

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}{2021-08-04}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2021-08-04}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2021-08-04}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2021-08-04}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2021-08-04}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2021-08-04}{ }
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 `"`).

`_kernel_backend_postscript:x`

```

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:`

`_kernel_backend_align_end:`

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

```

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:`

`_kernel_backend_scope_end:`

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

```

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 $\langle *luatex | pdftex \rangle$

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 $\backslash special\{pdf:\}$ 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   \langle *luatex \rangle
92   \tex_pdfextension:D literal
93   \langle /luatex \rangle
94   \langle *pdftex \rangle
95   \tex_pdfliteral:D
96   \langle /pdftex \rangle
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   \langle *luatex \rangle
103   \tex_pdfextension:D literal ~
104   \langle /luatex \rangle
105   \langle *pdftex \rangle
106   \tex_pdfliteral:D
107   \langle /pdftex \rangle
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   \langle *luatex \rangle
113   \tex_pdfextension:D save \scan_stop:
114   \langle /luatex \rangle
115   \langle *pdftex \rangle
116   \tex_pdfsave:D
117   \langle /pdftex \rangle
118 }
119 \cs_new_protected:Npn \_kernel_backend_scope_end:
120 {
121   \langle *luatex \rangle
122   \tex_pdfextension:D restore \scan_stop:
123   \langle /luatex \rangle
124   \langle *pdftex \rangle
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 XeTeX. 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 XeTeX as required. 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.

`_kernel_backend_literal_pdf:n`
`_kernel_backend_literal_pdf:x`

```

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 `xdvidfpmx (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 <@@=sys>

```

`\c__kernel_sys_dvipdfmx_version_int` A short excursion into the `sys` module to set up the backend version information.

```

151 \group_begin:
152 \cs_set:Npn \__sys_tmp:w #1 Version ~ #2 ~ #3 \q_stop {#2}
153 \sys_get_shell:nnNTF { extractbb--version }
154 { \char_set_catcode_space:n { '\ } }
155 \l__sys_internal_tl
156 {
157   \int_const:Nn \c__kernel_sys_dvipdfmx_version_int
158   {
159     \exp_after:wN \__sys_tmp:w \l__sys_internal_tl
160     \q_stop
161   }
162 }
163 { \int_const:Nn \c__kernel_sys_dvipdfmx_version_int { 0 } }
164 \group_end:

```

(End definition for `\c__kernel_sys_dvipdfmx_version_int`.)

```

165 <@@=
166 </dvipdfmx | xetex>

```

1.4 dvisvgm backend

```

167 <*dvisvgm>

```

`__kernel_backend_literal_svg:n` 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.

```

168 \cs_new_protected:Npn \__kernel_backend_literal_svg:n #1
169 { \__kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }
170 \cs_generate_variant:Nn \__kernel_backend_literal_svg:n { x }

```

(End definition for `__kernel_backend_literal_svg:n`.)

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

`\l__kernel_backend_scope_int`

```

171 \int_new:N \g__kernel_backend_scope_int
172 \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:` 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.

`__kernel_backend_scope_end:`

`__kernel_backend_scope_begin:n`

`__kernel_backend_scope_begin:x`

`__kernel_backend_scope:n`

`__kernel_backend_scope:x`

```

173 \cs_new_protected:Npn \__kernel_backend_scope_begin:
174 {
175   \__kernel_backend_literal_svg:n { <g> }
176   \int_set_eq:NN
177   \l__kernel_backend_scope_int
178   \g__kernel_backend_scope_int
179   \group_begin:
180   \int_gset:Nn \g__kernel_backend_scope_int { 1 }

```

```

181 }
182 \cs_new_protected:Npn \__kernel_backend_scope_end:
183 {
184   \prg_replicate:nn
185     { \g__kernel_backend_scope_int }
186     { \__kernel_backend_literal_svg:n { </g> } }
187   \group_end:
188   \int_gset_eq:NN
189     \g__kernel_backend_scope_int
190     \l__kernel_backend_scope_int
191 }
192 \cs_new_protected:Npn \__kernel_backend_scope_begin:n #1
193 {
194   \__kernel_backend_literal_svg:n { <g ~ #1 > }
195   \int_set_eq:NN
196     \l__kernel_backend_scope_int
197     \g__kernel_backend_scope_int
198   \group_begin:
199     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
200 }
201 \cs_generate_variant:Nn \__kernel_backend_scope_begin:n { x }
202 \cs_new_protected:Npn \__kernel_backend_scope:n #1
203 {
204   \__kernel_backend_literal_svg:n { <g ~ #1 > }
205   \int_gincr:N \g__kernel_backend_scope_int
206 }
207 \cs_generate_variant:Nn \__kernel_backend_scope:n { x }

```

(End definition for __kernel_backend_scope_begin: and others.)

```

208 </dvisvgm>
209 </package>

```

2 l3backend-box Implementation

```

210 <*package>
211 <@@=box>

```

2.1 dvips backend

```

212 <*dvips>

```

__box_backend_clip:N The **dvips** backend scales all absolute dimensions based on the output resolution selected and any T_EX magnification. Thus for any operation involving absolute lengths there is a correction to make. See **normalscale** from **special.pro** for the variables, noting that here everything is saved on the stack rather than as a separate variable. Once all of that is done, the actual clipping is trivial.

```

213 \cs_new_protected:Npn \__box_backend_clip:N #1
214 {
215   \__kernel_backend_scope_begin:
216   \__kernel_backend_align_begin:
217   \__kernel_backend_literal_postscript:n { matrix~currentmatrix }
218   \__kernel_backend_literal_postscript:n
219     { Resolution~72~div~VResolution~72~div~scale }

```

```

220 \__kernel_backend_literal_postscript:n { DVImag-dup~scale }
221 \__kernel_backend_literal_postscript:x
222 {
223   0 ~
224   \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
225   \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
226   \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
227   rectclip
228 }
229 \__kernel_backend_literal_postscript:n { setmatrix }
230 \__kernel_backend_align_end:
231 \hbox_overlap_right:n { \box_use:N #1 }
232 \__kernel_backend_scope_end:
233 \skip_horizontal:n { \box_wd:N #1 }
234 }

```

(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.

```

235 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
236 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
237 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
238 {
239   \__kernel_backend_scope_begin:
240   \__kernel_backend_align_begin:
241   \__kernel_backend_literal_postscript:x
242   {
243     \fp_compare:nNnTF {#2} = \c_zero_fp
244     { 0 }
245     { \fp_eval:n { round ( -(#2) , 5 ) } } ~
246     rotate
247   }
248   \__kernel_backend_align_end:
249   \box_use:N #1
250   \__kernel_backend_scope_end:
251 }

```

(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.

```

252 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
253 {
254   \__kernel_backend_scope_begin:
255   \__kernel_backend_align_begin:
256   \__kernel_backend_literal_postscript:x
257   {
258     \fp_eval:n { round ( #2 , 5 ) } ~
259     \fp_eval:n { round ( #3 , 5 ) } ~
260     scale
261   }
262   \__kernel_backend_align_end:
263   \hbox_overlap_right:n { \box_use:N #1 }

```



```

264     \__kernel_backend_scope_end:
265 }

(End definition for \__box_backend_scale:Nnn.)

266 </dvips>

```

2.2 LuaTeX and pdfTeX backends

```

267 <*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.

```

268 \cs_new_protected:Npn \__box_backend_clip:N #1
269 {
270     \__kernel_backend_scope_begin:
271     \__kernel_backend_literal_pdf:x
272     {
273         0~
274         \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
275         \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
276         \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
277         re~W~n
278     }
279     \hbox_overlap_right:n { \box_use:N #1 }
280     \__kernel_backend_scope_end:
281     \skip_horizontal:n { \box_wd:N #1 }
282 }

```

(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.

