \bool_gset_true:N \g__draw_draw_clip_bool }
1245 \bool_new:N \g__draw_draw_clip_bool
1246 \cs_new_protected:Npn \_draw_backend_discardpath:
1247 {
1248     \bool_if:NT \g__draw_draw_clip_bool
1249     {
1250         \_draw_backend_literal:x
1251         {
1252             \bool_if:NT \g__draw_draw_eor_bool { eo }
1253             clip
1254         }
1255     }
1256     \_draw_backend_literal:n { newpath }
1257     \bool_gset_false:N \g__draw_draw_clip_bool
1258 }

```

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

`_draw_backend_dash_pattern:nn` Converting paths to output is again a case of mapping directly to PostScript operations.

```

1259 \cs_new_protected:Npn \_draw_backend_dash_pattern:nn #1#2
1260 {
1261     \_draw_backend_literal:x
1262     {
1263         [
1264             \exp_args:Nf \use:n
1265             { \clist_map_function:nN {#1} \_draw_backend_dash:n }
1266         ] ~
1267         \dim_to_decimal_in_bp:n {#2} ~ setdash
1268     }
1269 }
1270 \cs_new:Npn \_draw_backend_dash:n #1
1271 { ~ \dim_to_decimal_in_bp:n {#1} }
1272 \cs_new_protected:Npn \_draw_backend_linewidth:n #1
1273 {
1274     \_draw_backend_literal:x
1275     { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }

```

```

1276 }
1277 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1278 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1279 \cs_new_protected:Npn \__draw_backend_cap_butt:
1280 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1281 \cs_new_protected:Npn \__draw_backend_cap_round:
1282 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1283 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1284 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1285 \cs_new_protected:Npn \__draw_backend_join_miter:
1286 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1287 \cs_new_protected:Npn \__draw_backend_join_round:
1288 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1289 \cs_new_protected:Npn \__draw_backend_join_bevel:
1290 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

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

`__draw_backend_cm:nmmm` 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.* `dvipdfmx/XYTeX`). Thus we take the shortest path available and simply dump the matrix as given.

```

1291 \cs_new_protected:Npn \__draw_backend_cm:nmmm #1#2#3#4
1292 {
1293   \__draw_backend_literal:n
1294   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1295 }

```

(End definition for `__draw_backend_cm:nmmm`.)

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

```

1296 \cs_new_protected:Npn \__draw_backend_box_use:Nmmmm #1#2#3#4#5
1297 {
1298   \__draw_backend_literal:n { @endspecial }
1299   \__draw_backend_literal:n { [end] }
1300   \__draw_backend_literal:n { [begin] }
1301   \__draw_backend_literal:n { save }
1302   \__draw_backend_literal:n { currentpoint }
1303   \__draw_backend_literal:n { currentpoint~translate }
1304   \__draw_backend_cm:nmmm { 1 } { 0 } { 0 } { -1 }
1305   \__draw_backend_cm:nmmm {#2} {#3} {#4} {#5}
1306   \__draw_backend_cm:nmmm { 1 } { 0 } { 0 } { -1 }
1307   \__draw_backend_literal:n { neg~exch~neg~exch~translate }

```

```

1308     \_draw_backend_literal:n { [end] }
1309     \hbox_overlap_right:n { \box_use:N #1 }
1310     \_draw_backend_literal:n { [begin] }
1311     \_draw_backend_literal:n { restore }
1312     \_draw_backend_literal:n { [end] }
1313     \_draw_backend_literal:n { [begin] }
1314     \_draw_backend_literal:n { @beginspecial }
1315 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```

1316 </dvips>

```

4.2 LuaTeX, pdfTeX, dvipdfmx and XeTeX

LuaTeX, pdfTeX, dvipdfmx and XeTeX directly produce PDF output and understand a shared set of specials for drawing commands.

```

1317 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

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

```

\_draw_backend_literal:x
1318 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_pdf:n
1319 \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:
1320 \cs_new_protected:Npn \_draw_backend_begin:
1321 { \_draw_backend_scope_begin: }
1322 \cs_new_protected:Npn \_draw_backend_end:
1323 { \_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:
1324 \cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:
1325 \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:nnnnn
  \_draw_backend_rectangle:nnnn
1326 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1327 {
1328   \_draw_backend_literal:x
1329   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1330 }
1331 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1332 {
1333   \_draw_backend_literal:x
1334   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1335 }
1336 \cs_new_protected:Npn \_draw_backend_curveto:nnnnn #1#2#3#4#5#6
1337 {

```

```

1338   \__draw_backend_literal:x
1339   {
1340     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1341     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1342     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1343     c
1344   }
1345 }
1346 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1347 {
1348   \__draw_backend_literal:x
1349   {
1350     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1351     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1352     re
1353   }
1354 }

```

(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`

```

1355 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1356 { \bool_gset_true:N \g__draw_draw_eor_bool }
1357 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1358 { \bool_gset_false:N \g__draw_draw_eor_bool }
1359 \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:` Converting paths to output is again a case of mapping directly to PDF operations.

```

\__draw_backend_stroke: 1360 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_closestroke: 1361 { \__draw_backend_literal:n { h } }
\__draw_backend_fill: 1362 \cs_new_protected:Npn \__draw_backend_stroke:
\__draw_backend_fillstroke: 1363 { \__draw_backend_literal:n { S } }
\__draw_backend_clip: 1364 \cs_new_protected:Npn \__draw_backend_closestroke:
\__draw_backend_discardpath: 1365 { \__draw_backend_literal:n { s } }
1366 \cs_new_protected:Npn \__draw_backend_fill:
1367 {
1368   \__draw_backend_literal:x
1369   { f \bool_if:NT \g__draw_draw_eor_bool * }
1370 }
1371 \cs_new_protected:Npn \__draw_backend_fillstroke:
1372 {
1373   \__draw_backend_literal:x
1374   { B \bool_if:NT \g__draw_draw_eor_bool * }
1375 }
1376 \cs_new_protected:Npn \__draw_backend_clip:
1377 {
1378   \__draw_backend_literal:x
1379   { W \bool_if:NT \g__draw_draw_eor_bool * }
1380 }
1381 \cs_new_protected:Npn \__draw_backend_discardpath:
1382 { \__draw_backend_literal:n { n } }

```

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

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n      1383 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
\__draw_backend_linewidth:n 1384 {
\__draw_backend_miterlimit:n 1385   \__draw_backend_literal:x
\__draw_backend_cap_but:    1386   {
\__draw_backend_cap_round:  1387     [
\__draw_backend_cap_rectangle: 1388     \exp_args:Nf \use:n
\__draw_backend_join_miter:  1389     { \clist_map_function:nN {#1} \__draw_backend_dash:n }
\__draw_backend_join_round:  1390     ] ~
\__draw_backend_join_bevel:  1391     \dim_to_decimal_in_bp:n {#2} ~ d
1392   }
1393 }
1394 \cs_new:Npn \__draw_backend_dash:n #1
1395 { ~ \dim_to_decimal_in_bp:n {#1} }
1396 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1397 {
1398   \__draw_backend_literal:x
1399   { \dim_to_decimal_in_bp:n {#1} ~ w }
1400 }
1401 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1402 { \__draw_backend_literal:x { #1 ~ M } }
1403 \cs_new_protected:Npn \__draw_backend_cap_but:
1404 { \__draw_backend_literal:n { 0 ~ J } }
1405 \cs_new_protected:Npn \__draw_backend_cap_round:
1406 { \__draw_backend_literal:n { 1 ~ J } }
1407 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1408 { \__draw_backend_literal:n { 2 ~ J } }
1409 \cs_new_protected:Npn \__draw_backend_join_miter:
1410 { \__draw_backend_literal:n { 0 ~ j } }
1411 \cs_new_protected:Npn \__draw_backend_join_round:
1412 { \__draw_backend_literal:n { 1 ~ j } }
1413 \cs_new_protected:Npn \__draw_backend_join_bevel:
1414 { \__draw_backend_literal:n { 2 ~ j } }

```

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

`_draw_backend_cm:nnnn` Another split here between LuaTeX/pdfTeX and dvipdfmx/X_qTeX. In the former, we have a direct method to maintain alignment: the backend can use a matrix itself. For dvipdfmx/X_qTeX, 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 dvipdfmx/X_qTeX, but as a matched pair so not suitable for the “stand alone” transformation set up here.) The specials used here are from xdvipdfmx originally: they are well-tested, but probably equivalent to the pdf: versions!

```

1415 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1416 {
1417   <*luatex | pdftex>
1418   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1419   </luatex | pdftex>
1420   <*dvipdfmx | xetex>
1421   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1422   \__draw_backend_cm_aux:nnnn
1423   </dvipdfmx | xetex>

```

```

1424 }
1425 <*dvipdfmx|xetex>
1426 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1427 {
1428   \__kernel_backend_literal:x
1429   {
1430     x:rotate~
1431     \fp_compare:nNnTF {#1} = \c_zero_fp
1432     { 0 }
1433     { \fp_eval:n { round ( -#1 , 5 ) } }
1434   }
1435   \__kernel_backend_literal:x
1436   {
1437     x:scale~
1438     \fp_eval:n { round ( #2 , 5 ) } ~
1439     \fp_eval:n { round ( #3 , 5 ) }
1440   }
1441   \__kernel_backend_literal:x
1442   {
1443     x:rotate~
1444     \fp_compare:nNnTF {#4} = \c_zero_fp
1445     { 0 }
1446     { \fp_eval:n { round ( -#4 , 5 ) } }
1447   }
1448 }
1449 </dvipdfmx|xetex>

```

(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.

```

1450 <*dviptdpmx|xetex>
1451 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1452 {
1453   \use:x
1454   {
1455     \__draw_backend_cm_decompose_auxi:nnnnN
1456     { \fp_eval:n { (#1 + #4) / 2 } }
1457     { \fp_eval:n { (#1 - #4) / 2 } }
1458     { \fp_eval:n { (#3 + #2) / 2 } }
1459     { \fp_eval:n { (#3 - #2) / 2 } }
1460   }
1461   #5
1462 }
1463 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1464 {
1465   \use:x
1466   {
1467     \__draw_backend_cm_decompose_auxii:nnnnN
1468     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1469     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1470     { \fp_eval:n { atand ( #3 , #2 ) } }
1471     { \fp_eval:n { atand ( #4 , #1 ) } }
1472   }
1473   #5
1474 }
1475 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1476 {
1477   \use:x
1478   {
1479     \__draw_backend_cm_decompose_auxiii:nnnnN
1480     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1481     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1482     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1483     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1484   }
1485   #5
1486 }
1487 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1488 {
1489   \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1490   { #5 {#1} {#2} {#3} {#4} }
1491   { #5 {#1} {#3} {#2} {#4} }
1492 }
1493 </dviptdpmx|xetex>

```

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

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

```

1494 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1495 {
1496   \__kernel_backend_scope_begin:
1497   <*luatex | pdftex>
1498   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1499   </luatex | pdftex>
1500   <*dviptfm | xetex>
1501   \__kernel_backend_literal:n
1502   { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1503   </dviptfm | xetex>
1504   \hbox_overlap_right:n { \box_use:N #1 }
1505   <*dviptfm | xetex>
1506   \__kernel_backend_literal:n { pdf:etrans }
1507   </dviptfm | xetex>
1508   \__kernel_backend_scope_end:
1509 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1510 </dviptfm | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1511 <*dvisvgm>

```

__draw_backend_literal:n The same as the more general literal call.

```

\__draw_backend_literal:x
1512 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1513 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for __draw_backend_literal:n.)

__draw_backend_scope_begin: Use the backend-level scope mechanisms.

```

\__draw_backend_scope_end:
1514 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1515 \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_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

```

\__draw_backend_end:
1516 \cs_new_protected:Npn \__draw_backend_begin:
1517 {
1518   \__kernel_backend_scope_begin:
1519   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1520 }
1521 \cs_new_eq:NN \__draw_backend_end: \__kernel_backend_scope_end:

```

(End definition for __draw_backend_begin: and __draw_backend_end:.)

__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. __draw_backend_lineto:nn Since paths should be fully expanded there is no need to worry about the internal x-type expansion. __draw_backend_rectangle:nnnnn __draw_backend_curveto:nnnnnn __draw_backend_add_to_path:n

```

\g__draw_backend_path_tl
1522 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1523 {

```

```

1524     \__draw_backend_add_to_path:n
1525     { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1526   }
1527 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1528   {
1529     \__draw_backend_add_to_path:n
1530     { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1531   }
1532 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1533   {
1534     \__draw_backend_add_to_path:n
1535     {
1536       M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1537       h ~ \dim_to_decimal:n {#3} ~
1538       v ~ \dim_to_decimal:n {#4} ~
1539       h ~ \dim_to_decimal:n { -#3 } ~
1540       Z
1541     }
1542   }
1543 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1544   {
1545     \__draw_backend_add_to_path:n
1546     {
1547       C ~
1548       \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1549       \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1550       \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1551     }
1552   }
1553 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1554   {
1555     \tl_gset:Nx \g__draw_backend_path_tl
1556     {
1557       \g__draw_backend_path_tl
1558       \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1559       #1
1560     }
1561   }
1562 \tl_new:N \g__draw_backend_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: 1563 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1564   { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1565 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1566   { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

(End definition for __draw_backend_evenodd_rule: and __draw_backend_nonzero_rule:.)

```

\__draw_backend_path:n
\__draw_backend_closepath:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int

```

Setting fill and stroke effects and doing clipping all has to be done using scopes. This means setting up the various requirements in a shared auxiliary which deals with the bits and pieces. Clipping paths are reused for path drawing: not essential but avoids constructing them twice. Discarding a path needs a separate function as it's not quite the same.

```

1567 \cs_new_protected:Npn \__draw_backend_closepath:
1568   { \__draw_backend_add_to_path:n { Z } }
1569 \cs_new_protected:Npn \__draw_backend_path:n #1
1570   {
1571     \bool_if:NTF \g__draw_draw_clip_bool
1572     {
1573       \int_gincr:N \g__kernel_clip_path_int
1574       \__draw_backend_literal:x
1575       {
1576         < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1577         { ?nl }
1578         <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1579         < /clipPath > { ? nl }
1580         <
1581           use~xlink:href =
1582             "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1583             #1
1584           />
1585         }
1586         \__kernel_backend_scope:x
1587         {
1588           clip-path =
1589             "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1590         }
1591       }
1592     {
1593       \__draw_backend_literal:x
1594       { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1595     }
1596     \tl_gclear:N \g__draw_backend_path_tl
1597     \bool_gset_false:N \g__draw_draw_clip_bool
1598   }
1599 \int_new:N \g__draw_backend_path_int
1600 \cs_new_protected:Npn \__draw_backend_stroke:
1601   { \__draw_backend_path:n { style="fill:none" } }
1602 \cs_new_protected:Npn \__draw_backend_closestroke:
1603   {
1604     \__draw_backend_closepath:
1605     \__draw_backend_stroke:
1606   }
1607 \cs_new_protected:Npn \__draw_backend_fill:
1608   { \__draw_backend_path:n { style="stroke:none" } }
1609 \cs_new_protected:Npn \__draw_backend_fillstroke:
1610   { \__draw_backend_path:n { } }
1611 \cs_new_protected:Npn \__draw_backend_clip:
1612   { \bool_gset_true:N \g__draw_draw_clip_bool }
1613 \bool_new:N \g__draw_draw_clip_bool
1614 \cs_new_protected:Npn \__draw_backend_discardpath:
1615   {
1616     \bool_if:NT \g__draw_draw_clip_bool
1617     {
1618       \int_gincr:N \g__kernel_clip_path_int
1619       \__draw_backend_literal:x
1620       {

```

```

1621         < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1622             { ?nl }
1623         <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1624         < /clipPath >
1625     }
1626     \__kernel_backend_scope:x
1627     {
1628         clip-path =
1629         "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1630     }
1631 }
1632 \tl_gclear:N \g__draw_path_tl
1633 \bool_gset_false:N \g__draw_draw_clip_bool
1634 }

```

(End definition for _draw_backend_path:n and others.)

_draw_backend_dash_pattern:nn 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:n
\_draw_backend_dash_aux:nn 1635 \cs_new_protected:Npn \_draw_backend_dash_pattern:nn #1#2
\_draw_backend_linewidth:n 1636 {
\_draw_backend_miterlimit:n 1637     \use:x
\_draw_backend_cap_but: 1638     {
\_draw_backend_cap_round: 1639         \_draw_backend_dash_aux:nn
    \_draw_backend_cap_rectangle: 1640             { \clist_map_function:nN {#1} \_draw_backend_dash:n }
    \_draw_backend_join_miter: 1641             { \dim_to_decimal:n {#2} }
    \_draw_backend_join_round: 1642     }
    \_draw_backend_join_bevel: 1643 }
1644 \cs_new:Npn \_draw_backend_dash:n #1
1645 { , \dim_to_decimal_in_bp:n {#1} }
1646 \cs_new_protected:Npn \_draw_backend_dash_aux:nn #1#2
1647 {
1648     \__kernel_backend_scope:x
1649     {
1650         stroke-dasharray =
1651         "
1652             \tl_if_empty:nTF {#1}
1653             { none }
1654             { \use_none:n #1 }
1655         " ~
1656         stroke-offset=" #2 "
1657     }
1658 }
1659 \cs_new_protected:Npn \_draw_backend_linewidth:n #1
1660 { \__kernel_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1661 \cs_new_protected:Npn \_draw_backend_miterlimit:n #1
1662 { \__kernel_backend_scope:x { stroke-miterlimit=" #1 " } }
1663 \cs_new_protected:Npn \_draw_backend_cap_but:
1664 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1665 \cs_new_protected:Npn \_draw_backend_cap_round:
1666 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1667 \cs_new_protected:Npn \_draw_backend_cap_rectangle:
1668 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1669 \cs_new_protected:Npn \_draw_backend_join_miter:

```

```

1670 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1671 \cs_new_protected:Npn \__draw_backend_join_round:
1672 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1673 \cs_new_protected:Npn \__draw_backend_join_bevel:
1674 { \__kernel_backend_scope:n { stroke-linejoin="bevel" } }

```

(End definition for __draw_backend_dash_pattern:nn and others.)

__draw_backend_cm:nnnn The four arguments here are floats (the affine matrix), the last two are a displacement vector.

```

1675 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1676 {
1677   \__kernel_backend_scope:n
1678   {
1679     transform =
1680     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1681   }
1682 }

```

(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.

```

1683 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1684 {
1685   \__kernel_backend_scope_begin:
1686   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1687   \__kernel_backend_literal_svg:n
1688   {
1689     < g~
1690     stroke="none"~
1691     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1692     >
1693   }
1694   \box_set_wd:Nn #1 { Opt }
1695   \box_set_ht:Nn #1 { Opt }
1696   \box_set_dp:Nn #1 { Opt }
1697   \box_use:N #1
1698   \__kernel_backend_literal_svg:n { </g> }
1699   \__kernel_backend_scope_end:
1700 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

1701 </divisvgm>

1702 </package>

5 l3backend-graphics Implementation

```

1703 <*package>
1704 <@=graphics>

```

__graphics_backend_loaded:n To deal with file load ordering. Plain users are on their own.

```

1705 \cs_new_protected:Npn \__graphics_backend_loaded:n #1

```

```

1706 {
1707   \cs_if_exist:NTF \hook_gput_code:nnn
1708   {
1709     \hook_gput_code:nnn
1710     { file / l3graphics.sty / after }
1711     { backend }
1712     {#1}
1713   }
1714   {#1}
1715 }

```

(End definition for `__graphics_backend_loaded:n`.)