```

283 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
284 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
285 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
286 {
287     \__kernel_backend_scope_begin:
288     \box_set_wd:Nn #1 { Opt }
289     \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
290     \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
291     { \fp_zero:N \l__box_backend_cos_fp }
292     \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
293     \__kernel_backend_matrix:x
294     {
295         \fp_use:N \l__box_backend_cos_fp \c_space_tl
296         \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp

```

```

297         { 0~0 }
298         {
299             \fp_use:N \l__box_backend_sin_fp
300             \c_space_tl
301             \fp_eval:n { -\l__box_backend_sin_fp }
302         }
303         \c_space_tl
304         \fp_use:N \l__box_backend_cos_fp
305     }
306     \box_use:N #1
307     \__kernel_backend_scope_end:
308 }
309 \fp_new:N \l__box_backend_cos_fp
310 \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.

```

311 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
312 {
313     \__kernel_backend_scope_begin:
314     \__kernel_backend_matrix:x
315     {
316         \fp_eval:n { round ( #2 , 5 ) } ~
317         0~0~
318         \fp_eval:n { round ( #3 , 5 ) }
319     }
320     \hbox_overlap_right:n { \box_use:N #1 }
321     \__kernel_backend_scope_end:
322 }

```

(End definition for __box_backend_scale:Nnn.)

323 </luatex | pdftex>

2.3 dvipdfmx/X_YTeX backend

324 <*dvipdfmx | xetex>

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

```

325 \cs_new_protected:Npn \__box_backend_clip:N #1
326 {
327     \__kernel_backend_scope_begin:
328     \__kernel_backend_literal_pdf:x
329     {
330         0~
331         \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
332         \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
333         \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
334         re~W~n
335     }
336     \hbox_overlap_right:n { \box_use:N #1 }
337     \__kernel_backend_scope_end:
338     \skip_horizontal:n { \box_wd:N #1 }
339 }

```

(End definition for `_box_backend_clip:N`.)

`_box_backend_rotate: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.

`_box_backend_rotate_aux:Nn`

```

340 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
341 { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
342 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
343 {
344   \_kernel_backend_scope_begin:
345   \_kernel_backend_literal:x
346   {
347     x:rotate~
348     \fp_compare:nNnTF {#2} = \c_zero_fp
349     { 0 }
350     { \fp_eval:n { round ( #2 , 5 ) } }
351   }
352   \box_use:N #1
353   \_kernel_backend_scope_end:
354 }

```

(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.

```

355 \cs_new_protected:Npn \_box_backend_scale:Nnn #1#2#3
356 {
357   \_kernel_backend_scope_begin:
358   \_kernel_backend_literal:x
359   {
360     x:scale~
361     \fp_eval:n { round ( #2 , 5 ) } ~
362     \fp_eval:n { round ( #3 , 5 ) }
363   }
364   \hbox_overlap_right:n { \box_use:N #1 }
365   \_kernel_backend_scope_end:
366 }

```

(End definition for `_box_backend_scale:Nnn`.)

367 `</dvipdfmx | xetex>`

2.4 dvisvgm backend

368 `<*dvisvgm>`

`_box_backend_clip:N`

`\g__box_clip_path_int`

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

down using scopes to allow for the depth of the \TeX box and keep the reference point the same!

```

369 \cs_new_protected:Npn \__box_backend_clip:N #1
370 {
371   \int_gincr:N \g__box_clip_path_int
372   \__kernel_backend_literal_svg:x
373   { < clipPath~id = " l3cp \int_use:N \g__box_clip_path_int " > }
374   \__kernel_backend_literal_svg:x
375   {
376     <
377     path ~ d =
378     "
379       M ~ 0 ~
380       \dim_to_decimal:n { -\box_dp:N #1 } ~
381       L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
382       \dim_to_decimal:n { -\box_dp:N #1 } ~
383       L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
384       \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
385       L ~ 0 ~
386       \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
387       Z
388     "
389     />
390   }
391   \__kernel_backend_literal_svg:n
392   { < /clipPath > }

```

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

```

393 \__kernel_backend_scope_begin:n
394 {
395   transform =
396   "
397     translate ( { ?x } , { ?y } ) ~
398     scale ( 1 , -1 )
399   "
400 }
401 \__kernel_backend_scope:x
402 {
403   clip-path =
404   "url ( \c_hash_str l3cp \int_use:N \g__box_clip_path_int ) "
405 }
406 \__kernel_backend_scope:n
407 {
408   transform =
409   "
410     scale ( -1 , 1 ) ~
411     translate ( { ?x } , { ?y } ) ~
412     scale ( -1 , -1 )
413   "
414 }

```

```

415     \box_use:N #1
416     \__kernel_backend_scope_end:
417   }
418   \int_new:N \g__box_clip_path_int
(End definition for \__box_backend_clip:N and \g__box_clip_path_int.)

```

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

```

419   \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
420   {
421     \__kernel_backend_scope_begin:x
422     {
423       transform =
424       "
425         rotate
426         ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
427       "
428     }
429     \box_use:N #1
430     \__kernel_backend_scope_end:
431   }
(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).

```

432   \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
433   {
434     \__kernel_backend_scope_begin:x
435     {
436       transform =
437       "
438         translate ( { ?x } , { ?y } ) ~
439         scale
440         (
441           \fp_eval:n { round ( -#2 , 5 ) } ,
442           \fp_eval:n { round ( -#3 , 5 ) }
443         ) ~
444         translate ( { ?x } , { ?y } ) ~
445         scale ( -1 )
446       "
447     }
448     \hbox_overlap_right:n { \box_use:N #1 }
449     \__kernel_backend_scope_end:
450   }
(End definition for \__box_backend_scale:Nnn.)
451   </dvisvgm>
452   </package>

```

3 l3backend-color Implementation

```

453 <*package>
454 <@@=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_AT_EX in particular. Whilst it is in some ways convenient to use the same approach in multiple backends, the fact that dvipdfmx/X_YL_AT_EX 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

```

455 <*dvisvgm | dvipdfmx | dvips | xetex>

```

Allow for L^AT_EX 2_ε color. Here, the possible input values are limited: dvips-style colors can mainly be taken as-is with the exception spot ones (here we need a model and a tint). The x-type expansion is there to cover the case where xcolor is in use.

```

456 \cs_new_protected:Npn \__color_backend_pickup:N #1 { }
457 \cs_if_exist:cT { ver@color.sty }
458 {
459   \cs_set_protected:Npn \__color_backend_pickup:N #1
460   {
461     \exp_args:NW \tl_if_head_is_space:nTF \current@color
462     {
463       \tl_set:Nx #1
464       {
465         { \exp_after:wN \use:n \current@color }
466         { 1 }
467       }
468     }
469     {
470       \exp_last_unbraced:Nx \__color_backend_pickup:w
471       { \current@color } \s__color_stop #1
472     }
473   }
474   \cs_new_protected:Npn \__color_backend_pickup:w #1 ~ #2 \s__color_stop #3
475   { \tl_set:Nn #3 { {#1} {#2} } }
476 }

```

(End definition for __color_backend_pickup:N and __color_backend_pickup:w.)

```

477 </dvisvgm | dvipdfmx | dvips | xetex>

```

3.1.2 Lua_T_EX and pdf_T_EX

```

478 <*luatex | pdftex>

```

The current color in driver-dependent format: pick up the package-mode data if available. We end up converting back and forward in this route as we store our color data in dvips format. The \current@color needs to be x-expanded before __color_backend_pickup:w breaks it apart, because for instance xcolor sets it to be instructions to generate a color

```

479 \cs_new_protected:Npn \__color_backend_pickup:N #1 { }
480 \cs_if_exist:cT { ver@color.sty }

```

```

481 {
482   \cs_set_protected:Npn \__color_backend_pickup:N #1
483   {
484     \exp_last_unbraced:Nx \__color_backend_pickup:w
485     { \current@color } ~ 0 ~ 0 ~ 0 \s__color_stop #1
486   }
487   \cs_new_protected:Npn \__color_backend_pickup:w
488   #1 ~ #2 ~ #3 ~ #4 ~ #5 ~ #6 \s__color_stop #7
489   {
490     \str_if_eq:nnTF {#2} { g }
491     { \tl_set:Nn #7 { { gray } {#1} } }
492     {
493       \str_if_eq:nnTF {#4} { rg }
494       { \tl_set:Nn #7 { { rgb } { #1 ~ #2 ~ #3 } } }
495       {
496         \str_if_eq:nnTF {#5} { k }
497         { \tl_set:Nn #7 { { cmyk } { #1 ~ #2 ~ #3 ~ #4 } } }
498         {
499           \str_if_eq:nnTF {#2} { cs }
500           {
501             \tl_set:Nx #7 { { \use:n #1 } { #5 } }
502           }
503           {
504             \tl_set:Nn #7 { { gray } { 0 } }
505           }
506         }
507       }
508     }
509   }
510 }

```

(End definition for __color_backend_pickup:N and __color_backend_pickup:w.)

511 </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. The exact form depends on the engine, and for dvipdfmx/X_YTeX the backend version.

3.2.1 Common code

512 <*dvipdfmx | luatex | pdftex | xetex>

\l__color_backend_stack_int pdfTeX, LuaTeX and recent (x)dvipdfmx have multiple stacks available, and to track which one is in use a variable is required.

513 \int_new:N \l__color_backend_stack_int

(End definition for \l__color_backend_stack_int.)

514 </dvipdfmx | luatex | pdftex | xetex>

3.2.2 dvipdfmx/X_YTeX

515 $\langle *dvipdfmx | xetex \rangle$

In (x)dvipdfmx, the base color stack is not set up, so we have to force that, as well as providing a mechanism more generally.

```

516 \int_compare:nNnTF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
517 { \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3 { } }
518 {
519   \int_new:N \g__color_backend_stack_int
520   \cs_new_protected:Npx \__kernel_color_backend_stack_init:Nnn #1#2#3
521   {
522     \int_gincr:N \exp_not:N \g__color_backend_stack_int
523     \int_const:Nn #1 { \exp_not:N \g__color_backend_stack_int }
524     \use:x
525     {
526       \__kernel_backend_first_shipout:n
527       {
528         \__kernel_backend_literal:n
529         {
530           pdfcolorstackinit ~
531           \exp_not:N \int_use:N \exp_not:N \g__color_backend_stack_int
532           \c_space_tl
533           \exp_not:N \tl_if_blank:nF {#2} { #2 ~ }
534           (#3)
535         }
536       }
537     }
538   }
539   \cs_if_exist:cTF { main@pdfcolorstack }
540   {
541     \int_set:Nn \l__color_backend_stack_int
542     { \int_use:c { main@pdfcolorstack } }
543   }
544   {
545     \__kernel_color_backend_stack_init:Nnn \c__color_backend_main_stack_int
546     { page ~ direct } { 0 ~ g ~ 0 ~ G }
547     \int_set_eq:NN \l__color_backend_stack_int
548     \c__color_backend_main_stack_int
549     \int_const:cn { main@pdfcolorstack } { \c__color_backend_main_stack_int }
550   }

```

The backend automatically restores the stack color from the “classical” approach (pdf:bcolor) after a scope. That will be an issue for us, so we manually ensure that the one we are using is inserted.

```

551   \cs_gset_protected:Npn \__kernel_backend_scope_end:
552   {
553     \__kernel_backend_literal:n { x:grestore }
554     \__kernel_backend_literal:n
555     { pdfcolorstack ~ \g__color_backend_stack_int current }
556   }
557 }

```

(End definition for __kernel_color_backend_stack_init:Nnn, \g__color_backend_stack_int, and \c__color_backend_main_stack_int.)


```

\__kernel_color_backend_stack_push:nn
\__kernel_color_backend_stack_push:nx
\__kernel_color_backend_stack_pop:n

```

Simple enough but needs a version check.

```

558 \int_compare:nNnF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
559 {
560   \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
561   {
562     \__kernel_backend_literal:x
563     {
564       pdfcolorstack ~
565       \int_eval:n {#1} ~
566       push ~ (#2)
567     }
568   }
569   \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
570   \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
571   {
572     \__kernel_backend_literal:x
573     {
574       pdfcolorstack ~
575       \int_eval:n {#1} ~
576       pop
577     }
578   }
579 }

```

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

```

580 </dvipdfmx | xetex>

```

3.2.3 LuaTeX and pdfTeX

```

581 <*luatex | pdftex>

```

```

\__kernel_color_backend_stack_init:Nnn

```

```

582 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
583 {
584   \int_const:Nn #1
585   {
586     <*luatex>
587     \tex_pdffeedback:D colorstackinit ~
588     </luatex>
589     <*pdftex>
590     \tex_pdfcolorstackinit:D
591     </pdftex>
592     \tl_if_blank:nF {#2} { #2 ~ }
593     {#3}
594   }
595 }

```

(End definition for __kernel_color_backend_stack_init:Nnn.)

```

\__kernel_color_backend_stack_push:nn
\__kernel_color_backend_stack_push:nx
\__kernel_color_backend_stack_pop:n

```

```

596 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
597 {
598   <*luatex>

```

```

599     \tex_pdfextension:D colorstack ~
600   </luatex>
601   <*pdfTex>
602     \tex_pdfcolorstack:D
603   </pdfTex>
604     \int_eval:n {#1} ~ push ~ {#2}
605   }
606   \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
607   \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
608   {
609     <*luatex>
610     \tex_pdfextension:D colorstack ~
611   </luatex>
612   <*pdfTex>
613     \tex_pdfcolorstack:D
614   </pdfTex>
615     \int_eval:n {#1} ~ pop \scan_stop:
616   }

(End definition for \__kernel_color_backend_stack_push:nn and \__kernel_color_backend_stack_
pop:n.)

617 </luatex | pdfTex>