5.1 dvips backend

```
1716 <dvips>
```

`\l_graphics_search_ext_seq`

```

1717 \__graphics_backend_loaded:n
1718 { \seq_set_from_clist:Nn \l_graphics_search_ext_seq { .eps , .ps } }

```

(End definition for `\l_graphics_search_ext_seq`. This variable is documented on page ??.)

`__graphics_backend_getbb_eps:n`
`__graphics_backend_getbb_ps:n`

Simply use the generic function.

```

1719 \__graphics_backend_loaded:n
1720 {
1721   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1722   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
1723 }

```

(End definition for `__graphics_backend_getbb_eps:n` and `__graphics_backend_getbb_ps:n`.)

`__graphics_backend_include_eps:n`
`__graphics_backend_include_ps:n`

The special syntax is relatively clear here: remember we need PostScript sizes here.

```

1724 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1725 {
1726   \__kernel_backend_literal:x
1727   {
1728     PSfile = #1 \c_space_tl
1729     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1730     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1731     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1732     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1733   }
1734 }
1735 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n

```

(End definition for `__graphics_backend_include_eps:n` and `__graphics_backend_include_ps:n`.)

`__graphics_backend_get_pagecount:n`

```

1736 \__graphics_backend_loaded:n
1737 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }

```

(End definition for `__graphics_backend_get_pagecount:n`.)

```
1738 </dvips>
```

5.2 LuaTeX and pdfTeX backends

1739 `*luatex | pdftex`

`\l_graphics_search_ext_seq`

```

1740 \__graphics_backend_loaded:n
1741 {
1742   \seq_set_from_clist:Nn
1743   \l_graphics_search_ext_seq
1744   { .pdf , .eps , .ps , .png , .jpg , .jpeg }
1745 }

```

(End definition for `\l_graphics_search_ext_seq`. This variable is documented on page ??.)

`\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.

```

1746 \tl_new:N \l__graphics_graphics_attr_tl

```

(End definition for `\l__graphics_graphics_attr_tl`.)

`_graphics_backend_getbb_jpg:n`

`_graphics_backend_getbb_jpeg:n`

`_graphics_backend_getbb_pdf:n`

`_graphics_backend_getbb_png:n`

`_graphics_backend_getbb_auxi:n`

`_graphics_backend_getbb_auxii:n`

`_graphics_backend_getbb_auxiii:n`

`_graphics_backend_dequote:w`

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.

```

1747 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1

```

```

1748 {
1749   \int_zero:N \l__graphics_page_int
1750   \tl_clear:N \l__graphics_pagebox_tl
1751   \tl_set:Nx \l__graphics_graphics_attr_tl
1752   {
1753     \tl_if_empty:NF \l__graphics_decodearray_str
1754     { :D \l__graphics_decodearray_str }
1755     \bool_if:NT \l__graphics_interpolate_bool
1756     { :I }
1757   }
1758   \tl_clear:N \l__graphics_graphics_attr_tl
1759   \__graphics_backend_getbb_auxi:n {#1}
1760 }

```

```

1761 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n

```

```

1762 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n

```

```

1763 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1

```

```

1764 {
1765   \tl_clear:N \l__graphics_decodearray_str
1766   \bool_set_false:N \l__graphics_interpolate_bool
1767   \tl_set:Nx \l__graphics_graphics_attr_tl
1768   {
1769     : \l__graphics_pagebox_tl
1770     \int_compare:nNnT \l__graphics_page_int > 1
1771     { :P \int_use:N \l__graphics_page_int }
1772   }

```

```

1773   \__graphics_backend_getbb_auxi:n {#1}
1774 }
1775 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1776 {
1777   \__graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1778   { \__graphics_backend_getbb_auxii:n {#1} }
1779 }

```

Measuring the graphic is done by boxing up: for PDF graphics we could use `\tex_pdfximagebbox:D`, but it 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. Quotes need to be *removed* as LuaTeX does not like them here.

```

1780 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1781 {
1782   \exp_args:Ne \__graphics_backend_getbb_auxiii:n
1783   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1784   \int_const:cn { c__graphics_ #1 \l__graphics_graphics_attr_tl _int }
1785   { \tex_the:D \tex_pdflastximage:D }
1786   \__graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1787 }
1788 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1789 {
1790   \tex_immediate:D \tex_pdfximage:D
1791   \bool_lazy_or:nnT
1792   { \l__graphics_interpolate_bool }
1793   { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1794   {
1795     attr ~
1796     {
1797       \tl_if_empty:NF \l__graphics_decodearray_str
1798       { /Decode~[ \l__graphics_decodearray_str ] }
1799       \bool_if:NT \l__graphics_interpolate_bool
1800       { /Interpolate~true }
1801     }
1802   }
1803   \int_compare:nNnT \l__graphics_page_int > 0
1804   { page ~ \int_use:N \l__graphics_page_int }
1805   \tl_if_empty:NF \l__graphics_pagebox_tl
1806   { \l__graphics_pagebox_tl }
1807   {#1}
1808   \hbox_set:Nn \l__graphics_internal_box
1809   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1810   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1811   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1812 }
1813 \cs_new:Npn \__graphics_backend_dequote:w #1 " #2 " #3 \s__graphics_stop {#2}

```

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

```

\__graphics_backend_include_jpg:n
\__graphics_backend_include_jpeg:n
\__graphics_backend_include_pdf:n
\__graphics_backend_include_png: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.

```

1814 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1815 {

```

```

1816 \tex_pdfrefximage:D
1817 \int_use:c { c__graphics_ #1 \l__graphics_graphics_attr_tl _int }
1818 }
1819 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1820 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1821 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

(End definition for __graphics_backend_include_jpg:n and others.)

EPS graphics may be included in LuaTeX/pdfTeX by conversion to PDF: this requires restricted shell escape. Modelled on the `epstopdf LATEX 2ε` package, but simplified, conversion takes place here if we have shell access.

```

\__graphics_backend_getbb_eps:n
\__graphics_backend_getbb_ps:n
\__graphics_backend_getbb_eps:nm
\__graphics_backend_include_eps:n
\__graphics_backend_include_ps:n
\l__graphics_backend_dir_str
\l__graphics_backend_name_str
\l__graphics_backend_ext_str
1822 \sys_if_shell:T
1823 {
1824 \str_new:N \l__graphics_backend_dir_str
1825 \str_new:N \l__graphics_backend_name_str
1826 \str_new:N \l__graphics_backend_ext_str
1827 \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1828 {
1829 \file_parse_full_name:nNNN {#1}
1830 \l__graphics_backend_dir_str
1831 \l__graphics_backend_name_str
1832 \l__graphics_backend_ext_str
1833 \exp_args:Nx \__graphics_backend_getbb_eps:nn
1834 {
1835 \exp_args:Ne \__kernel_file_name_quote:n
1836 {
1837 \l__graphics_backend_name_str
1838 - \str_tail:N \l__graphics_backend_ext_str
1839 -converted-to.pdf
1840 }
1841 }
1842 {#1}
1843 }
1844 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_backend_getbb_eps:n
1845 \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1846 {
1847 \file_compare_timestamp:nNnT {#2} > {#1}
1848 {
1849 \sys_shell_now:n
1850 { repstopdf ~ #2 ~ #1 }
1851 }
1852 \tl_set:Nn \l__graphics_final_name_str {#1}
1853 \__graphics_backend_getbb_pdf:n {#1}
1854 }
1855 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1856 {
1857 \file_parse_full_name:nNNN {#1}
1858 \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1859 \exp_args:Nx \__graphics_backend_include_pdf:n
1860 {
1861 \exp_args:Ne \__kernel_file_name_quote:n
1862 {
1863 \l__graphics_backend_name_str

```

```

1864         - \str_tail:N \l__graphics_backend_ext_str
1865         -converted-to.pdf
1866     }
1867 }
1868 }
1869 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1870 }

```

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

`__graphics_backend_get_pagecount:n` Simply load and store.

```

1871 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1872 {
1873     \tex_pdfximage:D {#1}
1874     \int_const:cn { c__graphics_ #1 _pages_int }
1875     { \int_use:N \tex_pdflastximagepages:D }
1876 }

```

(End definition for `__graphics_backend_get_pagecount:n`.)

```
1877 </luatex | pdftex>
```

5.3 dvipdfmx backend

```
1878 <*dvipdfmx | xetex>
```

`\l_graphics_search_ext_seq`

```

1879 \__graphics_backend_loaded:n
1880 {
1881     \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1882     { .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }
1883 }

```

(End definition for `\l_graphics_search_ext_seq`. This variable is documented on page ??.)

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

```

\__graphics_backend_getbb_ps:n 1884 \__graphics_backend_loaded:n
\__graphics_backend_getbb_jpg:n 1885 {
\__graphics_backend_getbb_jpeg:n 1886     \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
\__graphics_backend_getbb_pdf:n 1887     \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
\__graphics_backend_getbb_png:n 1888 }
\__graphics_backend_getbb_bmp:n 1889 <*dvipdfmx>
1890 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1891 {
1892     \int_zero:N \l__graphics_page_int
1893     \tl_clear:N \l__graphics_pagebox_tl
1894     \__graphics_extract_bb:n {#1}
1895 }
1896 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1897 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1898 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
1899 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1900 {
1901     \tl_clear:N \l__graphics_decodearray_str
1902     \bool_set_false:N \l__graphics_interpolate_bool

```

```

1903     \_graphics_extract_bb:n {#1}
1904   }
1905 </dviPDFfmX>

```

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

`\g__graphics_track_int` Used to track the object number associated with each graphic.

```

1906 \int_new:N \g__graphics_track_int

```

(End definition for `\g__graphics_track_int`.)

The special syntax depends on the file type. There is a difference in how PDF graphics are best handled between `dviPDFfmX` and `XYTeX`: for the latter it is better to use the primitive route. The relevant code for that is included later in this file.

```

\_graphics_backend_include_eps:n
\_graphics_backend_include_ps:n
\_graphics_backend_include_jpg:n
\_graphics_backend_include_jpeg:n
\_graphics_backend_include_pdf:n
\_graphics_backend_include_png:n
\_graphics_backend_include_bmp:n
\_graphics_backend_include_auxi:nn
\_graphics_backend_include_auxii:nnn
\_graphics_backend_include_auxiii:nnn
1907 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1908 {
1909   \_kernel_backend_literal:x
1910   {
1911     PSfile = #1 \c_space_tl
1912     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1913     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1914     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1915     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1916   }
1917 }
1918 \cs_new_eq:NN \_graphics_backend_include_ps:n \_graphics_backend_include_eps:n
1919 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1920 { \_graphics_backend_include_auxi:nn {#1} { image } }
1921 \cs_new_eq:NN \_graphics_backend_include_jpeg:n \_graphics_backend_include_jpg:n
1922 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1923 \cs_new_eq:NN \_graphics_backend_include_bmp:n \_graphics_backend_include_jpg:n
1924 <*dviPDFfmX>
1925 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1926 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1927 </dviPDFfmX>

```

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.

```

1928 \cs_new_protected:Npn \_graphics_backend_include_auxi:nn #1#2
1929 {
1930   \_graphics_backend_include_auxii:nnn
1931   {
1932     \tl_if_empty:NF \l__graphics_pagebox_tl
1933     { : \l__graphics_pagebox_tl }
1934     \int_compare:nNnT \l__graphics_page_int > 1
1935     { :P \int_use:N \l__graphics_page_int }
1936     \tl_if_empty:NF \l__graphics_decodearray_str
1937     { :D \l__graphics_decodearray_str }
1938     \bool_if:NT \l__graphics_interpolate_bool
1939     { :I }
1940   }
1941   {#1} {#2}

```

```

1942 }
1943 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1944 {
1945   \int_if_exist:cTF { c__graphics_ #2#1 _int }
1946   {
1947     \__kernel_backend_literal:x
1948     { pdf:useobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
1949   }
1950   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1951 }
1952 \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!

```

1953 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1954 {
1955   \int_gincr:N \g__graphics_track_int
1956   \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
1957   \__kernel_backend_literal:x
1958   {
1959     pdf:#3~
1960     @graphic \int_use:c { c__graphics_ #1#2 _int } ~
1961     \int_compare:nNnT \l__graphics_page_int > 1
1962     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
1963     \tl_if_empty:NF \l__graphics_pagebox_tl
1964     {
1965       pagebox ~ \l__graphics_pagebox_tl \c_space_tl
1966       bbox ~
1967         \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1968         \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1969         \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1970         \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
1971     }
1972     (#1)
1973     \bool_lazy_or:nnT
1974     { \l__graphics_interpolate_bool }
1975     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1976     {
1977       <<
1978         \tl_if_empty:NF \l__graphics_decodearray_str
1979         { /Decode~[ \l__graphics_decodearray_str ] }
1980         \bool_if:NT \l__graphics_interpolate_bool
1981         { /Interpolate~true> }
1982       >>
1983     }
1984   }
1985 }

```

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

`__graphics_backend_get_pagecount:n`

```

1986 < *dvipdfmx >
1987 \__graphics_backend_loaded:n
1988 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }

```

```
1989 </dviptfm>
```

(End definition for `_graphics_backend_get_pagecount:n`.)

```
1990 </dviptfm | xetex>
```

5.4 X_YT_EX backend

```
1991 <*xetex>
```

For X_YT_EX, 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 X_YT_EX 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_jpeg:n
\_graphics_backend_getbb_pdf:n
\_graphics_backend_getbb_png:n
\_graphics_backend_getbb_bmp:n
1992 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1993 {
1994   \int_zero:N \l__graphics_page_int
1995   \tl_clear:N \l__graphics_pagebox_tl
1996   \_graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1997 }
1998 \cs_new_eq:NN \_graphics_backend_getbb_jpeg:n \_graphics_backend_getbb_jpg:n
1999 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
2000 \cs_new_eq:NN \_graphics_backend_getbb_bmp:n \_graphics_backend_getbb_jpg:n
2001 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
2002 {
2003   \tl_clear:N \l__graphics_decodearray_str
2004   \bool_set_false:N \l__graphics_interpolate_bool
2005   \_graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2006 }
2007 \cs_new_protected:Npn \_graphics_backend_getbb_auxi:nN #1#2
2008 {
2009   \int_compare:nNnTF \l__graphics_page_int > 1
2010     { \_graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2011     { \_graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2012 }
2013 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:nnN #1#2#3
2014 { \_graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2015 \cs_generate_variant:Nn \_graphics_backend_getbb_auxii:nnN { V }
2016 \cs_new_protected:Npn \_graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2017 {
2018   \tl_if_empty:NTF \l__graphics_pagebox_tl
2019     { \_graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2020     { \_graphics_backend_getbb_auxv:nNnn }
2021     {#1} #2 {#3} {#4}
2022 }
2023 \cs_new_protected:Npn \_graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2024 {
2025   \use:x
2026   {
2027     \_graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2028     {
2029       #5
2030       \tl_if_blank:nF {#1}
2031         { \c_space_tl \_graphics_backend_getbb_pagebox:w #1 }
2032     }
2033   }
```

```

2033     }
2034   }
2035   \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2036   \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2037     {
2038       \__graphics_bb_restore:nF {#1#3}
2039       { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2040     }
2041   \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2042     {
2043       \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
2044       \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2045       \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2046       \__graphics_bb_save:n {#1#3}
2047     }
2048   \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

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

```

2049   \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2050     {
2051       \tex_XeTeXpdffile:D #1 ~
2052       \int_compare:nNnT \l__graphics_page_int > 0
2053         { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2054       \exp_after:wN \__graphics_backend_getbb_pagebox:w \l__graphics_pagebox_tl
2055     }

```

(End definition for __graphics_backend_include_pdf:n.)

__graphics_backend_get_pagecount:n Very little to do here other than cover the case of a non-PDF file.

```

2056   \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
2057     {
2058       \int_const:cn { c__graphics_#1 _pages_int }
2059       {
2060         \int_max:nn
2061           { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2062           { 1 }
2063       }
2064     }

```

(End definition for __graphics_backend_get_pagecount:n.)

2065 </xetex>

5.5 dvisvgm backend

2066 <*dvisvgm>

\l_graphics_search_ext_seq

```

2067   \__graphics_backend_loaded:n
2068     {

```

```

2069 \seq_set_from_clist:Nn
2070 \l_graphics_search_ext_seq
2071 { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }
2072 }

```

(End definition for `\l_graphics_search_ext_seq`. This variable is documented on page ??.)

This is relatively similar to reading bounding boxes for `.eps` files. Life is though made more tricky as we cannot pick a single line for the data. So we have to loop until we collect up both height and width. To do that, we can use a marker value. We also have to allow for the default units of the lengths: they are big points and may be omitted.

```

\_graphics_backend_getbb_svg:n
\_graphics_backend_getbb_svg_auxi:nNn
\_graphics_backend_getbb_svg_auxii:w
\_graphics_backend_getbb_svg_auxiii:Nw
\_graphics_backend_getbb_svg_auxiv:Nw
\_graphics_backend_getbb_svg_auxv:Nw
\_graphics_backend_getbb_svg_auxvi:Nn
\_graphics_backend_getbb_svg_auxvii:w

```

```

2073 \cs_new_protected:Npn \_graphics_backend_getbb_svg:n #1
2074 {
2075   \__graphics_bb_restore:nF {#1}
2076   {
2077     \ior_open:Nn \l__graphics_internal_ior {#1}
2078     \ior_if_eof:NTF \l__graphics_internal_ior
2079       { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2080     {
2081       \dim_zero:N \l__graphics_llx_dim
2082       \dim_zero:N \l__graphics_lly_dim
2083       \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2084       \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2085       \ior_str_map_inline:Nn \l__graphics_internal_ior
2086         {
2087           \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2088             {
2089               \_graphics_backend_getbb_svg_auxi:nNn
2090                 { width } \l__graphics_urx_dim {##1}
2091             }
2092           \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2093             {
2094               \_graphics_backend_getbb_svg_auxi:nNn
2095                 { height } \l__graphics_ury_dim {##1}
2096             }
2097           \bool_lazy_and:nnF
2098             { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2099             { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2100             { \ior_map_break: }
2101         }
2102       \__graphics_bb_save:n {#1}
2103     }
2104     \ior_close:N \l__graphics_internal_ior
2105   }
2106 }
2107 \cs_new_protected:Npn \_graphics_backend_getbb_svg_auxi:nNn #1#2#3
2108 {
2109   \use:x
2110   {
2111     \cs_set_protected:Npn \_graphics_backend_getbb_svg_auxii:w
2112       ###1 \tl_to_str:n {#1} = ####2 \tl_to_str:n {#1} = ####3
2113     \s__graphics_stop
2114   }
2115 }

```

```

2116     \tl_if_blank:nF {##2}
2117     {
2118         \peek_remove_spaces:n
2119         {
2120             \peek_meaning:NTF ' % '
2121             { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2122             {
2123                 \peek_meaning:NTF " % "
2124                 { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2125                 { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2126             }
2127         }
2128         ##2 \s__graphics_stop
2129     }
2130 }
2131 \use:x
2132 {
2133     \__graphics_backend_getbb_svg_auxii:w #3
2134     \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2135     \s__graphics_stop
2136 }
2137 }
2138 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2139 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop
2140 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2141 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2142 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2143 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2144 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2145 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2146 {
2147     \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2148     \l__graphics_internal_dim #2 bp \scan_stop:
2149     \dim_set_eq:NN #1 \l__graphics_internal_dim
2150 }
2151 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvii:w #1 \scan_stop: { }

```

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

`__graphics_backend_getbb_eps:n` Simply use the generic function.