```

3.3 General color

3.3.1 dvips-style

```

618 <*dvips | dvisvgm>

\__color_backend_select_cmyk:n Push the data to the stack. In the case of dvips also saves the drawing color in raw
\__color_backend_select_gray:n PostScript.
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc

619 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
620 { \__color_backend_select:n { cmyk ~ #1 } }
621 \cs_new_protected:Npn \__color_backend_select_gray:n #1
622 { \__color_backend_select:n { gray ~ #1 } }
623 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
624 { \__color_backend_select:n { rgb ~ #1 } }
625 \cs_new_protected:Npn \__color_backend_select:n #1
626 {
627   \__kernel_backend_literal:n { color~push~ #1 }
628 <*dvips>
629   \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
630 </dvips>
631   \group_insert_after:N \__color_backend_reset:
632 }
633 \cs_new_protected:Npn \__color_backend_reset:
634 { \__kernel_backend_literal:n { color~pop } }

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

635 </dvips | dvisvgm>

```

3.3.2 LuaTeX and pdfTeX

636 $\langle *dvipdfmx \mid \text{luatex} \mid \text{pdfTeX} \mid \text{xetex} \rangle$

$\backslash l_color_backend_fill_tl$
 $\backslash l_color_backend_stroke_tl$

637 $\backslash tl_new:N \backslash l_color_backend_fill_tl$
 638 $\backslash tl_new:N \backslash l_color_backend_stroke_tl$

(End definition for $\backslash l_color_backend_fill_tl$ and $\backslash l_color_backend_stroke_tl$.)

$\backslash_color_backend_select_cmyk:n$
 $\backslash_color_backend_select_gray:n$
 $\backslash_color_backend_select_rgb:n$
 $\backslash_color_backend_select:nn$
 $\backslash_color_backend_reset:$

Store the values then pass to the stack.

639 $\backslash cs_new_protected:Npn \backslash_color_backend_select_cmyk:n \#1$
 640 $\{ \backslash_color_backend_select:nn \{ \#1 \sim k \} \{ \#1 \sim K \} \}$
 641 $\backslash cs_new_protected:Npn \backslash_color_backend_select_gray:n \#1$
 642 $\{ \backslash_color_backend_select:nn \{ \#1 \sim g \} \{ \#1 \sim G \} \}$
 643 $\backslash cs_new_protected:Npn \backslash_color_backend_select_rgb:n \#1$
 644 $\{ \backslash_color_backend_select:nn \{ \#1 \sim rg \} \{ \#1 \sim RG \} \}$
 645 $\backslash cs_new_protected:Npn \backslash_color_backend_select:nn \#1\#2$
 646 $\{$
 647 $\backslash tl_set:Nn \backslash l_color_backend_fill_tl \{ \#1 \}$
 648 $\backslash tl_set:Nn \backslash l_color_backend_stroke_tl \{ \#2 \}$
 649 $\backslash_kernel_color_backend_stack_push:nn \backslash l_color_backend_stack_int \{ \#1 \sim \#2 \}$
 650 $\backslash group_insert_after:N \backslash_color_backend_reset:$
 651 $\}$
 652 $\backslash cs_new_protected:Npn \backslash_color_backend_reset:$
 653 $\{ \backslash_kernel_color_backend_stack_pop:n \backslash l_color_backend_stack_int \}$

(End definition for $\backslash_color_backend_select_cmyk:n$ and others.)

654 $\langle /dvipdfmx \mid \text{luatex} \mid \text{pdfTeX} \mid \text{xetex} \rangle$

3.3.3 dvipdfmx/X_YTeX

655 $\langle *dvipdfmx \mid \text{xetex} \rangle$

These backends have the most possible approaches: it recognises both dvips-based color specials and it's 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. Thus it is used in preference to the dvips-style interface or the “native” color specials (which have only one stack).

Push the data to the stack.

$\backslash_color_backend_select_cmyk:n$
 $\backslash_color_backend_select_gray:n$
 $\backslash_color_backend_select_rgb:n$
 $\backslash_color_backend_reset:$

656 $\backslash int_compare:nNnT \backslash c_kernel_sys_dvipdfmx_version_int < \{ 20201111 \}$
 657 $\{$
 658 $\backslash cs_gset_protected:Npn \backslash_color_backend_select_cmyk:n \#1$
 659 $\{$
 660 $\backslash_kernel_backend_literal:n \{ pdf: bc \sim [\#1] \}$
 661 $\backslash group_insert_after:N \backslash_color_backend_reset:$
 662 $\}$
 663 $\backslash cs_gset_eq:NN \backslash_color_backend_select_gray:n \backslash_color_backend_select_cmyk:n$
 664 $\backslash cs_gset_eq:NN \backslash_color_backend_select_rgb:n \backslash_color_backend_select_cmyk:n$
 665 $\backslash cs_gset_protected:Npn \backslash_color_backend_reset:$
 666 $\{ \backslash_kernel_backend_literal:n \{ pdf: ec \} \}$
 667 $\}$

(End definition for $\backslash_color_backend_select_cmyk:n$ and others.)

668 $\langle /dvipdfmx \mid \text{xetex} \rangle$

3.4 Separations

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

669 \langle *dvips \rangle

$\backslash_color_backend_select_separation:nn$
 $\backslash_color_backend_select_devicen:nn$

670 $\backslash cs_new_protected:Npn \backslash_color_backend_select_separation:nn \#1\#2$
 671 $\{ \backslash_color_backend_select:n \{ separation \sim \#1 \sim \#2 \} \}$
 672 $\backslash cs_new_eq:NN \backslash_color_backend_select_devicen:nn \backslash_color_backend_select_separation:nn$

(End definition for $\backslash_color_backend_select_separation:nn$ and $\backslash_color_backend_select_devicen:nn$.)

Initialising here means creating a small header set up plus massaging some data. This comes about as we have to deal with PDF-focussed data, which makes most sense “higher-up”. The approach is based on ideas from <https://tex.stackexchange.com/q/560093> plus using the PostScript manual for other aspects.

$\backslash_color_backend_separation_init:nnnnn$
 $\backslash_color_backend_separation_init:nxxnn$
 $\backslash_color_backend_separation_init_aux:nnnnn$
 $\backslash color_backend_separation_init_DeviceCMYK:nnn$
 $\backslash color_backend_separation_init_DeviceGray:nnn$
 $\backslash color_backend_separation_init_DeviceRGB:nnn$
 $\backslash_color_backend_separation_init_Device:Nn$
 $\backslash_color_backend_separation_init:nnn$
 $\backslash_color_backend_separation_init_count:n$
 $\backslash_color_backend_separation_init_count:w$
 $\backslash_color_backend_separation_init:nnnn$
 $\backslash_color_backend_separation_init:w$
 $\backslash_color_backend_separation_init:n$
 $\backslash_color_backend_separation_init:nw$
 $\backslash_color_backend_separation_init_CIELAB:nnn$