```

\__graphics_backend_getbb_ps:n
2152 \__graphics_backend_loaded:n
2153 {
2154     \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2155     \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
2156 }

```

(End definition for `__graphics_backend_getbb_eps:n` and `__graphics_backend_getbb_ps:n`.)

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

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_jpeg:n
2157 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2158 {
2159     \int_zero:N \l__graphics_page_int
2160     \tl_clear:N \l__graphics_pagebox_tl
2161     \__graphics_extract_bb:n {#1}
2162 }

```

```

2163 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2164 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n

```

(End definition for __graphics_backend_getbb_png:n, __graphics_backend_getbb_jpg:n, and __graphics_backend_getbb_jpeg:n.)

__graphics_backend_getbb_pdf:n Same as for dvipdfmx: use the generic function

```

2165 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2166 {
2167   \tl_clear:N \l__graphics_decodearray_str
2168   \bool_set_false:N \l__graphics_interpolate_bool
2169   \__graphics_extract_bb:n {#1}
2170 }

```

(End definition for __graphics_backend_getbb_pdf:n.)

__graphics_backend_include_eps:n __graphics_backend_include_ps:n __graphics_backend_include_pdf:n __graphics_backend_include:n The special syntax is relatively clear here: remember we need PostScript sizes here. (This is the same as the dvips code.)

```

2171 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2172 { \__graphics_backend_include:nn { PSfile } {#1} }
2173 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
2174 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2175 { \__graphics_backend_include:nn { pdffile } {#1} }
2176 \cs_new_protected:Npn \__graphics_backend_include:n #1#2
2177 {
2178   \__kernel_backend_literal:x
2179   {
2180     #1 = #2 \c_space_tl
2181     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2182     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2183     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2184     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2185   }
2186 }

```

(End definition for __graphics_backend_include_eps:n and others.)

__graphics_backend_include_svg:n __graphics_backend_include_png:n __graphics_backend_include_jpg:n __graphics_backend_include_jpeg:n __graphics_backend_include_dequote:w The backend here has built-in support for basic graphic inclusion (see dvisvgm.def for a more complex approach, needed if clipping, etc., is covered at the graphic backend level). We have to deal with the fact that the image reference point is at the top, so there is a need for a vertical shift to put it in the right place. The other 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.

```

2187 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2188 {
2189   \box_move_up:nn { \l__graphics_ury_dim }
2190   {
2191     \hbox:n
2192     {
2193       \__kernel_backend_literal:x
2194       {
2195         dvisvgm:img~
2196         \dim_to_decimal:n { \l__graphics_urx_dim } ~
2197         \dim_to_decimal:n { \l__graphics_ury_dim } ~

```

```

2198         \_graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2199     }
2200 }
2201 }
2202 }
2203 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_svg:n
2204 \cs_new_eq:NN \_graphics_backend_include_jpeg:n \_graphics_backend_include_svg:n
2205 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_svg:n
2206 \cs_new:Npn \_graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2207     {#2}

```

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

`_graphics_backend_get_pagecount:n`

```

2208 \_graphics_backend_loaded:n
2209 { \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n }

```

(End definition for `_graphics_backend_get_pagecount:n`.)

```

2210 </dvisvgm>
2211 </package>

```

6 I3backend-pdf Implementation

```

2212 <*package>
2213 <@=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`

```

2214 \box_new:N \l__pdf_internal_box

```

(End definition for `\l__pdf_internal_box`.)

6.2 dvips backend

```

2215 <*dvips>

```

`_pdf_backend_pdfmark:n` Used often enough it should be a separate function.

```

\_pdf_backend_pdfmark:x
2216 \cs_new_protected:Npn \_pdf_backend_pdfmark:n #1
2217     { \_kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2218 \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 2219 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2220 { \_pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2221 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2222 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }

(End definition for \_pdf_backend_catalog_gput:nn and \_pdf_backend_info_gput:nn.)
```

6.2.2 Objects

```
\g__pdf_backend_object_int For tracking objects.
2223 \int_new:N \g__pdf_backend_object_int

(End definition for \g__pdf_backend_object_int.)

\_pdf_backend_object_new:n
\_pdf_backend_object_ref:n 2224 \cs_new_protected:Npn \_pdf_backend_object_new:n #1
2225 {
2226   \int_gincr:N \g__pdf_backend_object_int
2227   \int_const:cn
2228   { c__pdf_object_ \tl_to_str:n {#1} _int }
2229   { \g__pdf_backend_object_int }
2230 }
2231 \cs_new:Npn \_pdf_backend_object_ref:n #1
2232 { { pdf.obj \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } } }

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

```
\_pdf_backend_object_write:nnn This is where we choose the actual type: some work to get things right. To allow code
\_pdf_backend_object_write:nnx sharing with the anonymous version, we use an auxiliary.
\_pdf_backend_object_write_aux:nnn 2233 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
\_pdf_backend_object_write_array:nn 2234 {
\_pdf_backend_object_write_dict:nn 2235   \_pdf_backend_object_write_aux:nnn
\_pdf_backend_object_write_fstream:nn 2236   { \_pdf_backend_object_ref:n {#1} }
\_pdf_backend_object_write_stream:nn 2237   {#2} {#3}
\_pdf_backend_object_write_stream:nnn 2238 }
2239 \cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nnx }
2240 \cs_new_protected:Npn \_pdf_backend_object_write_aux:nnn #1#2#3
2241 {
2242   \_pdf_backend_pdfmark:x
2243   {
2244     /_objdef ~ #1
2245     /type
2246     \str_case:nn {#2}
2247     {
2248       { array } { /array }
2249       { dict } { /dict }
2250       { fstream } { /stream }
2251       { stream } { /stream }
2252     }
2253     /OBJ
2254   }
2255   \use:c { __pdf_backend_object_write_ #2 :nn } {#1} {#3}
```

```

2256 }
2257 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2258 {
2259   \__pdf_backend_pdfmark:x
2260   { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2261 }
2262 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2263 {
2264   \__pdf_backend_pdfmark:x
2265   { #1 << \exp_not:n {#2} >> /PUT }
2266 }
2267 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2268 {
2269   \exp_args:Nx
2270   \__pdf_backend_object_write_fstream:nnn {#1} #2
2271 }
2272 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2273 {
2274   \__kernel_backend_postscript:n
2275   {
2276     SDict ~ begin ~
2277     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2278     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2279     end
2280   }
2281 }
2282 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2283 {
2284   \exp_args:Nx
2285   \__pdf_backend_object_write_stream:nnn {#1} #2
2286 }
2287 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2288 {
2289   \__kernel_backend_postscript:n
2290   {
2291     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2292     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2293   }
2294 }

```

(End definition for __pdf_backend_object_write:nnn and others.)

__pdf_backend_object_now:nn No anonymous objects, so things are done manually.

```

\__pdf_backend_object_now:nx
2295 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2296 {
2297   \int_gincr:N \g__pdf_backend_object_int
2298   \__pdf_backend_object_write_aux:nnn
2299   { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2300   {#1} {#2}
2301 }
2302 \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.

```

2303 \cs_new:Npn \_pdf_backend_object_last:
2304   { { 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.

```

2305 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2306   { { 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.

```

2307 \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.

```

2308 \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.

```

2309 \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 L^AT_EX 2_ε picture of zero size). Once the data is collected, use it to set up the annotation border.

```

2310 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2311   {
2312     \exp_args:Nf \_pdf_backend_annotation_aux:nnnn
2313       { \dim_eval:n {#1} } {#2} {#3} {#4}
2314   }
2315 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2316   {
2317     \box_move_down:nn {#3}
2318     { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2319     \box_move_up:nn {#2}
2320     {
2321       \hbox:n
2322         {
2323           \_kernel_kern:n {#1}
2324           \_kernel_backend_postscript:n { pdf.save.ur }
2325           \_kernel_kern:n { -#1 }
2326         }

```

```

2327     }
2328     \int_gincr:N \g__pdf_backend_object_int
2329     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2330     \__pdf_backend_pdfmark:x
2331     {
2332         /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2333         pdf.rect
2334         #4 ~
2335         /ANN
2336     }
2337 }

```

(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.

```

2338 \cs_new:Npn \__pdf_backend_annotation_last:
2339 { { 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.

```

2340 \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.

```

2341 \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.

```

2342 \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.

```

2343 \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.

```

2344 \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.

```

2345 \tl_new:N \l__pdf_breaklink_pdfmark_tl
2346 \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.

```

2347 \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.

```
2348 \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`.

Notice that the link setup here uses `/Action` not `/A`. That is because Distiller *requires* this trigger word, rather than a “raw” PDF dictionary key (Ghostscript can handle either form).

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 generic mode are still to re-examine.

```
2349 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2350 {
2351   \_pdf_backend_link_begin:nw
2352     { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2353 }
2354 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2355 { \_pdf_backend_link_begin:nw {#1#2} }
2356 \cs_new_protected:Npn \_pdf_backend_link_begin:nw #1
2357 {
2358   \bool_if:NF \g__pdf_backend_link_bool
2359     { \_pdf_backend_link_begin_aux:nw {#1} }
2360 }
```

The definition of `pdf.link.dict` here is needed as there is code in the PostScript headers for breaking links, and that can only work with this available.

```
2361 \cs_new_protected:Npn \_pdf_backend_link_begin_aux:nw #1
2362 {
2363   \bool_gset_true:N \g__pdf_backend_link_bool
2364   \_kernel_backend_postscript:n
2365     { /pdf.link.dict ( #1 ) def }
2366   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2367   \_pdf_backend_link_sf_save:
2368   \mode_if_math:TF
2369     { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2370     { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2371   \hbox_set:Nw \l__pdf_backend_content_box
2372     \_pdf_backend_link_sf_restore:
2373     \bool_if:NT \g__pdf_backend_link_math_bool
2374       { \c_math_toggle_token }
2375 }
2376 \cs_new_protected:Npn \_pdf_backend_link_end:
```

```

2377 {
2378   \bool_if:NT \g__pdf_backend_link_bool
2379     { \__pdf_backend_link_end_aux: }
2380 }
2381 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2382 {
2383   \bool_if:NT \g__pdf_backend_link_math_bool
2384     { \c_math_toggle_token }
2385   \__pdf_backend_link_sf_save:
2386   \hbox_set_end:
2387   \__pdf_backend_link_minima:
2388   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2389   \exp_args:Nx \__pdf_backend_link_outerbox:n
2390     {
2391       \int_if_odd:nTF { \value { page } }
2392         { \oddsidemargin }
2393         { \evensidemargin }
2394     }
2395   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2396     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2397   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2398   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2399   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2400   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2401     {
2402       \hbox:n
2403         { \__kernel_backend_postscript:n { pdf.save.linkur } }
2404     }
2405   \int_gincr:N \g__pdf_backend_object_int
2406   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2407   \__kernel_backend_postscript:x
2408     {
2409       mark
2410       /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2411       \g__pdf_backend_link_dict_tl \c_space_tl
2412       pdf.rect
2413       /ANN ~ \l__pdf_breaklink_pdfmark_tl
2414     }
2415   \__pdf_backend_link_sf_restore:
2416   \bool_gset_false:N \g__pdf_backend_link_bool
2417 }
2418 \cs_new_protected:Npn \__pdf_backend_link_minima:
2419 {
2420   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2421   \__kernel_backend_postscript:x
2422     {
2423       /pdf.linkdp.pad ~
2424       \dim_to_decimal:n
2425         {
2426           \dim_max:nn
2427             {
2428               \box_dp:N \l__pdf_backend_model_box
2429               - \box_dp:N \l__pdf_backend_content_box
2430             }

```

```

2431         { Opt }
2432     } ~
2433     pdf.pt.dvi ~ def
2434 /pdf.linkht.pad ~
2435 \dim_to_decimal:n
2436 {
2437     \dim_max:nn
2438     {
2439         \box_ht:N \l__pdf_backend_model_box
2440         - \box_ht:N \l__pdf_backend_content_box
2441     }
2442     { Opt }
2443 } ~
2444 pdf.pt.dvi ~ def
2445 }
2446 }
2447 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2448 {
2449     \__kernel_backend_postscript:x
2450     {
2451         /pdf.outerbox
2452         [
2453             \dim_to_decimal:n {#1} ~
2454             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2455             \dim_to_decimal:n { #1 + \textwidth } ~
2456             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2457         ]
2458         [ exch { pdf.pt.dvi } forall ] def
2459         /pdf.baselineskip ~
2460         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2461         { pdf.pt.dvi ~ def }
2462         { pop ~ pop }
2463         ifelse
2464     }
2465 }
2466 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2467 {
2468     \int_gset:Nn \g__pdf_backend_link_sf_int
2469     {
2470         \mode_if_horizontal:TF
2471         { \tex_spacefactor:D }
2472         { 0 }
2473     }
2474 }
2475 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2476 {
2477     \mode_if_horizontal:T
2478     {
2479         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2480         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2481     }
2482 }

```

(End definition for `__pdf_backend_link_begin_goto:nw` and others. These functions are documented on page ??.)

`\@makecol@hook` Hooks to allow link breaking: something will be needed in format mode at some stage. At present this code is disabled as there is an open question about the name of the hook: to be resolved at the L^AT_EX 2_ε end.

```

2483 \use_none:n
2484 {
2485   \cs_if_exist:NT \@makecol@hook
2486     {
2487       \tl_put_right:Nn \@makecol@hook
2488         {
2489           \box_if_empty:NF \@cclv
2490             {
2491               \vbox_set:Nn \@cclv
2492                 {
2493                   \__kernel_backend_postscript:n
2494                     {
2495                       pdf.globaldict /pdf.brokenlink.rect ~ known
2496                         { pdf.bordertracking.continue }
2497                       if
2498                     }
2499                   \vbox_unpack_drop:N \@cclv
2500                   \__kernel_backend_postscript:n
2501                     { pdf.bordertracking.endpage }
2502                 }
2503             }
2504         }
2505       \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2506       \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2507       \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2508     }
2509 }

```

(End definition for \@makecol@hook. This function is documented on page ??.)

`__pdf_backend_link_last:` The same as annotations, but with a custom integer.

```

2510 \cs_new:Npn \__pdf_backend_link_last:
2511   { { 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.

```

2512 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2513   {
2514     \__kernel_backend_postscript:x
2515       {
2516         /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2517       }
2518   }

```

(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. `ftr` without rule spec doesn't work, so it falls back to `/Fit` here.

`__pdf_backend_destination:nmnn`

`__pdf_backend_destination_aux:nmnn`

```

2519 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2520 {
2521   \__kernel_backend_postscript:n { pdf.dest.anchor }
2522   \__pdf_backend_pdfmark:x
2523   {
2524     /View
2525     [
2526       \str_case:nnF {#2}
2527       {
2528         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2529         { fit } { /Fit }
2530         { fitb } { /FitB }
2531         { fitbh } { /FitBH ~ pdf.dest.y }
2532         { fitbv } { /FitBV ~ pdf.dest.x }
2533         { fith } { /FitH ~ pdf.dest.y }
2534         { fitv } { /FitV ~ pdf.dest.x }
2535         { fitr } { /Fit }
2536       }
2537       {
2538         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2539       }
2540     ]
2541     /Dest ( \exp_not:n {#1} ) cvn
2542     /DEST
2543   }
2544 }
2545 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2546 {
2547   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2548   { \dim_eval:n {#2} } {#1} {#3} {#4}
2549 }
2550 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2551 {
2552   \vbox_to_zero:n
2553   {
2554     \__kernel_kern:n {#4}
2555     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2556     \tex_vss:D
2557   }
2558   \__kernel_kern:n {#1}
2559   \vbox_to_zero:n
2560   {
2561     \__kernel_kern:n { -#3 }
2562     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2563     \tex_vss:D
2564   }
2565   \__kernel_kern:n { -#1 }
2566   \__pdf_backend_pdfmark:n
2567   {
2568     /View
2569     [
2570       /FitR ~
2571       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2572       pdf.urx ~ pdf.ury ~ pdf.dest2device

```

```

2573     ]
2574     /Dest ( #2 ) cvn
2575     /DEST
2576   }
2577 }

```

(End definition for `_pdf_backend_destination:nn`, `_pdf_backend_destination:nnnn`, and `_pdf_backend_destination_aux:nnnn`.)

6.2.4 Structure

Doable for the usual ps2pdf method.

```

\_pdf_backend_compresslevel:n
\_pdf_backend_compress_objects:n
2578 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2579 {
2580   \int_compare:nNnT {#1} = 0
2581   {
2582     \_kernel_backend_literal_postscript:n
2583     {
2584       /setdistillerparams ~ where
2585       { pop << /CompressPages ~ false >> setdistillerparams }
2586       if
2587     }
2588   }
2589 }
2590 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2591 {
2592   \bool_if:nF {#1}
2593   {
2594     \_kernel_backend_literal_postscript:n
2595     {
2596       /setdistillerparams ~ where
2597       { pop << /CompressStreams ~ false >> setdistillerparams }
2598       if
2599     }
2600   }
2601 }

```

(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
2602 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2603 {
2604   \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2605 }
2606 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2607 {
2608   \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2609 }

```

(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:
2610 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2611 \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:`

```
2612 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2613   { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2614 \cs_new_protected:Npn \_pdf_backend_emc:
2615   { \_pdf_backend_pdfmark:n { /EMC } }

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

2616 </dvips>
```

6.3 LuaTeX and pdfTeX backend

```
2617 <*luatex | pdftex>
```

6.3.1 Annotations

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

```
2618 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2619   {
2620     <*luatex>
2621     \tex_pdfextension:D annot ~
2622     </luatex>
2623     <*pdftex>
2624     \tex_pdfannot:D
2625     </pdftex>
2626     width ~ \dim_eval:n {#1} ~
2627     height ~ \dim_eval:n {#2} ~
2628     depth ~ \dim_eval:n {#3} ~
2629     {#4}
2630   }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

`_pdf_backend_annotation_last:` A tiny amount of extra data gets added here; we use x-type expansion to get the space in the right place and form. The “extra” space in the LuaTeX version is *required* as it is consumed in finding the end of the keyword.

```
2631 \cs_new:Npx \_pdf_backend_annotation_last:
2632   {
2633     \exp_not:N \int_value:w
2634     <*luatex>
2635     \exp_not:N \tex_pdffeedback:D lastannot ~
2636     </luatex>
2637     <*pdftex>
2638     \exp_not:N \tex_pdflastannot:D
2639     </pdftex>
2640     \c_space_tl 0 ~ R
2641   }
```

(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:
2642 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2643   { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2644 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
```

```

2645 { \_pdf_backend_link_begin:nnw {#1} { user } {#2} }
2646 \cs_new_protected:Npn \_pdf_backend_link_begin:nnw #1#2#3
2647 {
2648 <*luatex>
2649   \tex_pdfextension:D startlink ~
2650 </luatex>
2651 <*pdftex>
2652   \tex_pdfstartlink:D
2653 </pdftex>
2654   attr {#1}
2655   #2 {#3}
2656 }
2657 \cs_new_protected:Npn \_pdf_backend_link_end:
2658 {
2659 <*luatex>
2660   \tex_pdfextension:D endlink \scan_stop:
2661 </luatex>
2662 <*pdftex>
2663   \tex_pdfendlink:D
2664 </pdftex>
2665 }

```

(End definition for _pdf_backend_link_begin_goto:nnw and others.)

_pdf_backend_link_last: Formatted for direct use.

```

2666 \cs_new:Npx \_pdf_backend_link_last:
2667 {
2668   \exp_not:N \int_value:w
2669 <*luatex>
2670   \exp_not:N \tex_pdffeedback:D lastlink ~
2671 </luatex>
2672 <*pdftex>
2673   \exp_not:N \tex_pdflastlink:D
2674 </pdftex>
2675   \c_space_tl 0 ~ R
2676 }

```

(End definition for _pdf_backend_link_last:.)

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

```

2677 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2678 {
2679 <*luatex>
2680   \tex_pdfvariable:D linkmargin
2681 </luatex>
2682 <*pdftex>
2683   \tex_pdflinkmargin:D
2684 </pdftex>
2685   \dim_eval:n {#1} \scan_stop:
2686 }

```

(End definition for _pdf_backend_link_margin:n.)

`_pdf_backend_destination:nn`
`_pdf_backend_destination:nnnn`

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.

```

2687 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2688 {
2689 <*luatex>
2690   \tex_pdfextension:D dest ~
2691 </luatex>
2692 <*pdftex>
2693   \tex_pdfdest:D
2694 </pdftex>
2695   name {#1}
2696   \str_case:nnF {#2}
2697   {
2698     { xyz } { xyz }
2699     { fit } { fit }
2700     { fitb } { fitb }
2701     { fitbh } { fitbh }
2702     { fitbv } { fitbv }
2703     { fith } { fith }
2704     { fitv } { fitv }
2705     { fitr } { fitr }
2706   }
2707   { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2708   \scan_stop:
2709 }
2710 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2711 {
2712 <*luatex>
2713   \tex_pdfextension:D dest ~
2714 </luatex>
2715 <*pdftex>
2716   \tex_pdfdest:D
2717 </pdftex>
2718   name {#1}
2719   fitr ~
2720   width \dim_eval:n {#2} ~
2721   height \dim_eval:n {#3} ~
2722   depth \dim_eval:n {#4} \scan_stop:
2723 }

```

(End definition for `_pdf_backend_destination:nn` and `_pdf_backend_destination:nnnn`.)

6.3.2 Catalogue entries

`_pdf_backend_catalog_gput:nn`
`_pdf_backend_info_gput:nn`

```

2724 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2725 {
2726 <*luatex>
2727   \tex_pdfextension:D catalog
2728 </luatex>
2729 <*pdftex>
2730   \tex_pdfcatalog:D
2731 </pdftex>

```

```

2732     { / #1 ~ #2 }
2733   }
2734 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2735   {
2736   <*luatex>
2737     \tex_pdfextension:D info
2738   </luatex>
2739   <*pdftex>
2740     \tex_pdfinfo:D
2741   </pdftex>
2742     { / #1 ~ #2 }
2743   }

```

(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.

```

2744 \prop_new:N \g__pdf_backend_object_prop

```

(End definition for \g__pdf_backend_object_prop.)

__pdf_backend_object_new:n Declaring objects means reserving at the PDF level plus starting tracking.

__pdf_backend_object_ref:n

```

2745 \cs_new_protected:Npn \__pdf_backend_object_new:n #1
2746   {
2747   <*luatex>
2748     \tex_pdfextension:D obj ~
2749   </luatex>
2750   <*pdftex>
2751     \tex_pdfobj:D
2752   </pdftex>
2753     reserveobjnum ~
2754     \int_const:cn
2755     { c__pdf_object_ \tl_to_str:n {#1} _int }
2756   <*luatex>
2757     { \tex_pdffeedback:D lastobj }
2758   </luatex>
2759   <*pdftex>
2760     { \tex_pdflastobj:D }
2761   </pdftex>
2762   }

```

```

2763 \cs_new:Npn \__pdf_backend_object_ref:n #1
2764   { \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

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

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

__pdf_backend_object_write:nnx

__pdf_backend_object_write:nn

__pdf_exp_not_i:nn

__pdf_exp_not_ii:nn

```

2765 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2766   {
2767   <*luatex>
2768     \tex_immediate:D \tex_pdfextension:D obj ~
2769   </luatex>
2770   <*pdftex>
2771     \tex_immediate:D \tex_pdfobj:D
2772   </pdftex>

```

```

2773     useobjnum ~
2774     \int_use:c
2775     { c__pdf_object_ \tl_to_str:n {#1} _int }
2776     \__pdf_backend_object_write:nn {#2} {#3}
2777   }
2778 \cs_new:Npn \__pdf_backend_object_write:nn #1#2
2779 {
2780   \str_case:nn {#1}
2781   {
2782     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2783     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2784     { fstream }
2785     {
2786       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2787       file ~ { \__pdf_exp_not_ii:nn #2 }
2788     }
2789     { stream }
2790     {
2791       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2792       { \__pdf_exp_not_ii:nn #2 }
2793     }
2794   }
2795 }
2796 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nnx }
2797 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2798 \cs_new:Npn \__pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

```

(End definition for __pdf_backend_object_write:nnn and others.)

__pdf_backend_object_now:nn Much like writing, but direct creation.

```

\__pdf_backend_object_now:nx
2799 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2800 {
2801   <*luatex>
2802   \tex_immediate:D \tex_pdfextension:D obj ~
2803   </luatex>
2804   <*pdftex>
2805   \tex_immediate:D \tex_pdfobj:D
2806   </pdftex>
2807   \__pdf_backend_object_write:nn {#1} {#2}
2808 }
2809 \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.

```

2810 \cs_new:Npx \__pdf_backend_object_last:
2811 {
2812   \exp_not:N \int_value:w
2813   <*luatex>
2814   \exp_not:N \tex_pdffeedback:D lastobj ~
2815   </luatex>
2816   <*pdftex>
2817   \exp_not:N \tex_pdflastobj:D
2818   </pdftex>
2819   \c_space_tl 0 ~ R

```

```
2820 }
(End definition for \_pdf_backend_object_last:.)
```

_pdf_backend_pageobject_ref:n The usual wrapper situation; the three spaces here are essential.

```
2821 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2822 {
2823   \exp_not:N \int_value:w
2824   <*luatex>
2825     \exp_not:N \tex_pdffeedback:D pageref
2826   </luatex>
2827   <*pdftex>
2828     \exp_not:N \tex_pdfpageref:D
2829   </pdftex>
2830   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2831 }
```

(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
\_pdf_backend_objcompresslevel:n
2832 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2833 {
2834   \tex_global:D
2835   <*luatex>
2836     \tex_pdfvariable:D compresslevel
2837   </luatex>
2838   <*pdftex>
2839     \tex_pdfcompresslevel:D
2840   </pdftex>
2841     \int_value:w \int_eval:n {#1} \scan_stop:
2842 }
2843 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2844 {
2845   \bool_if:nTF {#1}
2846     { \_pdf_backend_objcompresslevel:n { 2 } }
2847     { \_pdf_backend_objcompresslevel:n { 0 } }
2848 }
2849 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2850 {
2851   \tex_global:D
2852   <*luatex>
2853     \tex_pdfvariable:D objcompresslevel
2854   </luatex>
2855   <*pdftex>
2856     \tex_pdfobjcompresslevel:D
2857   </pdftex>
2858     #1 \scan_stop:
2859 }
```

(End definition for _pdf_backend_compresslevel:n, _pdf_backend_compress_objects:n, and _pdf_backend_objcompresslevel:n.)

`_pdf_backend_version_major_gset:n`
`_pdf_backend_version_minor_gset:n`

The availability of the primitive is not universal, so we have to test at load time.

```
2860 \cs_new_protected:Npx \_pdf_backend_version_major_gset:n #1
2861 {
2862 <*luatex>
2863   \int_compare:nNnT \tex luatexversion:D > { 106 }
2864   {
2865     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2866     \exp_not:N \int_eval:n {#1} \scan_stop:
2867   }
2868 </luatex>
2869 <*pdftex>
2870   \cs_if_exist:NT \tex_pdfmajorversion:D
2871   {
2872     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2873     \exp_not:N \int_eval:n {#1} \scan_stop:
2874   }
2875 </pdftex>
2876 }
2877 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2878 {
2879   \tex_global:D
2880 <*luatex>
2881   \tex_pdfvariable:D minorversion
2882 </luatex>
2883 <*pdftex>
2884   \tex_pdfminorversion:D
2885 </pdftex>
2886   \int_eval:n {#1} \scan_stop:
2887 }
```

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

`_pdf_backend_version_major:`
`_pdf_backend_version_minor:`

As above.

```
2888 \cs_new:Npx \_pdf_backend_version_major:
2889 {
2890 <*luatex>
2891   \int_compare:nNnTF \tex luatexversion:D > { 106 }
2892   { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2893   { 1 }
2894 </luatex>
2895 <*pdftex>
2896   \cs_if_exist:NTF \tex_pdfmajorversion:D
2897   { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2898   { 1 }
2899 </pdftex>
2900 }
2901 \cs_new:Npn \_pdf_backend_version_minor:
2902 {
2903   \tex_the:D
2904 <*luatex>
2905   \tex_pdfvariable:D minorversion
2906 </luatex>
2907 <*pdftex>
2908   \tex_pdfminorversion:D
```

```

2909 </pdftex>
2910 }

```

(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>.

```

2911 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2912 { \_kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2913 \cs_new_protected:Npn \_pdf_backend_emc:
2914 { \_kernel_backend_literal_page:n { EMC } }

```

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

```

2915 </luatex | pdftex>

```

6.4 dvipdfmx backend

```

2916 <*dvipdfmx | xetex>

```

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

```

\_pdf_backend:x
2917 \cs_new_protected:Npx \_pdf_backend:n #1
2918 { \_kernel_backend_literal:n { pdf: #1 } }
2919 \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
2920 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2921 { \_pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2922 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2923 { \_pdf_backend:n { docinfo << /#1 ~ #2 >> } }

```

(End definition for `_pdf_backend_catalog_gput:nn` and `_pdf_backend_info_gput:nn`.)

6.4.2 Objects

`\g__pdf_backend_object_int` For tracking objects to allow finalisation.

```

\g__pdf_backend_object_prop
2924 \int_new:N \g__pdf_backend_object_int
2925 \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:n` Objects are tracked at the macro level, but we don't have to do anything at this stage.

```

\_pdf_backend_object_ref:n
2926 \cs_new_protected:Npn \_pdf_backend_object_new:n #1
2927 {
2928   \int_gincr:N \g__pdf_backend_object_int
2929   \int_const:cn
2930   { c__pdf_object_ \tl_to_str:n {#1} _int }
2931   { \g__pdf_backend_object_int }
2932 }
2933 \cs_new:Npn \_pdf_backend_object_ref:n #1
2934 { @pdf.obj \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } }

```

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

This is where we choose the actual type.

```
\_pdf_backend_object_write:nnn
\_pdf_backend_object_write:nnx
\_pdf_backend_object_write_array:nn
\_pdf_backend_object_write_dict:nn
\_pdf_backend_object_write_fstream:nn
\_pdf_backend_object_write_stream:nn
\_pdf_backend_object_write_stream:nnnn
2935 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
2936 {
2937   \use:c { \_pdf_backend_object_write_ #2 :nn }
2938   { \_pdf_backend_object_ref:n {#1} } {#3}
2939 }
2940 \cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nnx }
2941 \cs_new_protected:Npn \_pdf_backend_object_write_array:nn #1#2
2942 {
2943   \_pdf_backend:x
2944   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2945 }
2946 \cs_new_protected:Npn \_pdf_backend_object_write_dict:nn #1#2
2947 {
2948   \_pdf_backend:x
2949   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2950 }
2951 \cs_new_protected:Npn \_pdf_backend_object_write_fstream:nn #1#2
2952 { \_pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2953 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nn #1#2
2954 { \_pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2955 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nnnn #1#2#3#4
2956 {
2957   \_pdf_backend:x
2958   {
2959     #1 stream ~ #2 ~
2960     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2961   }
2962 }
```

(End definition for `_pdf_backend_object_write:nnn` and others.)

No anonymous objects with dvipdfmx so we have to give an object name.

```
\_pdf_backend_object_now:nn
\_pdf_backend_object_now:nx
2963 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2964 {
2965   \int_gincr:N \g__pdf_backend_object_int
2966   \exp_args:Nnx \use:c { \_pdf_backend_object_write_ #1 :nn }
2967   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2968   {#2}
2969 }
2970 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }
```

(End definition for `_pdf_backend_object_now:nn`.)

`_pdf_backend_object_last:`

```
2971 \cs_new:Npn \_pdf_backend_object_last:
2972 { @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 dvipdfmx/X_qTeX.

```
2973 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2974 { @page #1 }
```

(End definition for `_pdf_backend_pageobject_ref:n`.)

6.4.3 Annotations

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

```
2975 \int_new:N \g_pdf_backend_annotation_int
```

(End definition for `\g_pdf_backend_annotation_int`.)

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

```
2976 \cs_new_protected:Npn \_pdf_backend_annotation:nmnn #1#2#3#4
```

```
2977 {
2978   \int_gincr:N \g_pdf_backend_object_int
2979   \int_gset_eq:NN \g_pdf_backend_annotation_int \g_pdf_backend_object_int
2980   \_pdf_backend:x
2981   {
2982     ann ~ @pdf.obj \int_use:N \g_pdf_backend_object_int \c_space_tl
2983     width ~ \dim_eval:n {#1} ~
2984     height ~ \dim_eval:n {#2} ~
2985     depth ~ \dim_eval:n {#3} ~
2986     << /Type /Annot #4 >>
2987   }
2988 }
```

(End definition for `_pdf_backend_annotation:nmnn`.)

`_pdf_backend_annotation_last:`

```
2989 \cs_new:Npn \_pdf_backend_annotation_last:
2990 { @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.

```
2991 \int_new:N \g_pdf_backend_link_int
```

(End definition for `\g_pdf_backend_link_int`.)

`_pdf_backend_link_begin_goto:nmw` All created using the same internals.

`_pdf_backend_link_begin_user:nmw`

```
2992 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nmw #1#2
```

```
2993 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
```

`_pdf_backend_link_begin:n`

```
2994 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nmw #1#2
```

```
2995 { \_pdf_backend_link_begin:n { #1#2 } }
```

`_pdf_backend_link_end:`

```
2996 \cs_new_protected:Npx \_pdf_backend_link_end:n #1
```

```
2997 {
2998   \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int
2999   \_pdf_backend:x
3000   {
3001     bann ~
3002     @pdf.lnk
3003     \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int
3004     \c_space_tl
3005     <<
3006     /Type /Annot
3007     #1
3008     >>
3009   }
3010 }
3011 \cs_new_protected:Npn \_pdf_backend_link_end:
3012 { \_pdf_backend:n { eann } }
```

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

`_pdf_backend_link_last:` Available using the backend mechanism with a suitably-recent version.

```
3013 \cs_new:Npn \_pdf_backend_link_last:
3014 { @pdf.lnk \int_use:N \g_pdf_backend_link_int }
```

(End definition for `_pdf_backend_link_last:.`)

`_pdf_backend_link_margin:n` Pass to `dvipdfmx`.

```
3015 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
3016 { \_kernel_backend_literal:x { dvipdfmx.config-g~ \dim_eval:n {#1} } }
```

(End definition for `_pdf_backend_link_margin:n`.)

`_pdf_backend_destination:nn`
`_pdf_backend_destination:nmnn`
`_pdf_backend_destination_aux:nmnn`

Here, we need to turn the zoom into a scale. The method for `FitR` is from Alexander Grahn: the idea is to avoid needing to do any calculations in `TeX` by using the backend data for `@xpos` and `@ypos`. `/FitR` without rule spec doesn't work, so it falls back to `/Fit` here.

```
3017 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
3018 {
3019   \_pdf_backend:x
3020   {
3021     dest ~ ( \exp_not:n {#1} )
3022     [
3023       @thispage
3024       \str_case:nmF {#2}
3025       {
3026         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
3027         { fit } { /Fit }
3028         { fitb } { /FitB }
3029         { fitbh } { /FitBH }
3030         { fitbv } { /FitBV ~ @xpos }
3031         { fith } { /FitH ~ @ypos }
3032         { fitv } { /FitV ~ @xpos }
3033         { fitr } { /Fit }
3034       }
3035       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
3036     ]
3037   }
3038 }
3039 \cs_new_protected:Npn \_pdf_backend_destination:nmnn #1#2#3#4
3040 {
3041   \exp_args:Ne \_pdf_backend_destination_aux:nmnn
3042   { \dim_eval:n {#2} } {#1} {#3} {#4}
3043 }
3044 \cs_new_protected:Npn \_pdf_backend_destination_aux:nmnn #1#2#3#4
3045 {
3046   \vbox_to_zero:n
3047   {
3048     \_kernel_kern:n {#4}
3049     \hbox:n
3050     {
3051       \_pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
3052       \_pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }

```

```

3053     }
3054     \tex_vss:D
3055   }
3056   \__kernel_kern:n {#1}
3057   \vbox_to_zero:n
3058   {
3059     \__kernel_kern:n { -#3 }
3060     \hbox:n
3061     {
3062       \__pdf_backend:n
3063       {
3064         dest ~ (#2)
3065         [
3066           @thispage
3067           /FitR ~
3068           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
3069           @xpos ~ @ypos
3070         ]
3071       }
3072     }
3073     \tex_vss:D
3074   }
3075   \__kernel_kern:n { -#1 }
3076 }

```

(End definition for `__pdf_backend_destination:nn`, `__pdf_backend_destination:nnnn`, and `__pdf_backend_destination_aux:nnnn`.)

6.4.4 Structure

`__pdf_backend_compresslevel:n` Pass data to the backend: these are a one-shot.
`__pdf_backend_compress_objects:n`

```

3077 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
3078 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
3079 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
3080 {
3081   \bool_if:nF {#1}
3082   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
3083 }

```

(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`

```

3084 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
3085 {
3086   \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
3087   \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
3088 }
3089 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
3090 {
3091   \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
3092   \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
3093 }

```

(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: 3094 \cs_new:Npn \__pdf_backend_version_major: { 1 }
                               3095 \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: 3096 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
                    3097 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
                    3098 \cs_new_protected:Npn \__pdf_backend_emc:
                    3099 { \__kernel_backend_literal_page:n { EMC } }

(End definition for \__pdf_backend_bdc:nn and \__pdf_backend_emc:.)
3100 </dviPDFmx | xetex>

```

6.5 dvisvgm backend

```
3101 <*dvisvgm>
```

6.5.1 Annotations

```

\__pdf_backend_annotation:nmmn
3102 \cs_new_protected:Npn \__pdf_backend_annotation:nmmn #1#2#3#4 { }

(End definition for \__pdf_backend_annotation:nmmn.)