673 $\backslash cs_new_protected:Npx \backslash_color_backend_separation_init:nnnnn \#1\#2\#3\#4\#5$
 674 $\{$
 675 $\backslash bool_if:NT \backslash g_kernel_backend_header_bool$
 676 $\{$
 677 $\backslash_kernel_backend_first_shipout:n$
 678 $\{$
 679 $\backslash exp_not:N \backslash_color_backend_separation_init_aux:nnnnn$
 680 $\{ \#1 \} \{ \#2 \} \{ \#3 \} \{ \#4 \} \{ \#5 \}$
 681 $\}$
 682 $\}$
 683 $\}$
 684 $\backslash cs_generate_variant:Nn \backslash_color_backend_separation_init:nnnnn \{ nxx \}$
 685 $\backslash cs_new_protected:Npn \backslash_color_backend_separation_init_aux:nnnnn \#1\#2\#3\#4\#5$
 686 $\{$
 687 $\backslash_kernel_backend_literal:e$
 688 $\{$
 689 $!$
 690 $TeXDict \sim begin \sim$
 691 $/color \backslash int_use:N \backslash g_color_model_int$
 692 $\{$
 693 $[\sim$
 694 $/Separation \sim (\backslash str_convert_pdfname:n \{ \#1 \}) \sim$
 695 $[\sim \#2 \sim] \sim$
 696 $\{$
 697 $\backslash cs_if_exist_use:cF \{ \backslash_color_backend_separation_init_ \#2 :nnn \}$
 698 $\{ \backslash_color_backend_separation_init:nnn \}$
 699 $\{ \#3 \} \{ \#4 \} \{ \#5 \}$
 700 $\}$
 701 $] \sim setcolorspace$
 702 $\} \sim def \sim$
 703 end
 704 $\}$
 705 $\}$
 706 $\backslash cs_new:cpn \{ \backslash_color_backend_separation_init_ /DeviceCMYK :nnn \} \#1\#2\#3$
 707 $\{ \backslash_color_backend_separation_init_Device:Nn 4 \{ \#3 \} \}$
 708 $\backslash cs_new:cpn \{ \backslash_color_backend_separation_init_ /DeviceGray :nnn \} \#1\#2\#3$
 709 $\{ \backslash_color_backend_separation_init_Device:Nn 1 \{ \#3 \} \}$
 710 $\backslash cs_new:cpn \{ \backslash_color_backend_separation_init_ /DeviceRGB :nnn \} \#1\#2\#3$

```

711 { \_color_backend_separation_init_Device:Nn 2 {#3} }
712 \cs_new:Npn \_color_backend_separation_init_Device:Nn #1#2
713 {
714   #2 ~
715   \prg_replicate:nn {#1}
716   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
717   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
718 }

```

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.

```

719 \cs_new:Npn \_color_backend_separation_init:nnn #1#2#3
720 {
721   \exp_args:Ne \_color_backend_separation_init:nnnn
722   { \_color_backend_separation_init_count:n {#2} }
723   {#1} {#2} {#3}
724 }
725 \cs_new:Npn \_color_backend_separation_init_count:n #1
726 { \int_eval:n { 0 \_color_backend_separation_init_count:w #1 ~ \s_color_stop } }
727 \cs_new:Npn \_color_backend_separation_init_count:w #1 ~ #2 \s_color_stop
728 {
729   +1
730   \tl_if_blank:nF {#2}
731   { \_color_backend_separation_init_count:w #2 \s_color_stop }
732 }

```

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.

```

733 \cs_new:Npn \_color_backend_separation_init:nnnn #1#2#3#4
734 {
735   \_color_backend_separation_init:w #3 ~ \s_color_stop #4 ~ \s_color_stop
736   \prg_replicate:nn {#1}
737   {
738     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
739     \int_eval:n { 3 * #1 } ~ index ~ mul ~
740     2 ~ index ~ add ~
741     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
742   }
743   \int_step_function:nnnN {#1} { -1 } { 1 }
744   \_color_backend_separation_init:n
745   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
746   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
747   \tl_if_blank:nF {#2}

```

```

748     { \_color_backend_separation_init:nw {#1} #2 ~ \s_color_stop }
749   }
750   \cs_new:Npn \_color_backend_separation_init:w
751     #1 ~ #2 \s_color_stop #3 ~ #4 \s_color_stop
752   {
753     #1 ~ #3 ~ 0 ~
754     \tl_if_blank:nF {#2}
755     { \_color_backend_separation_init:w #2 \s_color_stop #4 \s_color_stop }
756   }
757   \cs_new:Npn \_color_backend_separation_init:n #1
758   { \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.

```

759   \cs_new:Npn \_color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s_color_stop
760   {
761     #2 ~ #3 ~
762     2 ~ index ~ 2 ~ index ~ lt ~
763     { ~ pop ~ excl ~ pop ~ } ~
764     { ~
765       2 ~ index ~ 1 ~ index ~ gt ~
766       { ~ excl ~ pop ~ excl ~ pop ~ } ~
767       { ~ pop ~ pop ~ } ~
768       ifelse ~
769     }
770     ifelse ~
771     #1 ~ 1 ~ roll ~
772     \tl_if_blank:nF {#4}
773     { \_color_backend_separation_init:nw {#1} #4 \s_color_stop }
774   }

```

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.

```

775   \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
776   {
777     \_color_backend_separation_init:nxxxnn
778     {#2}
779     {
780       /CIEBasedABC ~
781       << ~
782       /RangeABC ~ [ ~ \c_color_model_range_CIELAB_tl \c_space_tl ] ~
783       /DecodeABC ~
784       [ ~
785         { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
786         { ~ 500 ~ div ~ } ~ bind ~
787         { ~ 200 ~ div ~ } ~ bind ~
788       ] ~
789       /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
790       /DecodeLMN ~
791       [ ~
792         { ~
793           dup ~ 6 ~ 29 ~ div ~ ge ~
794           { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
795           { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~

```

```

796         ifelse ~
797         0.9505 ~ mul ~
798     } ~ bind ~
799     { ~
800         dup ~ 6 ~ 29 ~ div ~ ge ~
801         { ~ dup ~ dup ~ mul ~ mul ~ } ~
802         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
803         ifelse ~
804     } ~ bind ~
805     { ~
806         dup ~ 6 ~ 29 ~ div ~ ge ~
807         { ~ dup ~ dup ~ mul ~ mul ~ } ~
808         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
809         ifelse ~
810         1.0890 ~ mul ~
811     } ~ bind
812 ] ~
813 /WhitePoint ~
814 [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
815 >>
816 }
817 { \c__color_model_range_CIELAB_tl }
818 { 100 ~ 0 ~ 0 }
819 {#3}
820 }

```

(End definition for _color_backend_separation_init:nnnnn and others.)

_color_backend_devicen_init:nnn Trivial as almost all of the work occurs in the shared code.

```

821 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
822 {
823     \_kernel_backend_literal:e
824     {
825         !
826         TeXDict ~ begin ~
827         /color \int_use:N \g__color_model_int
828         {
829             [ ~
830                 /DeviceN ~
831                 [ ~ #1 ~ ] ~
832                 #2 ~
833                 { ~ #3 ~ } ~
834             ] ~ setcolorspace
835         } ~ def ~
836     end
837 }
838 }

```

(End definition for _color_backend_devicen_init:nnn.)

```

839 </dvips>
840 <*dvisvgm>

```

_color_backend_select_separation:nn No support at present.

```

\_color_backend_select_devicen:nn 841 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2 { }
842 \cs_new_protected:Npn \_color_backend_select_devicen:nn #1#2 { }

```

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

No support at present.

```

843 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }
844 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnnnnn #1#2#3 { }

```

(End definition for `_color_backend_separation_init:nnnnn` and `_color_backend_separation_init_CIELAB:nnnn`.)

```
845 </dvisvgm>
```

```
846 <*dvipdfmx | luatex | pdftex | xetex>
```

Although (x)dvipdfmx has a built-in approach to color spaces, that can't be used with the generic color stacks. So we take an approach in which we share the same code as for pdfTeX.

```

847 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
848 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
849 \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`.)