```

```

\__pdf_backend_annotation_last:
3103 \cs_new:Npn \__pdf_backend_annotation_last: { }

(End definition for \__pdf_backend_annotation_last:.)

```

```

\__pdf_backend_link_begin_goto:nmw
\__pdf_backend_link_begin_user:nmmn 3104 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nmw #1#2 { }
\__pdf_backend_link_begin:nmmnw 3105 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nmmn #1#2 { }
\__pdf_backend_link_end: 3106 \cs_new_protected:Npn \__pdf_backend_link_begin:nmmnw #1#2#3 { }
3107 \cs_new_protected:Npn \__pdf_backend_link_end: { }

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

```

```

\__pdf_backend_link_last:
3108 \cs_new:Npx \__pdf_backend_link_last: { }

(End definition for \__pdf_backend_link_last:.)

```

```

\__pdf_backend_link_margin:n A simple task: pass the data to the primitive.
3109 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1 { }

(End definition for \__pdf_backend_link_margin:n.)

```

```

\__pdf_backend_destination:nn
\__pdf_backend_destination:nmmn 3110 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2 { }
3111 \cs_new_protected:Npn \__pdf_backend_destination:nmmn #1#2#3#4 { }

(End definition for \__pdf_backend_destination:nn and \__pdf_backend_destination:nmmn.)

```

6.5.2 Catalogue entries

No-op.

```
\_pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn 3112 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2 { }
3113 \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.)

6.5.3 Objects

All no-ops here.

```
\_pdf_backend_object_new:n
\__pdf_backend_object_ref:n 3114 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1 { }
\_pdf_backend_object_write:mn 3115 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
\_pdf_backend_object_write:nx 3116 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3 { }
\_pdf_backend_object_now:nn 3117 \cs_new_protected:Npn \_pdf_backend_object_write:nxx #1#2#3 { }
\_pdf_backend_object_now:nx 3118 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
\_pdf_backend_object_last: 3119 \cs_new_protected:Npn \_pdf_backend_object_now:nx #1#2 { }
\_pdf_backend_pageobject_ref:n 3120 \cs_new:Npn \_pdf_backend_object_last: { }
3121 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1 { }
```

(End definition for _pdf_backend_object_new:n and others.)

6.5.4 Structure

These are all no-ops.

```
\_pdf_backend_compresslevel:n
\_pdf_backend_compress_objects:n 3122 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1 { }
3123 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1 { }
```

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

Data not available!

```
\_pdf_backend_version_major_gset:n
\_pdf_backend_version_minor_gset:n 3124 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
3125 \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.)

Data not available!

```
\_pdf_backend_version_major:
\_pdf_backend_version_minor: 3126 \cs_new:Npn \_pdf_backend_version_major: { -1 }
3127 \cs_new:Npn \_pdf_backend_version_minor: { -1 }
```

(End definition for _pdf_backend_version_major: and _pdf_backend_version_minor:.)

More no-ops.

```
\__pdf_backend_bdc:nn
\_pdf_backend_emc: 3128 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
3129 \cs_new_protected:Npn \_pdf_backend_emc: { }
```

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

```
3130 </dvisvgm>
```

6.6 PDF Page size (media box)

For setting the media box, the split between backends is somewhat different to other areas, thus we approach this separately. The code here assumes a recent L^AT_EX 2_ε: that is ensured at the level above.

```
3131 <*dvipdfmx | dvips>
```

```
\_pdf_backend_pagesize_set:nn  
\_pdf_backend_pagesize_set:n
```

This is done as a backend literal, so we deal with it using the shipout hook. The inclusion of `\tex_mag:D` here is essentially to work with (u)pL^AT_EX, where this is used to adjust apparent font size.

```
3132 \cs_new_protected:Npn \_pdf_backend_pagesize_set:nn #1#2  
3133 {  
3134   \__kernel_backend_first_shipout:n  
3135   {  
3136     \__kernel_backend_literal:e  
3137     {  
3138       <*dvipdfmx>  
3139         pdf:pagesize ~  
3140         width ~ \_pdf_backend_pagesize_set:n {#1} ~  
3141         height ~ \_pdf_backend_pagesize_set:n {#2}  
3142       </dvipdfmx>  
3143       <*dvips>  
3144         papersize =  
3145         \_pdf_backend_pagesize_set:n {#1} ,  
3146         \_pdf_backend_pagesize_set:n {#2}  
3147       </dvips>  
3148     }  
3149   }  
3150 }  
3151 \cs_new:Npn \_pdf_backend_pagesize_set:n #1  
3152 { \fp_to_dim:n { \tex_mag:D / 1000 * (#1) } }
```

(End definition for `_pdf_backend_pagesize_set:nn` and `_pdf_backend_pagesize_set:n`.)

```
3153 </dvipdfmx | dvips>  
3154 <*luatex | pdftex | xetex>
```

```
\_pdf_backend_pagesize_set:nn
```

Pass to the primitives.

```
3155 \cs_new_protected:Npn \_pdf_backend_pagesize_set:nn #1#2  
3156 {  
3157   \dim_set:Nn \tex_pagewidth:D {#1}  
3158   \dim_set:Nn \tex_pageheight:D {#2}  
3159 }
```

(End definition for `_pdf_backend_pagesize_set:nn`.)

```
3160 </luatex | pdftex | xetex>  
3161 <*dvisvgm>
```

```
\_pdf_backend_pagesize_set:nn
```

A no-op.

```
3162 \cs_new_protected:Npn \_pdf_backend_pagesize_set:nn #1#2 { }  
(End definition for \_pdf_backend_pagesize_set:nn.)  
3163 </dvisvgm>  
3164 </package>
```

7 I3backend-opacity Implementation

```
3165 (*package)
3166 (@@=opacity)
```

Although opacity is not color, it needs to be managed in a somewhat similar way: using a dedicated stack if possible. Depending on the backend, that may not be possible. There is also the need to cover fill/stroke setting as well as more general running opacity. It is easiest to describe the value used in terms of opacity, although commonly this is referred to as transparency.

```
3167 (*dvips)
```

`_opacity_backend_select:n` No stack so set values directly. The need to deal with Distiller and Ghostscript separately means we use a common auxiliary: the two systems require different PostScript for transparency. This is of course not quite as efficient as doing one test for setting all transparency, but it keeps things clearer here. Thanks to Alex Grahn for the detail on testing for GhostScript.

```
\\_opacity_backend_select_aux:n
\\_opacity_backend_fill:n
\\_opacity_backend_stroke:n
\\_opacity_backend:nnn
\\_opacity_backend:xnn
3168 \\cs_new_protected:Npn \\_opacity_backend_select:n #1
3169 {
3170   \\exp_args:Nx \\_opacity_backend_select_aux:n
3171   { \\fp_eval:n { min(max(0,#1),1) } }
3172 }
3173 \\cs_new_protected:Npn \\_opacity_backend_select_aux:n #1
3174 {
3175   \\_opacity_backend:nnn {#1} { fill } { ca }
3176   \\_opacity_backend:nnn {#1} { stroke } { CA }
3177 }
3178 \\cs_new_protected:Npn \\_opacity_backend_fill:n #1
3179 {
3180   \\_opacity_backend:xnn
3181   { \\fp_eval:n { min(max(0,#1),1) } }
3182   { fill }
3183   { ca }
3184 }
3185 \\cs_new_protected:Npn \\_opacity_backend_stroke:n #1
3186 {
3187   \\_opacity_backend:xnn
3188   { \\fp_eval:n { min(max(0,#1),1) } }
3189   { stroke }
3190   { CA }
3191 }
3192 \\cs_new_protected:Npn \\_opacity_backend:nnn #1#2#3
3193 {
3194   \\_kernel_backend_postscript:n
3195   {
3196     product ~ (Ghostscript) ~ search
3197     {
3198       pop ~ pop ~ pop ~
3199       #1 ~ .set #2 constantalpha
3200     }
3201     {
3202       pop ~
3203       mark ~
3204       /#3 ~ #1
```

```

3205         /SetTransparency ~
3206         pdfmark
3207     }
3208     ifelse
3209 }
3210 }
3211 \cs_generate_variant:Nn \_opacity_backend:nnn { x }

```

(End definition for _opacity_backend_select:n and others.)

```

3212 </dvips>
3213 <*dvipdfmx | luatex | pdftex | xetex>

```

\c__opacity_backend_stack_int Set up a stack, where that is applicable.

```

3214 \bool_lazy_and:nnT
3215 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3216 { \pdfmanagement_if_active_p:}
3217 {
3218 <*luatex | pdftex>
3219     \__kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3220     { page ~ direct } { /opacity 1 ~ gs }
3221 </luatex | pdftex>
3222     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3223     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3224 }

```

(End definition for \c__opacity_backend_stack_int.)

\l__opacity_backend_fill_tl We use tl here for speed: at the backend, this should be reasonable.

```

\l__opacity_backend_stroke_tl
3225 \tl_new:N \l__opacity_backend_fill_tl
3226 \tl_new:N \l__opacity_backend_stroke_tl

```

(End definition for \l__opacity_backend_fill_tl and \l__opacity_backend_stroke_tl.)

__opacity_backend_select:n Other than the need to evaluate the opacity as an fp, much the same as color.

```

\__opacity_backend_select_aux:n
\__opacity_backend_reset:
3227 \cs_new_protected:Npn \__opacity_backend_select:n #1
3228 {
3229     \exp_args:Nx \__opacity_backend_select_aux:n
3230     { \fp_eval:n { min(max(0,#1),1) } }
3231 }
3232 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3233 {
3234     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3235     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3236     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3237     { opacity #1 }
3238     { << /ca ~ #1 /CA ~ #1 >> }
3239 <*dvipdfmx | xetex>
3240     \__kernel_backend_literal_pdf:n
3241 </dvipdfmx | xetex>
3242 <*luatex | pdftex>
3243     \__kernel_color_backend_stack_push:nm \c__opacity_backend_stack_int
3244 </luatex | pdftex>
3245     { /opacity #1 ~ gs }
3246     \group_insert_after:N \__opacity_backend_reset:

```

```

3247 }
3248 \bool_lazy_and:nnF
3249 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3250 { \pdfmanagement_if_active_p:}
3251 {
3252   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1 { }
3253 }
3254 \cs_new_protected:Npn \__opacity_backend_reset:
3255 {
3256   <*dvipdfmx | xetex>
3257   \__kernel_backend_literal_pdf:n
3258   { /opacity1 ~ gs }
3259   </dvipdfmx | xetex>
3260   <*luatex | pdftex>
3261   \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int
3262   </luatex | pdftex>
3263 }

```

(End definition for __opacity_backend_select:n, __opacity_backend_select_aux:n, and __opacity_backend_reset:.)

__opacity_backend_fill:n For separate fill and stroke, we need to work out if we need to do more work or if we can
 __opacity_backend_stroke:n stick to a single setting.

```

\__opacity_backend_fillstroke:mn 3264 \cs_new_protected:Npn \__opacity_backend_fill:n #1
\__opacity_backend_fillstroke:xx 3265 {
3266   \__opacity_backend_fill_stroke:xx
3267   { \fp_eval:n { min(max(0,#1),1) } }
3268   \l__opacity_backend_stroke_tl
3269 }
3270 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3271 {
3272   \__opacity_backend_fill_stroke:xx
3273   \l__opacity_backend_fill_tl
3274   { \fp_eval:n { min(max(0,#1),1) } }
3275 }
3276 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3277 {
3278   \str_if_eq:nnTF {#1} {#2}
3279   { \__opacity_backend_select_aux:n {#1} }
3280   {
3281     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3282     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3283     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3284     { opacity.fill #1 }
3285     { << /ca ~ #1 >> }
3286     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3287     { opacity.stroke #1 }
3288     { << /CA ~ #2 >> }
3289   }
3290   <*dvipdfmx | xetex>
3291   \__kernel_backend_literal_pdf:n
3292   </dvipdfmx | xetex>
3293   <*luatex | pdftex>
3294   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3295   </luatex | pdftex>

```

```

3295         { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3296         \group_insert_after:N \_opacity_backend_reset:
3297     }
3298 }
3299 \cs_generate_variant:Nn \_opacity_backend_fill_stroke:nn { xx }

(End definition for \_opacity_backend_fill:n, \_opacity_backend_stroke:n, and \_opacity_backend_fillstroke:nn.)

3300 </dviptfm | luatex | pdftex | xetex>
3301 <*dvisvgm>

```

`_opacity_backend_select:n` Once again, we use a scope here. There is a general opacity function for SVG, but that is of course not set up using the stack.

```

\_opacity_backend_fill:n
\_opacity_backend_stroke:n
\_opacity_backend:nn
3302 \cs_new_protected:Npn \_opacity_backend_select:n #1
3303   { \_opacity_backend:nn {#1} { } }
3304 \cs_new_protected:Npn \_opacity_backend_fill:n #1
3305   { \_opacity_backend:nn {#1} { fill- } }
3306 \cs_new_protected:Npn \_opacity_backend_stroke:n #1
3307   { \_opacity_backend:nn {#1} { stroke- } }
3308 \cs_new_protected:Npn \_opacity_backend:nn #1#2
3309   { \_kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

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

3310 </dvisvgm>
3311 </package>

```

8 I3backend-header Implementation

```

3312 <*dvips & header>

```

`color.sc` Empty definition for color at the top level.

```

3313 /color.sc { } def

```

(End definition for `color.sc`. This function is documented on page ??.)

`TeXcolorseparation separation` Support for separation/spot colors: this strange naming is so things work with the color stack.

```

3314 TeXDict begin
3315 /TeXcolorseparation { setcolor } def
3316 end

```

(End definition for `TeXcolorseparation separation`. These functions are documented on page ??.)

`pdf.globaldict` A small global dictionary for backend use.

```

3317 true setglobal
3318 /pdf.globaldict 4 dict def
3319 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 to allow for Resolution. The total height of a rectangle (an array) needs a little maths, in contrast to simply extracting a value.

```

pdf.dvi.pt
pdf.pt.dvi
pdf.rect.ht
3320 /pdf.cvs { 65534 string cvs } def
3321 /pdf.dvi.pt { 72.27 mul Resolution div } def
3322 /pdf.pt.dvi { 72.27 div Resolution mul } def
3323 /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 3324 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad 3325 /pdf.linkdp.pad { 0 } def
3326 /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 separate links for generic annotations to support adding a margin and setting a minimal size.

```

pdf.save.ll
pdf.save.ur
pdf.save.linkll 3327 /pdf.rect
pdf.save.linkur 3328 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.llx 3329 /pdf.save.ll
pdf.lly 3330 {
pdf.urx 3331 currentpoint
pdf.ury 3332 /pdf.lly exch def
3333 /pdf.llx exch def
3334 }
3335 def
3336 /pdf.save.ur
3337 {
3338 currentpoint
3339 /pdf.ury exch def
3340 /pdf.urx exch def
3341 }
3342 def
3343 /pdf.save.linkll
3344 {
3345 currentpoint
3346 pdf.linkmargin add
3347 pdf.linkdp.pad add
3348 /pdf.lly exch def
3349 pdf.linkmargin sub
3350 /pdf.llx exch def
3351 }
3352 def
3353 /pdf.save.linkur
3354 {
3355 currentpoint
3356 pdf.linkmargin sub
3357 pdf.linkht.pad sub
3358 /pdf.ury exch def
3359 pdf.linkmargin add

```

```

3360 /pdf.urx exch def
3361 }
3362 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 function as it comes up a lot, and as this makes it easier to adjust if we need additional effects. We also need a more complex approach to convert a co-ordinate pair correctly when defining a rectangle: this can otherwise be out when using a landscape page. (Thanks to Alexander Grahn for the approach here.)

```

pdf.dev.x 3363 /pdf.dest.anchor
pdf.dev.y 3364 {
pdf.tmpa 3365 currentpoint exch
pdf.tmpb 3366 pdf.dvi.pt 72 add
pdf.tmpc 3367 /pdf.dest.x exch def
pdf.tmpd 3368 pdf.dvi.pt
3369 vsize 72 sub exch sub
3370 /pdf.dest.y exch def
3371 }
3372 def
3373 /pdf.dest.point
3374 { pdf.dest.x pdf.dest.y } def
3375 /pdf.dest2device
3376 {
3377 /pdf.dest.y exch def
3378 /pdf.dest.x exch def
3379 matrix currentmatrix
3380 matrix defaultmatrix
3381 matrix invertmatrix
3382 matrix concatmatrix
3383 cvx exec
3384 /pdf.dev.y exch def
3385 /pdf.dev.x exch def
3386 /pdf.tmpd exch def
3387 /pdf.tmpc exch def
3388 /pdf.tmpb exch def
3389 /pdf.tmpa exch def
3390 pdf.dest.x pdf.tmpa mul
3391 pdf.dest.y pdf.tmpc mul add
3392 pdf.dev.x add
3393 pdf.dest.x pdf.tmpb mul
3394 pdf.dest.y pdf.tmpd mul add
3395 pdf.dev.y add
3396 }
3397 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 can be done by hooking into *a* and *x* operations: those names have to be retained. The boundary is stored at the end of the operation. Special effort is needed at the start and end of pages (or rather galleys), such that everything works properly.

```

3398 /pdf.bordertracking false def

```

```

pdf.bordertracking.begin
pdf.bordertracking.end
pdf.leftboundary
pdf.rightboundary
pdf.brokenlink.rect
pdf.brokenlink.skip
pdf.brokenlink.dict
pdf.bordertracking.endpage
pdf.bordertracking.continue
pdf.originx
pdf.originy

```

```

3399 /pdf.bordertracking.begin
3400 {
3401   SDict /pdf.bordertracking true put
3402   SDict /pdf.leftboundary undef
3403   SDict /pdf.rightboundary undef
3404   /a where
3405     {
3406       /a
3407       {
3408         currentpoint pop
3409         SDict /pdf.rightboundary known dup
3410         {
3411           SDict /pdf.rightboundary get 2 index lt
3412           { not }
3413           if
3414         }
3415         if
3416           { pop }
3417           { SDict exch /pdf.rightboundary exch put }
3418         ifelse
3419         moveto
3420         currentpoint pop
3421         SDict /pdf.leftboundary known dup
3422         {
3423           SDict /pdf.leftboundary get 2 index gt
3424           { not }
3425           if
3426         }
3427         if
3428           { pop }
3429           { SDict exch /pdf.leftboundary exch put }
3430         ifelse
3431       }
3432     } put
3433   }
3434   if
3435 }
3436 def
3437 /pdf.bordertracking.end
3438 {
3439   /a where { /a { moveto } put } if
3440   /x where { /x { 0 exch rmoveto } put } if
3441   SDict /pdf.leftboundary known
3442     { pdf.outerbox 0 pdf.leftboundary put }
3443   if
3444   SDict /pdf.rightboundary known
3445     { pdf.outerbox 2 pdf.rightboundary put }
3446   if
3447   SDict /pdf.bordertracking false put
3448 }
3449 def
3450 /pdf.bordertracking.endpage
3451 {
3452 pdf.bordertracking

```

```

3453 {
3454   pdf.bordertracking.end
3455   true setglobal
3456   pdf.globaldict
3457   /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3458   pdf.globaldict
3459   /pdf.brokenlink.skip pdf.baselineskip put
3460   pdf.globaldict
3461   /pdf.brokenlink.dict
3462   pdf.link.dict pdf.cvs put
3463   false setglobal
3464   mark pdf.link.dict cvx exec /Rect
3465   [
3466     pdf.llx
3467     pdf.lly
3468     pdf.outerbox 2 get pdf.linkmargin add
3469     currentpoint exch pop
3470     pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3471   ]
3472   /ANN pdf.pdfmark
3473 }
3474 if
3475 }
3476 def
3477 /pdf.bordertracking.continue
3478 {
3479   /pdf.link.dict pdf.globaldict
3480   /pdf.brokenlink.dict get def
3481   /pdf.outerbox pdf.globaldict
3482   /pdf.brokenlink.rect get def
3483   /pdf.baselineskip pdf.globaldict
3484   /pdf.brokenlink.skip get def
3485   pdf.globaldict dup dup
3486   /pdf.brokenlink.dict undef
3487   /pdf.brokenlink.skip undef
3488   /pdf.brokenlink.rect undef
3489   currentpoint
3490   /pdf.originy exch def
3491   /pdf.originx exch def
3492   /a where
3493   {
3494     /a
3495     {
3496       moveto
3497       SDict
3498       begin
3499         currentpoint pdf.originy ne exch
3500         pdf.originx ne or
3501         {
3502           pdf.save.linkll
3503           /pdf.lly
3504             pdf.lly pdf.outerbox 1 get sub def
3505           pdf.bordertracking.begin
3506         }

```

```

3507         if
3508         end
3509     }
3510     put
3511 }
3512 if
3513 /x where
3514 {
3515     /x
3516     {
3517         0 exch rmoveto
3518         SDict
3519         begin
3520         currentpoint
3521         pdf.originy ne exch pdf.originx ne or
3522         {
3523             pdf.save.linkll
3524             /pdf.lly
3525             pdf.lly pdf.outerbox 1 get sub def
3526             pdf.bordertracking.begin
3527         }
3528         if
3529         end
3530     }
3531     put
3532 }
3533 if
3534 }
3535 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 Rect entry in the dictionary, looping over key–value pairs. The first line is handled first, adjusting pdf.breaklink.write the rectangle to stay inside the text area. The second phase is a loop over the height of pdf.count the bulk of the link area, done on the basis of a number of baselines. Finally, the end of pdf.currentrect the link area is tidied up, again from the boundary of the text area.

```

3536 /pdf.breaklink
3537 {
3538     pop
3539     counttomark 2 mod 0 eq
3540     {
3541         counttomark /pdf.count exch def
3542         {
3543             pdf.count 0 eq { exit } if
3544             counttomark 2 roll
3545             1 index /Rect eq
3546             {
3547                 dup 4 array copy
3548                 dup dup
3549                 1 get
3550                 pdf.outerbox pdf.rect.ht
3551                 pdf.linkmargin 2 mul add sub
3552                 3 exch put

```

```

3553     dup
3554         pdf.outerbox 2 get
3555         pdf.linkmargin add
3556         2 exch put
3557     dup dup
3558         3 get
3559         pdf.outerbox pdf.rect.ht
3560         pdf.linkmargin 2 mul add add
3561         1 exch put
3562     /pdf.currentrect exch def
3563     pdf.breaklink.write
3564     {
3565         pdf.currentrect
3566         dup
3567             pdf.outerbox 0 get
3568             pdf.linkmargin sub
3569             0 exch put
3570         dup
3571             pdf.outerbox 2 get
3572             pdf.linkmargin add
3573             2 exch put
3574         dup dup
3575             1 get
3576             pdf.baselineskip add
3577             1 exch put
3578         dup dup
3579             3 get
3580             pdf.baselineskip add
3581             3 exch put
3582         /pdf.currentrect exch def
3583         pdf.breaklink.write
3584     }
3585     1 index 3 get
3586     pdf.linkmargin 2 mul add
3587     pdf.outerbox pdf.rect.ht add
3588     2 index 1 get sub
3589     pdf.baselineskip div round cvi 1 sub
3590     exch
3591     repeat
3592     pdf.currentrect
3593     dup
3594         pdf.outerbox 0 get
3595         pdf.linkmargin sub
3596         0 exch put
3597     dup dup
3598         1 get
3599         pdf.baselineskip add
3600         1 exch put
3601     dup dup
3602         3 get
3603         pdf.baselineskip add
3604         3 exch put
3605     dup 2 index 2 get 2 exch put
3606     /pdf.currentrect exch def

```

```

3607         pdf.breaklink.write
3608         SDict /pdf.pdfmark.good false put
3609         exit
3610     }
3611     { pdf.count 2 sub /pdf.count exch def }
3612     ifelse
3613 }
3614 loop
3615 }
3616 if
3617 /ANN
3618 }
3619 def
3620 /pdf.breaklink.write
3621 {
3622     counttomark 1 sub
3623     index /_objdef eq
3624     {
3625         counttomark -2 roll
3626         dup wcheck
3627         {
3628             readonly
3629             counttomark 2 roll
3630         }
3631         { pop pop }
3632     } ifelse
3633 }
3634 if
3635 counttomark 1 add copy
3636 pop pdf.currentrect
3637 /ANN pdfmark
3638 }
3639 def

```

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

pdf.pdfmark The business end of breaking links starts by hooking into pdfmarks. Unlike hypdvips, pdf.pdfmark.good we avoid altering any links we have not created by using a copy of the core pdfmarks pdf.outerbox function. Only mark types which are known are altered. At present, this is purely ANN pdf.baselineskip marks, which are measured relative to the size of the baseline skip. If they are more than pdf.pdfmark.dict one apparent line high, breaking is applied.

```

3640 /pdf.pdfmark
3641 {
3642     SDict /pdf.pdfmark.good true put
3643     dup /ANN eq
3644     {
3645         pdf.pdfmark.store
3646         pdf.pdfmark.dict
3647         begin
3648             Subtype /Link eq
3649             currentdict /Rect known and
3650             SDict /pdf.outerbox known and
3651             SDict /pdf.baselineskip known and
3652             {

```

```

3653         Rect 3 get
3654         pdf.linkmargin 2 mul add
3655         pdf.outerbox pdf.rect.ht add
3656         Rect 1 get sub
3657         pdf.baselineskip div round cvi 0 gt
3658         { pdf.breaklink }
3659         if
3660     }
3661     if
3662     end
3663     SDict /pdf.outerbox undef
3664     SDict /pdf.baselineskip undef
3665     currentdict /pdf.pdfmark.dict undef
3666 }
3667 if
3668 pdf.pdfmark.good
3669 { pdfmark }
3670 { cleartomark }
3671 ifelse
3672 }
3673 def
3674 /pdf.pdfmark.store
3675 {
3676     /pdf.pdfmark.dict 65534 dict def
3677     counttomark 1 add copy
3678     pop
3679     {
3680         dup mark eq
3681         {
3682             pop
3683             exit
3684         }
3685         {
3686             pdf.pdfmark.dict
3687             begin def end
3688         }
3689     } ifelse
3690 }
3691 loop
3692 }
3693 def

```

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

```
3694 </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**
- `\AtBeginDvi` 57
- B**
- bool commands:
- `\bool_gset_false:N`
..... 1177, 1196, 1219, 1241,
1257, 1358, 1597, 1633, 2370, 2416
 - `\bool_gset_true:N`
.. 1175, 1244, 1356, 1612, 2363, 2369
 - `\bool_if:NTF` 67,
610, 1187, 1191, 1207, 1210, 1214,
1225, 1232, 1236, 1248, 1252, 1369,
1374, 1379, 1571, 1616, 1755, 1799,
1938, 1980, 2358, 2373, 2378, 2383
 - `\bool_if:nTF` 2592, 2845, 3081
 - `\bool_lazy_and:nnTF`
..... 823, 2097, 3214, 3248
 - `\bool_lazy_or:nnTF` 1791, 1973
 - `\bool_new:N`
.. 1178, 1245, 1359, 1613, 2343, 2344
 - `\bool_set_false:N`
..... 1766, 1902, 2004, 2168
- box commands:
- `\box_dp:N`
. 208, 210, 258, 260, 315, 317, 364,
366, 368, 370, 2395, 2428, 2429, 2454
 - `\box_ht:N` 210, 260, 317, 368,
370, 1811, 2045, 2400, 2439, 2440, 2456
 - `\box_if_empty:N` 2489
 - `\box_move_down:nn` 2317, 2395
 - `\box_move_up:nn` 2189, 2319, 2400
 - `\box_new:N` 2214, 2307, 2308
 - `\box_set_dp:Nn` 1696
 - `\box_set_ht:Nn` 1695
 - `\box_set_wd:Nn` 272, 1694
 - `\box_use:N` 215, 233,
247, 263, 290, 304, 320, 336, 348,
399, 413, 432, 1309, 1504, 1697, 2348
 - `\box_wd:N` 209, 217,
259, 265, 316, 322, 365, 367, 1810, 2044
- box internal commands:
- `__box_backend_clip:N`
197, 197, 252, 252, 309, 309, 353, 353
 - `\l_box_backend_cos_fp` 267
 - `__box_backend_rotate:Nn`
219, 219, 267, 267, 324, 324, 403, 403
- `__box_backend_rotate_aux:Nn` 219,
220, 221, 267, 268, 269, 324, 325, 326
 - `__box_backend_scale:Nnn`
236, 236, 295, 295, 339, 339, 416, 416
 - `\l__box_backend_sin_fp` 267
- C**
- clist commands:
- `\clist_map_function:nN`
..... 1265, 1389, 1640
- color internal commands:
- `__color_backend:nnn`
..... 1059, 1066, 1081, 1089, 1095
 - `__color_backend_cmyk:w` 1060
 - `\g_color_backend_colorant_prop` .
..... 576, 595, 598, 618, 859
 - `__color_backend_devicen_`
colorants:n 577, 577, 779, 917
 - `__color_backend_devicen_`
colorants:w 577, 585, 592, 600
 - `__color_backend_devicen_`
init:nnn
..... 766, 766, 884, 884, 1116, 1116
 - `__color_backend_devicen_init:w` .
..... 884, 893, 922, 926
 - `__color_backend_fill:n`
..... 963, 963, 965,
966, 967, 989, 990, 992, 994, 995,
1014, 1023, 1024, 1026, 1028, 1029,
1040, 1049, 1050, 1052, 1054, 1055
 - `__color_backend_fill_cmyk:n` 963,
965, 989, 989, 1023, 1023, 1049, 1049
 - `__color_backend_fill_devicen:nn`
..... 973,
983, 1013, 1017, 1039, 1043, 1110, 1112
 - `__color_backend_fill_gray:n` 963,
966, 989, 991, 1023, 1025, 1049, 1051
 - `__color_backend_fill_reset:` 985,
985, 1019, 1019, 1045, 1045, 1114, 1114
 - `__color_backend_fill_rgb:n` 963,
967, 989, 993, 1023, 1027, 1049, 1053
 - `__color_backend_fill_separation:nn`
.. 973, 973, 983, 1013, 1013, 1017,
1039, 1039, 1043, 1110, 1110, 1112
 - `\l__color_backend_fill_tl`
..... 541, 551, 997, 1011
 - `__color_backend_iccbased_`
device:nnn 946, 946

_color_backend_iccbased_-	_color_backend_separation_-
init:nnn	init_aux:nnnnn 608, 614, 630
. 785, 785, 928, 928, 1116, 1117	_color_backend_separation_-
_color_backend_init_resource:n	init_CIELAB:mnn
. 820, 820, 849, 920, 944, 959 608, 720, 790, 837, 862
_color_backend_pickup:N	_color_backend_separation_-
. 440, 440, 457, 457	init_CIELAB:nnnnnn 791
_color_backend_pickup:w	_color_backend_separation_-
. 440, 449, 453, 457, 459, 462	init_count:n 608, 667, 670
_color_backend_reset:	_color_backend_separation_-
. 522, 537, 543, 555, 559,	init_count:w 608, 671, 672, 676
564, 985, 986, 1019, 1020, 1045, 1114	_color_backend_separation_-
_color_backend_rgb:w 1083	init_Device:Nn
_color_backend_select:n 608, 652, 654, 656, 657
. 522, 523, 525, 527,	\l_color_backend_stack_int
529, 530, 559, 559, 561, 562, 563, 605 483, 553, 556, 998, 1010
_color_backend_select:mnn	_color_backend_stroke:n
. 543, 544, 546, 548, 549, 816 963, 968, 970, 971,
_color_backend_select_cmyk:n	972, 989, 1002, 1004, 1006, 1007, 1016
. 522, 522, 543, 543, 559, 561	_color_backend_stroke_cmyk:n
_color_backend_select_devicen:mnn 963,
. 604, 606, 788, 789, 810, 818	970, 989, 1001, 1023, 1033, 1059, 1059
_color_backend_select_gray:n	_color_backend_stroke_cmyk:w
. 522, 524, 543, 545, 559, 562, 569 1059, 1061
_color_backend_select_iccbased:mnn	_color_backend_stroke_devicen:mnn
. 607, 607, 792, 792, 810, 819 973,
_color_backend_select_named:n	984, 1013, 1018, 1039, 1044, 1110, 1113
. 522, 526, 566, 566	_color_backend_stroke_gray:n
_color_backend_select_rgb:n 963,
. 522, 528, 543, 547, 559, 563	971, 989, 1003, 1023, 1035, 1059, 1072
_color_backend_select_separation:mnn	_color_backend_stroke_gray_-
. 604, 604, 606,	aux:n 1059, 1076, 1080
788, 788, 789, 810, 811, 815, 818, 819	_color_backend_stroke_reset:
_color_backend_separation_- 985,
init:n 608, 689, 702	986, 1019, 1020, 1045, 1046, 1114, 1115
_color_backend_separation_-	_color_backend_stroke_rgb:n
init:nn 837, 847, 851 963,
_color_backend_separation_-	972, 989, 1005, 1023, 1037, 1059, 1082
init:nnn 608, 643, 664	_color_backend_stroke_rgb:w
_color_backend_separation_- 1059, 1084
init:nnnn 608, 666, 678	_color_backend_stroke_separation:nn
_color_backend_separation_- 973, 978, 984, 1013, 1015, 1018,
init:nnnnn 608,	1039, 1041, 1044, 1110, 1111, 1113
608, 629, 722, 790, 790, 837, 837, 877	\l_color_backend_stroke_tl
_color_backend_separation_- 541, 552, 999, 1009
init:nw 608, 693, 704, 718	\g_color_model_int 615, 624, 772,
_color_backend_separation_-	800, 849, 855, 856, 910, 911, 920, 944
init:w 608, 680, 695, 700	\c_color_model_range_CIELAB_tl
_color_backend_separation_- 727, 762, 873, 880
init_/DeviceCMYK:nnn 608	color.sc 522, 3313
_color_backend_separation_-	cs commands:
init_/DeviceGray:nnn 608	\cs_generate_variant:Nn
_color_backend_separation_-	49, 63, 66, 99, 138, 143, 154, 185,
init_/DeviceRGB:nnn 608	191, 629, 1124, 1319, 1513, 1952,

2015, 2035, 2218, 2239, 2302, 2796,	1346, 1355, 1357, 1360, 1362, 1364,
2809, 2919, 2940, 2970, 3211, 3299	1366, 1371, 1376, 1381, 1383, 1396,
\cs_gset:Npx .. 2604, 2608, 3086, 3091	1401, 1403, 1405, 1407, 1409, 1411,
\cs_gset_protected:Npn 3252	1413, 1415, 1426, 1451, 1463, 1475,
\cs_if_exist:NTF	1487, 1494, 1516, 1522, 1527, 1532,
..... 27, 50, 1707, 2485, 2870, 2896	1543, 1553, 1563, 1565, 1567, 1569,
\cs_if_exist_p:N 824, 3215, 3249	1600, 1602, 1607, 1609, 1611, 1614,
\cs_if_exist_use:NTF 38, 642	1635, 1646, 1659, 1661, 1663, 1665,
\cs_new:Npn	1667, 1669, 1671, 1673, 1675, 1683,
592,	1705, 1724, 1747, 1763, 1775, 1780,
651, 653, 655, 657, 664, 670, 672,	1788, 1814, 1827, 1845, 1855, 1871,
678, 695, 702, 704, 922, 1270, 1394,	1890, 1899, 1907, 1919, 1925, 1928,
1644, 1813, 2048, 2206, 2231, 2303,	1943, 1953, 1992, 2001, 2007, 2013,
2305, 2338, 2510, 2610, 2611, 2763,	2016, 2023, 2036, 2041, 2049, 2056,
2778, 2797, 2798, 2901, 2933, 2971,	2073, 2107, 2138, 2139, 2141, 2143,
2973, 2989, 3013, 3094, 3095, 3103,	2145, 2151, 2157, 2165, 2171, 2174,
3115, 3120, 3121, 3126, 3127, 3151	2176, 2187, 2216, 2219, 2221, 2224,
\cs_new:Npx	2233, 2240, 2257, 2262, 2267, 2272,
577, 2631, 2666, 2810, 2821, 2888, 3108	2282, 2287, 2295, 2310, 2315, 2347,
\cs_new_eq:NN 46, 57, 59, 561,	2349, 2354, 2356, 2361, 2376, 2381,
562, 563, 606, 789, 818, 819, 965,	2418, 2447, 2466, 2475, 2512, 2519,
966, 967, 970, 971, 972, 983, 984,	2545, 2550, 2578, 2590, 2602, 2606,
985, 986, 1017, 1018, 1019, 1020,	2612, 2614, 2618, 2642, 2644, 2646,
1043, 1044, 1045, 1112, 1113, 1114,	2657, 2677, 2687, 2710, 2724, 2734,
1123, 1318, 1324, 1325, 1512, 1514,	2745, 2765, 2799, 2832, 2843, 2849,
1515, 1521, 1721, 1722, 1735, 1737,	2877, 2911, 2913, 2920, 2922, 2926,
1761, 1762, 1819, 1820, 1821, 1844,	2935, 2941, 2946, 2951, 2953, 2955,
1869, 1886, 1887, 1896, 1897, 1898,	2963, 2976, 2992, 2994, 3011, 3015,
1918, 1921, 1922, 1923, 1988, 1998,	3017, 3039, 3044, 3077, 3079, 3084,
1999, 2000, 2154, 2155, 2163, 2164,	3089, 3096, 3098, 3102, 3104, 3105,
2173, 2203, 2204, 2205, 2209, 2348	3106, 3107, 3109, 3110, 3111, 3112,
\cs_new_protected:Npn	3113, 3114, 3116, 3117, 3118, 3119,
..... 47, 54, 61, 64, 72,	3122, 3123, 3124, 3125, 3128, 3129,
78, 83, 85, 89, 100, 110, 119, 128,	3132, 3155, 3162, 3168, 3173, 3178,
141, 144, 146, 148, 152, 157, 166,	3185, 3192, 3227, 3232, 3254, 3264,
176, 186, 197, 219, 221, 236, 252,	3270, 3276, 3302, 3304, 3306, 3308
267, 269, 295, 309, 324, 326, 339,	\cs_new_protected:Npx
353, 403, 416, 440, 453, 457, 462, 608, 1095, 2860, 2917, 2996
486, 500, 510, 522, 524, 526, 528,	\cs_set_eq:NN
530, 537, 543, 545, 547, 549, 555, 2506, 2507
559, 564, 566, 604, 607, 630, 720,	\cs_set_protected:Npn
766, 785, 788, 790, 791, 792, 811, 2111
815, 820, 837, 851, 862, 884, 928,	
946, 963, 968, 973, 978, 989, 991,	
993, 995, 1001, 1003, 1005, 1007,	
1013, 1015, 1023, 1025, 1027, 1029,	
1033, 1035, 1037, 1039, 1041, 1046,	
1049, 1051, 1053, 1055, 1059, 1061,	
1072, 1080, 1082, 1084, 1110, 1111,	
1115, 1116, 1117, 1125, 1130, 1135,	
1137, 1139, 1147, 1155, 1164, 1174,	
1176, 1179, 1181, 1198, 1203, 1221,	
1243, 1246, 1259, 1272, 1277, 1279,	
1281, 1283, 1285, 1287, 1289, 1291,	
1296, 1320, 1322, 1326, 1331, 1336,	