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.

```

850 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5
851 {
852   \pdf_object_unnamed_write:nx { dict }
853   {
854     /FunctionType ~ 2
855     /Domain ~ [0 ~ 1]
856     \t1_if_blank:nF {#3} { /Range ~ [#3] }
857     /C0 ~ [#4] ~
858     /C1 ~ [#5] /N ~ 1
859   }
860   \_color_backend_separation_init:n
861   {
862     /Separation ~
863     / \str_convert_pdfname:n {#1} ~ #2 ~
864     \pdf_object_ref_last:
865   }
866   \bool_lazy_and:nnT
867   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
868   { \pdfmanagement_if_active_p: }
869   {
870     \use:x
871     {
872       \pdfmanagement_add:nnn
873       { Page / Resources / ColorSpace }
874       { color \int_use:N \g__color_model_int }
875       { \pdf_object_ref_last: }
876     }
877   }
878 }
879 \cs_new_protected:Npn \_color_backend_separation_init:n #1

```



```

880 {
881   \pdf_object_unnamed_write:nx { array } {#1}
882 }

```

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

```

883 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
884 {
885   \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
886   {
887     \pdf_object_new:nn { __color_illuminant_CIELAB_ #1 } { array }
888     \pdf_object_write:nx { __color_illuminant_CIELAB_ #1 }
889     {
890       /Lab ~
891       <<
892       /WhitePoint ~
893       [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
894       /Range ~ [ \c__color_model_range_CIELAB_tl ]
895       >>
896     }
897   }
898   \__color_backend_separation_init:nnnnn
899   {#2}
900   { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
901   { \c__color_model_range_CIELAB_tl }
902   { 100 ~ 0 ~ 0 }
903   {#3}
904 }

```

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

```

\__color_backend_devicen_init:nnn
\__color_backend_devicen_init:w
\__color_backend_devicen_init:n

```

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

```

905 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
906 {
907   \pdf_object_unnamed_write:nx { stream }
908   {
909     {
910       /FunctionType ~ 4 ~
911       /Domain ~
912       [ ~
913         \prg_replicate:nn
914         { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
915         { 0 ~ 1 ~ } ~
916       ] ~
917       /Range ~
918       [ ~
919         \str_case:nn {#2}
920         {
921           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
922           { /DeviceGray } { 0 ~ 1 }
923           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
924         } ~
925       ]

```

```

926     }
927     {#3}
928   }
929   \_color_backend_separation_init:n
930   {
931     /DeviceN ~
932     [ ~ #1 ~ ] ~
933     #2 ~
934     \pdf_object_ref_last:
935   }
936   \bool_lazy_and:nnT
937   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
938   { \pdfmanagement_if_active_p: }
939   {
940     \use:x
941     {
942       \pdfmanagement_add:nnn
943       { Page / Resources / ColorSpace }
944       { color \int_use:N \g__color_model_int }
945       { \pdf_object_ref_last: }
946     }
947   }
948 }
949 \cs_new:Npn \_color_backend_devicen_init:w #1 ~ #2 \s__color_stop
950 {
951   + 1
952   \tl_if_blank:nF {#2}
953   { \_color_backend_devicen_init:w #2 \s__color_stop }
954 }
955 \cs_new_eq:NN \_color_backend_devicen_init:n \_color_backend_separation_init:n
(End definition for \_color_backend_devicen_init:nnn, \_color_backend_devicen_init:w, and \_color_backend_devicen_init:n.)
956 </dvipdfmx | luatex | pdftex | xetex>
957 <*dvipdfmx | xetex>

```

_color_backend_select_separation:nn For older (x)dvipdfmx, we *could* support separations using a dedicated mechanism, but it was not added that long before the color stacks. So instead of having two complex paths, just disable here.

```

958 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
959 {
960   \cs_gset_protected:Npn \_color_backend_select_separation:nn #1#2 { }
961   \cs_gset_eq:NN \_color_backend_select_devicen:nn
962   \_color_backend_select_separation:nn
963 }
(End definition for \_color_backend_select_separation:nn and \_color_backend_select_devicen:nn.)
964 </dvipdfmx | xetex>

```

3.5 Fill and stroke color

Here, dvipdfmx/X_YTeX follows LuaTeX and pdfTeX, while for dvips we have to manage fill and stroke color ourselves. We also handle dvisvgm independently, as there we can create SVG directly.

965 <*dvipdfmx | luatex | pdftex | xetex>

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.

```

966 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
967   { \__color_backend_fill:n { #1 ~ k } }
968 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
969   { \__color_backend_fill:n { #1 ~ g } }
970 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
971   { \__color_backend_fill:n { #1 ~ rg } }
972 \cs_new_protected:Npn \__color_backend_fill:n #1
973   {
974     \tl_set:Nn \l__color_backend_fill_tl {#1}
975     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
976       { #1 ~ \l__color_backend_stroke_tl }
977     \group_insert_after:N \__color_backend_reset:
978   }
979 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
980   { \__color_backend_stroke:n { #1 ~ K } }
981 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
982   { \__color_backend_stroke:n { #1 ~ G } }
983 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
984   { \__color_backend_stroke:n { #1 ~ RG } }
985 \cs_new_protected:Npn \__color_backend_stroke:n #1
986   {
987     \tl_set:Nn \l__color_backend_stroke_tl {#1}
988     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
989       { \l__color_backend_fill_tl \c_space_tl #1 }
990     \group_insert_after:N \__color_backend_reset:
991   }

```

(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
992 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
993   { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
994 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
995   { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
996 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
997 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

998 </dvipdfmx | luatex | pdftex | xetex>

999 <*dvipdfmx | xetex>

Deal with older (x)dvipdfmx.

```

1000 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
1001   {
1002     \cs_gset_protected:Npn \__color_backend_fill_cmyk:n #1
1003       {
1004         \__kernel_backend_literal:n { pdf: bc ~ [#1] }

```

```

1005     \group_insert_after:N \__color_backend_reset:
1006   }
1007   \cs_gset_eq:NN \__color_backend_fill_gray:n \__color_backend_fill_cmyk:n
1008   \cs_gset_eq:NN \__color_backend_fill_rgb:n \__color_backend_fill_cmyk:n
1009   \cs_gset_protected:Npn \__color_backend_reset:
1010     { \__kernel_backend_literal:n { pdf: ec } }
1011   \cs_gset_protected:Npn \__color_backend_stroke:n #1
1012     { \__kernel_backend_literal:n { #1 } }
1013   \cs_gset_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1014   \cs_gset_eq:NN \__color_backend_fill_devicen:nn
1015     \__color_backend_fill_separation:nn
1016   \cs_gset_eq:NN \__color_backend_stroke_separation:nn
1017     \__color_backend_fill_separation:nn
1018   \cs_gset_eq:NN \__color_backend_stroke_devicen:nn
1019     \__color_backend_stroke_separation:nn
1020 }