D

dim commands:

\dim_compare:nNnTF	2087, 2092
\dim_compare_p:nNn	2098, 2099
\dim_eval:n	2313, 2548,
2626, 2627, 2628, 2685, 2720, 2721,	2722, 2983, 2984, 2985, 3016, 3042
\dim_max:nn	2426, 2437
\dim_set:Nn	1810, 1811,
2044, 2045, 2083, 2084, 3157, 3158	
\dim_set_eq:NN	2149
\dim_to_decimal:n ..	364, 365, 366,
367, 368, 370, 1525, 1530, 1536,	

1537, 1538, 1539, 1548, 1549, 1550,
 1641, 1660, 2196, 2197, 2424, 2435,
 2453, 2454, 2455, 2456, 2460, 2516
 \dim_to_decimal_in_bp:n
 208, 209, 210, 258, 259, 260,
 315, 316, 317, 1143, 1144, 1151,
 1152, 1159, 1160, 1168, 1169, 1170,
 1267, 1271, 1275, 1329, 1334, 1340,
 1341, 1342, 1350, 1351, 1391, 1395,
 1399, 1645, 1729, 1730, 1731, 1732,
 1912, 1913, 1914, 1915, 1967, 1968,
 1969, 1970, 2181, 2182, 2183, 2184
 \dim_zero:N 2081, 2082
 \c_max_dim
 .. 2083, 2084, 2087, 2092, 2098, 2099
 draw internal commands:
 __draw_align_currentpoint... .. 35
 __draw_backend_add_to_path:n ...
 1522,
 1524, 1529, 1534, 1545, 1553, 1568
 __draw_backend_begin:
 .. 1125, 1125, 1320, 1320, 1516, 1516
 __draw_backend_box_use:Nnnnn ...
 31, 1296, 1296, 1494, 1494, 1683, 1683
 __draw_backend_cap_but:
 .. 1259, 1279, 1383, 1403, 1635, 1663
 __draw_backend_cap_rectangle: ..
 .. 1259, 1283, 1383, 1407, 1635, 1667
 __draw_backend_cap_round:
 .. 1259, 1281, 1383, 1405, 1635, 1665
 __draw_backend_clip:
 .. 1179, 1243, 1360, 1376, 1567, 1611
 __draw_backend_closepath:
 1179, 1179,
 1200, 1360, 1360, 1567, 1567, 1604
 __draw_backend_closestroke: ...
 .. 1179, 1198, 1360, 1364, 1567, 1602
 __draw_backend_cm:nnnn
 ... 1291, 1291, 1304, 1305, 1306,
 1415, 1415, 1498, 1675, 1675, 1686
 __draw_backend_cm_aux:nnnn
 1415, 1422, 1426
 __draw_backend_cm_decompose:nnnnN
 1421, 1450, 1451
 __draw_backend_cm_decompose_
 auxi:nnnnN 1450, 1455, 1463
 __draw_backend_cm_decompose_
 auxii:nnnnN 1450, 1467, 1475
 __draw_backend_cm_decompose_
 auxiii:nnnnN 1450, 1479, 1487
 __draw_backend_curveto:nnnnnn ..
 .. 1139, 1164, 1326, 1336, 1522, 1543
 __draw_backend_dash:n
 1259, 1265, 1270,
 1383, 1389, 1394, 1635, 1640, 1644
 __draw_backend_dash_aux:nn
 1635, 1639, 1646
 __draw_backend_dash_pattern:nn ..
 .. 1259, 1259, 1383, 1383, 1635, 1635
 __draw_backend_discardpath: ...
 .. 1179, 1246, 1360, 1381, 1567, 1614
 __draw_backend_end:
 .. 1125, 1130, 1320, 1322, 1516, 1521
 __draw_backend_evenodd_rule: ...
 .. 1174, 1174, 1355, 1355, 1563, 1563
 __draw_backend_fill:
 .. 1179, 1203, 1360, 1366, 1567, 1607
 __draw_backend_fillstroke:
 .. 1179, 1221, 1360, 1371, 1567, 1609
 __draw_backend_join_bevel:
 .. 1259, 1289, 1383, 1413, 1635, 1673
 __draw_backend_join_miter:
 .. 1259, 1285, 1383, 1409, 1635, 1669
 __draw_backend_join_round:
 .. 1259, 1287, 1383, 1411, 1635, 1671
 __draw_backend_lineto:nn
 .. 1139, 1147, 1326, 1331, 1522, 1527
 __draw_backend_linewidth:n ...
 .. 1259, 1272, 1383, 1396, 1635, 1659
 __draw_backend_literal:n
 1123, 1123, 1124, 1128,
 1132, 1136, 1138, 1141, 1149, 1157,
 1166, 1180, 1183, 1184, 1185, 1186,
 1189, 1195, 1205, 1212, 1218, 1223,
 1228, 1229, 1230, 1231, 1234, 1240,
 1250, 1256, 1261, 1274, 1278, 1280,
 1282, 1284, 1286, 1288, 1290, 1293,
 1298, 1299, 1300, 1301, 1302, 1303,
 1307, 1308, 1310, 1311, 1312, 1313,
 1314, 1318, 1318, 1319, 1328, 1333,
 1338, 1348, 1361, 1363, 1365, 1368,
 1373, 1378, 1382, 1385, 1398, 1402,
 1404, 1406, 1408, 1410, 1412, 1414,
 1512, 1512, 1513, 1574, 1593, 1619
 __draw_backend_miterlimit:n ...
 .. 1259, 1277, 1383, 1401, 1635, 1661
 __draw_backend_moveto:nn
 .. 1139, 1139, 1326, 1326, 1522, 1522
 __draw_backend_nonzero_rule: ...
 .. 1174, 1176, 1355, 1357, 1563, 1565
 __draw_backend_path:n
 1567, 1569, 1601, 1608, 1610
 \g__draw_backend_path_int 1582, 1599
 \g__draw_backend_path_tl
 1522, 1578, 1594, 1596, 1623
 __draw_backend_rectangle:nnnn ..
 .. 1139, 1155, 1326, 1346, 1522, 1532

<code>__draw_backend_scope_begin:</code>	1135 , 1135 , 1321 , 1324 , 1324 , 1514 , 1514	<code>\fp_set:Nn</code>	273 , 276
<code>__draw_backend_scope_end:</code>	1135 , 1137 , 1323 , 1324 , 1325 , 1514 , 1515	<code>\fp_to_dim:n</code>	3152
<code>__draw_backend_stroke:</code>	1179 , 1181 , 1201 , 1360 , 1362 , 1567 , 1600 , 1605	<code>\fp_use:N</code>	279 , 283 , 288
<code>\g__draw_draw_clip_bool</code>	1179 , 1567	<code>\fp_zero:N</code>	275
<code>\g__draw_draw_eor_bool</code>	1174 , 1191 , 1207 , 1214 , 1225 , 1236 , 1252 , 1355 , 1369 , 1374 , 1379	<code>\c_zero_fp</code>	227 , 274 , 280 , 332 , 1431 , 1444
<code>\g__draw_draw_path_int</code>	1567	G	
<code>\g__draw_path_tl</code>	1632	graphics commands:	
E		<code>\l_graphics_search_ext_seq</code>	1717 , 1740 , 1879 , 2067
<code>\errmessage</code>	38	graphics internal commands:	
<code>\evensidemargin</code>	2393	<code>__graphics_backend_dequote:w</code>	1747 , 1783 , 1813
exp commands:		<code>\l_graphics_backend_dir_str</code>	1822
<code>\exp_after:wN</code>	2054	<code>\l_graphics_backend_ext_str</code>	1822
<code>\exp_args:Ne</code>	666 , 1782 , 1835 , 1861 , 2547 , 3041	<code>__graphics_backend_get_pagecount:n</code>	1736 , 1737 , 1871 , 1871 , 1986 , 1988 , 2056 , 2056 , 2208 , 2209
<code>\exp_args:Nf</code>	1264 , 1388 , 2312	<code>__graphics_backend_getbb_auxi:n</code>	1747 , 1759 , 1773 , 1775
<code>\exp_args:NNf</code>	220 , 268 , 325	<code>__graphics_backend_getbb_-auxi:nN</code>	1992 , 1996 , 2005 , 2007
<code>\exp_args:Nnx</code>	2966	<code>__graphics_backend_getbb_-auxii:n</code>	1747 , 1778 , 1780
<code>\exp_args:NV</code>	442	<code>__graphics_backend_getbb_-auxii:nnN</code>	1992 , 2010 , 2013 , 2015
<code>\exp_args:Nx</code>	612 , 847 , 1833 , 1859 , 2269 , 2284 , 2389 , 3170 , 3229	<code>__graphics_backend_getbb_-auxiii:n</code>	1747 , 1782 , 1788
<code>\exp_last_unbraced:Nx</code>	449 , 459	<code>__graphics_backend_getbb_-auxiii:nNnn</code>	1992 , 2011 , 2014 , 2016
<code>\exp_not:N</code>	579 , 585 , 586 , 587 , 614 , 615 , 618 , 619 , 624 , 2633 , 2635 , 2638 , 2668 , 2670 , 2673 , 2812 , 2814 , 2817 , 2823 , 2825 , 2828 , 2865 , 2866 , 2872 , 2873 , 2892 , 2897 , 2998 , 3003	<code>__graphics_backend_getbb_-auxiv:nnNnn</code>	1992 , 2019 , 2023 , 2035
<code>\exp_not:n</code>	48 , 97 , 108 , 136 , 936 , 2260 , 2265 , 2541 , 2782 , 2783 , 2797 , 2798 , 2944 , 2949 , 2960 , 3021	<code>__graphics_backend_getbb_-auxv:nNnn</code>	1992 , 2020 , 2027 , 2036
<code>\ExplBackendFileDate</code>	1	<code>__graphics_backend_getbb_-auxvi:nNnn</code>	2039 , 2041
F		<code>__graphics_backend_getbb_bmp:n</code>	1884 , 1898 , 1992 , 2000
file commands:		<code>__graphics_backend_getbb_eps:n</code>	1719 , 1721 , 1822 , 1827 , 1844 , 1884 , 1886 , 2152 , 2154
<code>\file_compare_timestamp:nNnTF</code>	1847	<code>__graphics_backend_getbb_eps:nm</code>	1822
<code>\file_parse_full_name:nNNN</code>	1829 , 1857	<code>__graphics_backend_getbb_eps:nn</code>	1833 , 1845
<code>\fmtversion</code>	52	<code>__graphics_backend_getbb_jpeg:n</code>	1747 , 1761 , 1884 , 1896 , 1992 , 1998 , 2157 , 2163
fp commands:		<code>__graphics_backend_getbb_jpg:n</code>	1747 , 1747 , 1761 , 1762 , 1884 , 1890 , 1896 , 1897 , 1898 , 1992 , 1992 , 1998 , 1999 , 2000 , 2157 , 2157 , 2163 , 2164
<code>\fp_compare:nNnTF</code>	227 , 274 , 280 , 332 , 1431 , 1444 , 1489	<code>__graphics_backend_getbb_-pagebox:w</code>	1992 , 2031 , 2048 , 2054
<code>\fp_eval:n</code>	220 , 229 , 242 , 243 , 268 , 285 , 300 , 302 , 325 , 334 , 345 , 346 , 410 , 425 , 426 , 1067 , 1068 , 1069 , 1077 , 1090 , 1091 , 1092 , 1433 , 1438 , 1439 , 1446 , 1456 , 1457 , 1458 , 1459 , 1468 , 1469 , 1470 , 1471 , 1480 , 1481 , 1482 , 1483 , 2538 , 2707 , 3035 , 3171 , 3181 , 3188 , 3230 , 3267 , 3274 , 3309		
<code>\fp_new:N</code>	293 , 294		

`__graphics_backend_getbb_pdf:n` [1747](#), [1763](#), [1853](#),
[1884](#), [1899](#), [1992](#), [2001](#), [2165](#), [2165](#)
`__graphics_backend_getbb_png:n` [1747](#), [1762](#),
[1884](#), [1897](#), [1992](#), [1999](#), [2157](#), [2164](#)
`__graphics_backend_getbb_ps:n` [1719](#), [1722](#),
[1822](#), [1844](#), [1884](#), [1887](#), [2152](#), [2155](#)
`__graphics_backend_getbb_svg:n` [2073](#), [2073](#)
`__graphics_backend_getbb_svg_-auxi:nNn` [2073](#), [2089](#), [2094](#), [2107](#)
`__graphics_backend_getbb_svg_-auxii:w` [2073](#), [2111](#), [2133](#), [2138](#)
`__graphics_backend_getbb_svg_-auxiii:Nw` [2073](#), [2121](#), [2139](#)
`__graphics_backend_getbb_svg_-auxiv:Nw` [2073](#), [2124](#), [2141](#)
`__graphics_backend_getbb_svg_-auxv:Nw` [2073](#), [2125](#), [2143](#)
`__graphics_backend_getbb_svg_-auxvi:Nn` [2073](#), [2140](#), [2142](#), [2144](#), [2145](#)
`__graphics_backend_getbb_svg_-auxvii:w` [2073](#), [2147](#), [2151](#)
`__graphics_backend_include:nn` [2171](#), [2172](#), [2175](#), [2176](#)
`__graphics_backend_include_-auxi:nn` [1907](#), [1920](#), [1926](#), [1928](#)
`__graphics_backend_include_-auxii:nnn` [1907](#), [1930](#), [1943](#), [1952](#)
`__graphics_backend_include_-auxiii:nnn` [1907](#), [1950](#), [1953](#)
`__graphics_backend_include_-bmp:n` [1907](#), [1923](#)
`__graphics_backend_include_-dequote:w` [2187](#), [2198](#), [2206](#)
`__graphics_backend_include_-eps:n` [1724](#),
[1724](#), [1735](#), [1822](#), [1855](#), [1869](#),
[1907](#), [1907](#), [1918](#), [2171](#), [2171](#), [2173](#)
`__graphics_backend_include_-jpeg:n` . [1814](#), [1819](#), [1921](#), [2187](#), [2204](#)
`__graphics_backend_include_-jpg:n` [1814](#),
[1814](#), [1819](#), [1820](#), [1821](#), [1907](#),
[1919](#), [1921](#), [1922](#), [1923](#), [2187](#), [2205](#)
`__graphics_backend_include_-jpsseg:n` [1907](#)
`__graphics_backend_include_-pdf:n` [1814](#), [1820](#), [1859](#),
[1907](#), [1925](#), [2049](#), [2049](#), [2171](#), [2174](#)
`__graphics_backend_include_-png:n`
. [1814](#), [1821](#), [1907](#), [1922](#), [2187](#), [2203](#)
`__graphics_backend_include_ps:n` [1724](#), [1735](#),
[1822](#), [1869](#), [1907](#), [1918](#), [2171](#), [2173](#)
`__graphics_backend_include_-svg:n` [2187](#), [2187](#), [2203](#), [2204](#), [2205](#)
`__graphics_backend_loaded:n`
[1705](#), [1705](#), [1717](#), [1719](#), [1736](#), [1740](#),
[1879](#), [1884](#), [1987](#), [2067](#), [2152](#), [2208](#)
`\l__graphics_backend_name_str` . [1822](#)
`__graphics_bb_restore:nTF`
. [1777](#), [2038](#), [2075](#)
`__graphics_bb_save:n` [1786](#), [2046](#), [2102](#)
`\l__graphics_decodearray_str`
. [1753](#), [1754](#),
[1765](#), [1793](#), [1797](#), [1798](#), [1901](#), [1936](#),
[1937](#), [1975](#), [1978](#), [1979](#), [2003](#), [2167](#)
`__graphics_extract_bb:n`
. [1894](#), [1903](#), [2161](#), [2169](#)
`\l__graphics_final_name_str` [1852](#)
`__graphics_get_pagecount:n`
. [1737](#), [1988](#), [2209](#)
`\l__graphics_graphics_attr_tl`
. [1746](#), [1751](#),
[1758](#), [1767](#), [1777](#), [1784](#), [1786](#), [1817](#)
`\l__graphics_internal_box`
. [1808](#), [1810](#), [1811](#), [2043](#), [2044](#), [2045](#)
`\l__graphics_internal_dim` [2148](#), [2149](#)
`\l__graphics_internal_ior`
. [2077](#), [2078](#), [2085](#), [2104](#)
`\l__graphics_interpolate_bool`
. [1755](#), [1766](#), [1792](#), [1799](#),
[1902](#), [1938](#), [1974](#), [1980](#), [2004](#), [2168](#)
`\l__graphics_llx_dim`
. [1729](#), [1912](#), [1967](#), [2081](#), [2181](#)
`\l__graphics_lly_dim`
. [1730](#), [1913](#), [1968](#), [2082](#), [2182](#)
`\l__graphics_page_int`
. [1749](#), [1770](#), [1771](#), [1803](#),
[1804](#), [1892](#), [1934](#), [1935](#), [1961](#), [1962](#),
[1994](#), [2009](#), [2010](#), [2052](#), [2053](#), [2159](#)
`\l__graphics_pagebox_tl`
. [54](#), [1750](#), [1769](#),
[1805](#), [1806](#), [1893](#), [1932](#), [1933](#), [1963](#),
[1965](#), [1995](#), [2018](#), [2019](#), [2054](#), [2160](#)
`__graphics_read_bb:n`
. [1721](#), [1722](#), [1886](#), [1887](#), [2154](#), [2155](#)
`\g__graphics_track_int`
. [1906](#), [1955](#), [1956](#)
`\l__graphics_urx_dim`
. [1731](#), [1810](#), [1914](#), [1969](#), [2044](#),
[2083](#), [2087](#), [2090](#), [2098](#), [2183](#), [2196](#)
`\l__graphics_ury_dim`
. [1732](#), [1811](#), [1915](#), [1970](#), [2045](#), [2084](#),

	2092, 2095, 2099, 2184, 2189, 2197	ior commands:	
group commands:		\ior_close:N	2104
\group_begin:	163, 182	\ior_if_eof:NTF	2078
\group_end:	171	\ior_map_break:	2100
\group_insert_after:N	3246, 3296	\ior_open:Nn	2077
		\ior_str_map_inline:Nn	2085
H		K	
hbox commands:		kernel internal commands:	
\hbox:n	2191, 2318, 2321, 2396, 2402, 2555, 2562, 3049, 3060	__kernel_backend_align_begin:	72, 72, 200, 224, 239
\hbox_overlap_right:n	215, 247, 263, 304, 320, 348, 432, 1309, 1504	__kernel_backend_align_end:	72, 78, 214, 232, 246
\hbox_set:Nn	1808, 2043, 2388, 2420	__kernel_backend_first_shipout:n	50, 54, 57, 59, 69, 612, 3134
\hbox_set:Nw	2371	\g__kernel_backend_header_bool	67, 610
\hbox_set_end:	2386	__kernel_backend_literal:n	46, 46, 47, 48, 49, 62, 65, 70, 74, 81, 84, 86, 142, 145, 147, 149, 153, 329, 342, 532, 538, 560, 565, 632, 768, 812, 964, 969, 975, 980, 1031, 1057, 1127, 1133, 1428, 1435, 1441, 1501, 1506, 1726, 1909, 1947, 1957, 2178, 2193, 2918, 3016, 3078, 3082, 3087, 3092, 3136
\hbox_unpack:N	2507	__kernel_backend_literal_page:n	100, 100, 144, 144, 2912, 2914, 3097, 3099
hook commands:		__kernel_backend_literal_pdf:n	89, 89, 99, 141, 141, 143, 255, 312, 1318, 3240, 3257, 3290
\hook_gput_code:nnn	55, 1707, 1709	__kernel_backend_literal_- postscript:n	61, 61, 63, 75, 76, 80, 201, 202, 204, 205, 213, 225, 240, 1123, 2582, 2594
I		__kernel_backend_literal_svg:n	152, 152, 154, 159, 170, 178, 188, 356, 358, 375, 794, 1512, 1687, 1698
int commands:		__kernel_backend_matrix:n	128, 128, 138, 277, 298, 1418
\int_compare:nNnTF	1770, 1803, 1934, 1961, 2009, 2052, 2479, 2580, 2863, 2891	__kernel_backend_postscript:n	64, 64, 66, 534, 1034, 1036, 1038, 1042, 2217, 2274, 2289, 2318, 2324, 2364, 2396, 2403, 2407, 2421, 2449, 2493, 2500, 2506, 2514, 2521, 2555, 2562, 3194
\int_const:Nn	488, 1784, 1874, 1956, 2058, 2227, 2754, 2929	__kernel_backend_scope:n	157, 186, 191, 385, 390, 1097, 1519, 1564, 1566, 1586, 1626, 1648, 1660, 1662, 1664, 1666, 1668, 1670, 1672, 1674, 1677, 3309
\int_eval:n	508, 518, 662, 671, 684, 686, 690, 703, 2604, 2608, 2841, 2866, 2873, 2886, 3078, 3086, 3091	__kernel_backend_scope_begin:	83, 83, 110, 110, 146, 146, 157, 157,
\int_gincr:N	189, 355, 1573, 1618, 1955, 2226, 2297, 2328, 2405, 2928, 2965, 2978, 2998		
\int_gset:Nn	164, 183, 2468		
\int_gset_eq:NN	172, 2329, 2406, 2979		
\int_if_exist:NTF	1945		
\int_if_odd:nTF	2391		
\int_max:nn	2060		
\int_new:N	155, 156, 402, 483, 1599, 1906, 2223, 2309, 2340, 2342, 2924, 2975, 2991		
\int_set_eq:NN	160, 179, 2480		
\int_step_function:nnnN	688		
\int_use:N	357, 388, 615, 624, 772, 800, 849, 855, 856, 910, 911, 920, 944, 1576, 1582, 1589, 1621, 1629, 1771, 1804, 1817, 1875, 1935, 1948, 1960, 1962, 2053, 2061, 2232, 2299, 2304, 2332, 2339, 2410, 2511, 2764, 2774, 2934, 2967, 2972, 2982, 2990, 3003, 3014		
\int_value:w	2633, 2668, 2812, 2823, 2841		
\int_zero:N	1749, 1892, 1994, 2159		

199, 223, 238, 254, 271, 297, 311,
328, 341, 1324, 1496, 1514, 1518, 1685

_kernel_backend_scope_begin:n .
..... 157, 176, 185, 377, 405, 418

_kernel_backend_scope_end: . . .
..... 83, 85, 110, 119,
146, 148, 157, 166, 216, 234, 248,
264, 291, 305, 321, 337, 349, 400,
414, 433, 1325, 1508, 1515, 1521, 1699

\g_kernel_backend_scope_int . . .
155, 162, 164, 169, 173, 181, 183, 189

\l_kernel_backend_scope_int . . .
..... 155, 161, 174, 180

\g_kernel_clip_path_int
353, 1573, 1576, 1589, 1618, 1621, 1629

_kernel_color_backend_stack_
init:Nnn 486, 486, 3219

_kernel_color_backend_stack_
pop:n 500, 510, 556, 3261

_kernel_color_backend_stack_
push:nn
. 500, 500, 553, 998, 1010, 3243, 3293

_kernel_dependency_version_
check:Nn 1

_kernel_dependency_version_
check:nn 27, 29

_kernel_file_name_quote:n
..... 1835, 1861

_kernel_kern:n
..... 2323, 2325, 2554, 2558,
2561, 2565, 3048, 3056, 3059, 3075

M

\MessageBreak 40

mode commands:

\mode_if_horizontal:TF . . . 2470, 2477

\mode_if_math:TF 2368

msg commands:

\msg_error:nnn 570, 2079

\msg_new:nnn 572

\msg_warning:nnn 445, 475

O

\oddsidemargin 2392

opacity internal commands:

_opacity_backend:nn
..... 3302, 3303, 3305, 3307, 3308

_opacity_backend:nnn 3168,
3175, 3176, 3180, 3187, 3192, 3211

_opacity_backend_fill:n
.. 3168, 3178, 3264, 3264, 3302, 3304

_opacity_backend_fill_stroke:nn
..... 3266, 3272, 3276, 3299

\l_opacity_backend_fill_tl
..... 3225, 3234, 3273, 3281

_opacity_backend_fillstroke:nn
..... 3264

_opacity_backend_reset:
..... 3227, 3246, 3254, 3296

_opacity_backend_select:n
.. 3168, 3168, 3227, 3227, 3302, 3302

_opacity_backend_select_aux:n .
..... 3168, 3170,
3173, 3227, 3229, 3232, 3252, 3279

\c_opacity_backend_stack_int . . .
..... 3214, 3243, 3261, 3293

_opacity_backend_stroke:n
.. 3168, 3185, 3264, 3270, 3302, 3306

\l_opacity_backend_stroke_tl . . .
..... 3225, 3235, 3268, 3282

P

pdf commands:

\pdf_object_if_exist:nTF 864, 930, 948

\pdf_object_new:n
..... 855, 866, 910, 932, 950

\pdf_object_ref:n
..... 812, 879, 943, 958, 976, 981

\pdf_object_ref_last:
..... 832, 857, 860, 916

\pdf_object_unnamed_write:nn . . .
..... 839, 886, 942, 957

\pdf_object_write:nnn
..... 856, 867, 911, 933, 951

pdf internal commands:

_pdf_backend:n . 2917, 2917, 2919,
2921, 2923, 2943, 2948, 2957, 2980,
2999, 3012, 3019, 3051, 3052, 3062

_pdf_backend_annotation:nnnn . .
..... 2310, 2310,
2618, 2618, 2976, 2976, 3102, 3102

_pdf_backend_annotation_
aux:nnnn 2312, 2315

\g_pdf_backend_annotation_int . .
.. 2309, 2329, 2339, 2975, 2979, 2990

_pdf_backend_annotation_last: .
..... 2338, 2338,
2631, 2631, 2989, 2989, 3103, 3103

_pdf_backend_bdc:nn 2612, 2612,
2911, 2911, 3096, 3096, 3128, 3128

_pdf_backend_catalog_gput:nn . .
..... 2219, 2219,
2724, 2724, 2920, 2920, 3112, 3112

_pdf_backend_compress_objects:n
..... 2578, 2590,
2832, 2843, 3077, 3079, 3122, 3123

_pdf_backend_compresslevel:n ..	\g_pdf_backend_link_math_bool ..
..... 2578, 2578, 2343, 2369, 2370, 2373, 2383
2832, 2832, 3077, 3077, 3122, 3122	_pdf_backend_link_minima:
\l_pdf_backend_content_box 2307, 2349, 2387, 2418
2371, 2395, 2398, 2400, 2429, 2440	_pdf_backend_link_outerbox:n ..
_pdf_backend_destination:nn 2349, 2389, 2447
..... 2519, 2519,	\g_pdf_backend_link_sf_int
2687, 2687, 3017, 3017, 3110, 3110 2342, 2468, 2479, 2480
_pdf_backend_destination:nnnn .	_pdf_backend_link_sf_restore: .
..... 2519, 2545, 2349, 2372, 2415, 2475
2687, 2710, 3017, 3039, 3110, 3111	_pdf_backend_link_sf_save: ...
_pdf_backend_destination_- 2349, 2367, 2385, 2466
aux:nnnn .. 2519, 2547, 2550, 3017, 3041, 3044	\l_pdf_backend_model_box . 2308,
_pdf_backend_emc: .. 2612, 2614,	2388, 2420, 2428, 2439, 2454, 2456
2911, 2913, 3096, 3098, 3128, 3129	_pdf_backend_objcompresslevel:n
_pdf_backend_info_gput:nn 2832, 2846, 2847, 2849
..... 2219, 2221,	\g_pdf_backend_object_int
2724, 2734, 2920, 2922, 3112, 3113 2223, 2226,
_pdf_backend_link:nw	2229, 2297, 2299, 2304, 2328, 2329,
2349	2332, 2405, 2406, 2924, 2928, 2931,
_pdf_backend_link_aux:nw ... 2349	2965, 2967, 2972, 2978, 2979, 2982
_pdf_backend_link_begin:n	_pdf_backend_object_last:
..... 2992, 2993, 2995, 2996 2303, 2303,
_pdf_backend_link_begin:nnnw ..	2810, 2810, 2971, 2971, 3114, 3120
.. 2642, 2643, 2645, 2646, 3104, 3106	_pdf_backend_object_new:n 2224,
_pdf_backend_link_begin:nw ...	2224, 2745, 2745, 2926, 2926, 3114
..... 2351, 2355, 2356	_pdf_backend_object_new:nn . 3114
_pdf_backend_link_begin_aux:nw	_pdf_backend_object_now:nn ...
..... 2359, 2361	2295, 2295, 2302, 2799, 2799, 2809,
_pdf_backend_link_begin_-	2963, 2963, 2970, 3114, 3118, 3119
goto:nnw	\g_pdf_backend_object_prop
2349, 2349, 2744, 2924
2642, 2642, 2992, 2992, 3104, 3104	_pdf_backend_object_ref:n
_pdf_backend_link_begin_- 2224, 2231, 2236, 2745,
user:nnw	2763, 2926, 2933, 2938, 3114, 3115
2642, 2644, 2992, 2994, 3104, 3105	_pdf_backend_object_write:nn ..
\g_pdf_backend_link_bool 2765, 2776, 2778, 2807, 3114
..... 2344, 2358, 2363, 2378, 2416	_pdf_backend_object_write:nnn .
\g_pdf_backend_link_dict_tl ...	2233, 2233, 2239, 2765, 2765, 2796,
..... 2341, 2366, 2411	2935, 2935, 2940, 3114, 3116, 3117
_pdf_backend_link_end:	_pdf_backend_object_write_-
..... 2349, 2376,	array:nn ... 2233, 2257, 2935, 2941
2642, 2657, 2992, 3011, 3104, 3107	_pdf_backend_object_write_-
_pdf_backend_link_end_aux: ...	aux:nnn 2233, 2235, 2240, 2298
..... 2349, 2379, 2381	_pdf_backend_object_write_-
\g_pdf_backend_link_int	dict:nn 2233, 2262, 2935, 2946
..... 2340, 2406,	_pdf_backend_object_write_-
2410, 2511, 2991, 2998, 3003, 3014	fstream:nn . 2233, 2267, 2935, 2951
_pdf_backend_link_last:	_pdf_backend_object_write_-
..... 2510, 2510,	fstream:nnn
2666, 2666, 3013, 3013, 3108, 3108	2270, 2272
_pdf_backend_link_margin:n ...	_pdf_backend_object_write_-
..... 2512, 2512,	stream:nn .. 2233, 2282, 2935, 2953
2677, 2677, 3015, 3015, 3109, 3109	_pdf_backend_object_write_-
	stream:nnn
	2233, 2285, 2287

S

scan commands:	
<code>\scan_stop:</code>	113, 122, 518, 2148, 2151, 2660, 2685, 2708, 2722, 2841, 2858, 2866, 2873, 2886
scan internal commands:	
<code>\s__color_stop</code> 450, 453, 460, 463, 671, 672, 676, 680, 693, 696, 700, 704, 718, 893, 922, 926, 1060, 1062, 1083, 1085
<code>\s__graphics_stop</code> 1783, 1813, 2113, 2128, 2135, 2139, 2141, 2143, 2198, 2206
separation	3314
seq commands:	
<code>\seq_set_from_clist:Nn</code> 1718, 1742, 1881, 2069
skip commands:	
<code>\skip_horizontal:n</code>	217, 265, 322
str commands:	
<code>\c_hash_str</code>	388, 1582, 1589, 1629
<code>\c_percent_str</code>	1103, 1104, 1105
<code>\str_case:nn</code>	898, 2246, 2780
<code>\str_case:nnTF</code>	2526, 2696, 3024
<code>\str_convert_pdfname:n</code>	619, 639, 848
<code>\str_if_eq:nnTF</code> 465, 468, 471, 568, 798, 3278
<code>\str_new:N</code>	1824, 1825, 1826
<code>\str_tail:N</code>	1838, 1864
sys commands:	
<code>\sys_if_shell:TF</code>	1822
<code>\sys_shell_now:n</code>	1849
<h2 style="margin: 0;">T</h2>	
TeX and L ^A T _E X 2 _ε commands:	
<code>\@cclv</code>	2489, 2491, 2499
<code>\@ifl@t@r</code>	50, 52
<code>\@makecol@hook</code>	2483
<code>\current@color</code>	442, 446, 450, 460, 476
<code>\special</code>	2
tex commands:	
<code>\tex_afterassignment:D</code>	2147
<code>\tex_baselineskip:D</code>	2460
<code>\tex_endinput:D</code>	44
<code>\tex_global:D</code> 2834, 2851, 2865, 2872, 2879
<code>\tex_immediate:D</code> 1790, 2768, 2771, 2802, 2805
<code>\tex_luatexversion:D</code>	2863, 2891
<code>\tex_mag:D</code>	83, 3152
<code>\tex_pageheight:D</code>	3158
<code>\tex_pagewidth:D</code>	3157
<code>\tex_pdfannot:D</code>	2624
<code>\tex_pdfcatalog:D</code>	2730
<code>\tex_pdfcolorstack:D</code>	506, 516
<code>\tex_pdfcolorstackinit:D</code>	494
<code>\tex_pdfcompresslevel:D</code>	2839
<code>\tex_pdfdest:D</code>	2693, 2716
<code>\tex_pdfendlink:D</code>	2663
<code>\tex_pdfextension:D</code> 92, 103, 113, 122, 131, 503, 513, 2621, 2649, 2660, 2690, 2713, 2727, 2737, 2748, 2768, 2802
<code>\tex_pdffeedback:D</code> 491, 2635, 2670, 2757, 2814, 2825
<code>\tex_pdfinfo:D</code>	2740
<code>\tex_pdflastannot:D</code>	2638
<code>\tex_pdflastlink:D</code>	2673
<code>\tex_pdflastobj:D</code>	2760, 2817
<code>\tex_pdflastximage:D</code>	1785, 1809
<code>\tex_pdflastximagepages:D</code>	1875
<code>\tex_pdflinkmargin:D</code>	2683
<code>\tex_pdfliteral:D</code>	95, 106
<code>\tex_pdfmajorversion:D</code> 2870, 2872, 2896, 2897
<code>\tex_pdfminorversion:D</code>	2884, 2908
<code>\tex_pdfobj:D</code>	2751, 2771, 2805
<code>\tex_pdfobjcompresslevel:D</code>	2856
<code>\tex_pdfpageref:D</code>	2828
<code>\tex_pdfrefximage:D</code>	1809, 1816
<code>\tex_pdfrestore:D</code>	125
<code>\tex_pdfsave:D</code>	116
<code>\tex_pdfsetmatrix:D</code>	134
<code>\tex_pdfstartlink:D</code>	2652
<code>\tex_pdfvariable:D</code>	2680, 2836, 2853, 2865, 2881, 2892, 2905
<code>\tex_pdfximage:D</code>	1790, 1873
<code>\tex_spacefactor:D</code>	2471, 2480
<code>\tex_special:D</code>	46
<code>\tex_the:D</code>	1785, 2892, 2897, 2903
<code>\tex_vss:D</code>	2556, 2563, 3054, 3073
<code>\tex_XeTeXpdffile:D</code>	2005, 2051
<code>\tex_XeTeXpdfpagecount:D</code>	2061
<code>\tex_XeTeXpicfile:D</code>	1996
TeXcolorseparation	3314
<code>\textwidth</code>	2455
tl commands:	
<code>\c_space_tl</code> 279, 284, 287, 581, 586, 624, 727, 801, 1011, 1558, 1728, 1729, 1730, 1731, 1911, 1912, 1913, 1914, 1962, 1965, 1967, 1968, 1969, 1970, 2031, 2053, 2180, 2181, 2182, 2183, 2411, 2640, 2675, 2819, 2830, 2982, 3004
<code>\tl_clear:N</code>	1750, 1758, 1765, 1893, 1901, 1995, 2003, 2160, 2167
<code>\tl_gclear:N</code>	1596, 1632
<code>\tl_gset:Nn</code>	1555, 2366

<code>\tl_if_blank:nTF</code>	496, 579, 675, 692, 699, 717, 843, 925, 2030, 2116	token commands:	
<code>\tl_if_empty:N</code>	1558, 1753, 1797, 1805, 1932, 1936, 1963, 1978, 2018	<code>\c_math_toggle_token</code>	2374, 2384
<code>\tl_if_empty:nTF</code>	937, 1652		U
<code>\tl_if_empty_p:N</code>	1793, 1975	use commands:	
<code>\tl_if_head_is_space:nTF</code>	442	<code>\use:N</code>	43, 2255, 2937, 2966
<code>\tl_new:N</code>	541, 542, 1562, 1746, 2341, 2345, 3225, 3226	<code>\use:n</code>	59, 827, 853, 908, 1064, 1074, 1087, 1264, 1388, 1453, 1465, 1477, 1637, 2025, 2109, 2131
<code>\tl_put_right:Nn</code>	2487	<code>\use_none:n</code>	1654, 2483
<code>\tl_set:Nn</code> 444, 454, 466, 469, 472, 474, 551, 552, 997, 1009, 1751, 1767, 1852, 2346, 2505, 3234, 3235, 3281, 3282		V
<code>\tl_to_str:n</code>	2112, 2134, 2228, 2232, 2755, 2764, 2775, 2930, 2934	<code>\value</code>	2391
<code>\tl_use:N</code>	759, 872	vbox commands:	
		<code>\vbox_set:Nn</code>	2491
		<code>\vbox_to_zero:n</code>	2552, 2559, 3046, 3057
		<code>\vbox_unpack_drop:N</code>	2499