```

(End definition for __color_backend_fill_cmyk:n and others.)

```
1021 </dvipdfmx | xetex>
```

```
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 1023 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
\__color_backend_fill_rgb:n 1024 { \__color_backend_fill:n { cmyk ~ #1 } }
\__color_backend_fill:n 1025 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
\__color_backend_stroke_cmyk:n 1026 { \__color_backend_fill:n { gray ~ #1 } }
\__color_backend_stroke_gray:n 1027 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
\__color_backend_stroke_rgb:n 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   \group_insert_after:N \__color_backend_reset:
1033 }
1034 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1035 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1036 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1037 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1038 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1039 { \__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 1040 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
\__color_backend_fill_devicen:nn 1041 { \__color_backend_fill:n { separation ~ #1 ~ #2 } }
\__color_backend_stroke_devicen:nn 1042 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1043 { \__kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1044 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1045 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```
1046 </dvips>
```

```
1047 <*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 1048 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
\_color_backend_fill_rgb:n 1049 { \_color_backend_fill:n { cmyk ~ #1 } }
\_color_backend_fill:n 1050 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1051 { \_color_backend_fill:n { gray ~ #1 } }
1052 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1053 { \_color_backend_fill:n { rgb ~ #1 } }
1054 \cs_new_protected:Npn \_color_backend_fill:n #1
1055 {
1056 \_kernel_backend_literal:n { color~push~ #1 }
1057 \group_insert_after:N \_color_backend_reset:
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 1059 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
\_color_backend_stroke_gray:n 1060 { \_color_backend_cmyk:w #1 \s_color_stop }
\_color_backend_stroke_gray_aux:n 1061 \cs_new_protected:Npn \_color_backend_stroke_cmyk:w
\_color_backend_stroke_rgb:n 1062 #1 ~ #2 ~ #3 ~ #4 \s_color_stop
\_color_backend_stroke_rgb:w 1063 {
\_color_backend:nnn 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.)

```

\_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.)

```

1114 </dvisvgm>
1115 </package>

```

4 l3backend-draw Implementation

```

1116 <*package>
1117 <@@=draw>

```

4.1 dvips backend

```

1118 <*dvips>

```

```

\_draw_backend_literal:n
\_draw_backend_literal:x
1119 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
1120 \cs_generate_variant:Nn \_draw_backend_literal:n { x }

```

(End definition for _draw_backend_literal:n.)

```

\_draw_backend_begin:
  \_draw_backend_end:

```

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.

```

1121 \cs_new_protected:Npn \_draw_backend_begin:
1122 {

```

```

1123     \__kernel_backend_literal:n { ps::[begin] }
1124     \__draw_backend_literal:n { @beginspecial }
1125   }
1126   \cs_new_protected:Npn \__draw_backend_end:
1127   {
1128     \__draw_backend_literal:n { @endspecial }
1129     \__kernel_backend_literal:n { ps::[end] }
1130   }

```

(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:

```

1131   \cs_new_protected:Npn \__draw_backend_scope_begin:
1132   { \__draw_backend_literal:n { save } }
1133   \cs_new_protected:Npn \__draw_backend_scope_end:
1134   { \__draw_backend_literal:n { restore } }

```

(End definition for __draw_backend_scope_begin: and __draw_backend_scope_end:.)

__draw_backend_moveto:nn

__draw_backend_lineto:nn

__draw_backend_rectangle:nnnn

__draw_backend_curveto:nnnnnn

Path creation operations mainly resolve directly to PostScript primitive steps, with only the need to convert to bp. Notice that x-type expansion is included here to ensure that any variable values are forced to literals before any possible caching. There is no native rectangular path command (without also clipping, filling or stroking), so that task is done using a small amount of PostScript.

```

1135   \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1136   {
1137     \__draw_backend_literal:x
1138     {
1139       \dim_to_decimal_in_bp:n {#1} ~
1140       \dim_to_decimal_in_bp:n {#2} ~ moveto
1141     }
1142   }
1143   \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1144   {
1145     \__draw_backend_literal:x
1146     {
1147       \dim_to_decimal_in_bp:n {#1} ~
1148       \dim_to_decimal_in_bp:n {#2} ~ lineto
1149     }
1150   }
1151   \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1152   {
1153     \__draw_backend_literal:x
1154     {
1155       \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1156       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1157       moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1158     }
1159   }
1160   \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1161   {
1162     \__draw_backend_literal:x
1163     {

```

```

1164         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1165         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1166         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1167         curveto
1168     }
1169 }

```

(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
1170 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1171 { \bool_gset_true:N \g__draw_draw_eor_bool }
1172 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1173 { \bool_gset_false:N \g__draw_draw_eor_bool }
1174 \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:
\g__draw_draw_clip_bool
1175 \cs_new_protected:Npn \__draw_backend_closepath:
1176 { \__draw_backend_literal:n { closepath } }
1177 \cs_new_protected:Npn \__draw_backend_stroke:
1178 {
1179     \__draw_backend_literal:n { gsave }
1180     \__draw_backend_literal:n { color.sc }
1181     \__draw_backend_literal:n { stroke }
1182     \__draw_backend_literal:n { grestore }
1183     \bool_if:NT \g__draw_draw_clip_bool
1184     {
1185         \__draw_backend_literal:x
1186         {
1187             \bool_if:NT \g__draw_draw_eor_bool { eo }
1188             clip
1189         }
1190     }
1191     \__draw_backend_literal:n { newpath }
1192     \bool_gset_false:N \g__draw_draw_clip_bool
1193 }
1194 \cs_new_protected:Npn \__draw_backend_closestroke:
1195 {
1196     \__draw_backend_closepath:
1197     \__draw_backend_stroke:
1198 }
1199 \cs_new_protected:Npn \__draw_backend_fill:
1200 {
1201     \__draw_backend_literal:x
1202     {
1203         \bool_if:NT \g__draw_draw_eor_bool { eo }

```



```

1204         fill
1205     }
1206     \bool_if:NT \g__draw_draw_clip_bool
1207     {
1208         \__draw_backend_literal:x
1209         {
1210             \bool_if:NT \g__draw_draw_eor_bool { eo }
1211             clip
1212         }
1213     }
1214     \__draw_backend_literal:n { newpath }
1215     \bool_gset_false:N \g__draw_draw_clip_bool
1216 }
1217 \cs_new_protected:Npn \__draw_backend_fillstroke:
1218 {
1219     \__draw_backend_literal:x
1220     {
1221         \bool_if:NT \g__draw_draw_eor_bool { eo }
1222         fill
1223     }
1224     \__draw_backend_literal:n { gsave }
1225     \__draw_backend_literal:n { color.sc }
1226     \__draw_backend_literal:n { stroke }
1227     \__draw_backend_literal:n { grestore }
1228     \bool_if:NT \g__draw_draw_clip_bool
1229     {
1230         \__draw_backend_literal:x
1231         {
1232             \bool_if:NT \g__draw_draw_eor_bool { eo }
1233             clip
1234         }
1235     }
1236     \__draw_backend_literal:n { newpath }
1237     \bool_gset_false:N \g__draw_draw_clip_bool
1238 }
1239 \cs_new_protected:Npn \__draw_backend_clip:
1240 { \bool_gset_true:N \g__draw_draw_clip_bool }
1241 \bool_new:N \g__draw_draw_clip_bool
1242 \cs_new_protected:Npn \__draw_backend_discardpath:
1243 {
1244     \bool_if:NT \g__draw_draw_clip_bool
1245     {
1246         \__draw_backend_literal:x
1247         {
1248             \bool_if:NT \g__draw_draw_eor_bool { eo }
1249             clip
1250         }
1251     }
1252     \__draw_backend_literal:n { newpath }
1253     \bool_gset_false:N \g__draw_draw_clip_bool
1254 }

```

(End definition for __draw_backend_closepath: and others.)

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_but:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1255 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1256 {
1257   \__draw_backend_literal:x
1258   {
1259     [
1260       \exp_args:Nf \use:n
1261       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1262     ] ~
1263     \dim_to_decimal_in_bp:n {#2} ~ setdash
1264   }
1265 }
1266 \cs_new:Npn \__draw_backend_dash:n #1
1267 { ~ \dim_to_decimal_in_bp:n {#1} }
1268 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1269 {
1270   \__draw_backend_literal:x
1271   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1272 }
1273 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1274 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1275 \cs_new_protected:Npn \__draw_backend_cap_but:
1276 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1277 \cs_new_protected:Npn \__draw_backend_cap_round:
1278 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1279 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1280 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1281 \cs_new_protected:Npn \__draw_backend_join_miter:
1282 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1283 \cs_new_protected:Npn \__draw_backend_join_round:
1284 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1285 \cs_new_protected:Npn \__draw_backend_join_bevel:
1286 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

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

```

`__draw_backend_cm:nnnn` In dvips, keeping the transformations in line with the engine is unfortunately not possible for scaling and rotations: even if we decompose the matrix into those operations, there is still no backend tracking (*cf.* dvipdfmx/X_YTEX). Thus we take the shortest path available and simply dump the matrix as given.

```

1287 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1288 {
1289   \__draw_backend_literal:n
1290   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1291 }

(End definition for \__draw_backend_cm:nnnn.)

```

`__draw_backend_box_use:Nnnnn` Inside a picture `@beginspecial/@endspecial` are active, which is normally a good thing but means that the position and scaling would be off if the box was inserted directly. To deal with that, there are a number of possible approaches. The implementation here was suggested by Tom Rokici (author of dvips). We end the current special placement, then set the current point with a literal `[begin]`. As for general literals, we then use the stack to store the current point and move to it. To insert the required transformation, we have

to flip the y -axis, once before and once after it. Then we get back to the \TeX reference point to insert our content. The clean up has to happen in the right places, hence the `[begin]/[end]` pair around `restore`. Finally, we can return to “normal” drawing mode. Notice that the set up here is very similar to that in `__draw_align_currentpoint_...`, but the ordering of saving and restoring is different (intermixed).

```

1292 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1293 {
1294   \__draw_backend_literal:n { @endspecial }
1295   \__draw_backend_literal:n { [end] }
1296   \__draw_backend_literal:n { [begin] }
1297   \__draw_backend_literal:n { save }
1298   \__draw_backend_literal:n { currentpoint }
1299   \__draw_backend_literal:n { currentpoint~translate }
1300   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1301   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1302   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1303   \__draw_backend_literal:n { neg~exch~neg~exch~translate }
1304   \__draw_backend_literal:n { [end] }
1305   \hbox_overlap_right:n { \box_use:N #1 }
1306   \__draw_backend_literal:n { [begin] }
1307   \__draw_backend_literal:n { restore }
1308   \__draw_backend_literal:n { [end] }
1309   \__draw_backend_literal:n { [begin] }
1310   \__draw_backend_literal:n { @beginspecial }
1311 }

```

(End definition for `__draw_backend_box_use:Nnnnn`.)

```

1312 </dvips>

```

4.2 Lua \TeX , pdf \TeX , dvipdfmx and X \TeX

Lua \TeX , pdf \TeX , dvipdfmx and X \TeX directly produce PDF output and understand a shared set of specials for drawing commands.

```

1313 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

```

\__draw_backend_literal:n Pass data through using a dedicated interface.
\__draw_backend_literal:x
1314 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1315 \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:
1316 \cs_new_protected:Npn \__draw_backend_begin:
1317 { \__draw_backend_scope_begin: }
1318 \cs_new_protected:Npn \__draw_backend_end:
1319 { \__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:`

```

1320 \cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:
1321 \cs_new_eq:NN \_draw_backend_scope_end: \_kernel_backend_scope_end:

```

(End definition for `_draw_backend_scope_begin:` and `_draw_backend_scope_end:.`)

`_draw_backend_moveto:nn` Path creation operations all resolve directly to PDF primitive steps, with only the need

`_draw_backend_lineto:nn` to convert to bp.

`_draw_backend_curveto:nnnnnn`

`_draw_backend_rectangle:nnnn`

```

1322 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1323 {
1324   \_draw_backend_literal:x
1325   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1326 }
1327 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1328 {
1329   \_draw_backend_literal:x
1330   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1331 }
1332 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1333 {
1334   \_draw_backend_literal:x
1335   {
1336     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1337     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1338     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1339     c
1340   }
1341 }
1342 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1343 {
1344   \_draw_backend_literal:x
1345   {
1346     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1347     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1348     re
1349   }
1350 }

```

(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`

```

1351 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1352 { \bool_gset_true:N \g__draw_draw_eor_bool }
1353 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1354 { \bool_gset_false:N \g__draw_draw_eor_bool }
1355 \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:`

`_draw_backend_closestroke:`

`_draw_backend_fill:`

`_draw_backend_fillstroke:`

`_draw_backend_clip:`

`_draw_backend_discardpath:`

```

1359 { \__draw_backend_literal:n { S } }
1360 \cs_new_protected:Npn \__draw_backend_closestroke:
1361 { \__draw_backend_literal:n { s } }
1362 \cs_new_protected:Npn \__draw_backend_fill:
1363 {
1364   \__draw_backend_literal:x
1365   { f \bool_if:NT \g__draw_draw_eor_bool * }
1366 }
1367 \cs_new_protected:Npn \__draw_backend_fillstroke:
1368 {
1369   \__draw_backend_literal:x
1370   { B \bool_if:NT \g__draw_draw_eor_bool * }
1371 }
1372 \cs_new_protected:Npn \__draw_backend_clip:
1373 {
1374   \__draw_backend_literal:x
1375   { W \bool_if:NT \g__draw_draw_eor_bool * }
1376 }
1377 \cs_new_protected:Npn \__draw_backend_discardpath:
1378 { \__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
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1379 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1380 {
1381   \__draw_backend_literal:x
1382   {
1383     [
1384       \exp_args:Nf \use:n
1385       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1386     ] ~
1387     \dim_to_decimal_in_bp:n {#2} ~ d
1388   }
1389 }
1390 \cs_new:Npn \__draw_backend_dash:n #1
1391 { ~ \dim_to_decimal_in_bp:n {#1} }
1392 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1393 {
1394   \__draw_backend_literal:x
1395   { \dim_to_decimal_in_bp:n {#1} ~ w }
1396 }
1397 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1398 { \__draw_backend_literal:x { #1 ~ M } }
1399 \cs_new_protected:Npn \__draw_backend_cap_butt:
1400 { \__draw_backend_literal:n { 0 ~ J } }
1401 \cs_new_protected:Npn \__draw_backend_cap_round:
1402 { \__draw_backend_literal:n { 1 ~ J } }
1403 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1404 { \__draw_backend_literal:n { 2 ~ J } }
1405 \cs_new_protected:Npn \__draw_backend_join_miter:
1406 { \__draw_backend_literal:n { 0 ~ j } }
1407 \cs_new_protected:Npn \__draw_backend_join_round:
1408 { \__draw_backend_literal:n { 1 ~ j } }

```

```

1409 \cs_new_protected:Npn \__draw_backend_join_bevel:
1410 { \__draw_backend_literal:n { 2 ~ j } }

```

(End definition for __draw_backend_dash_pattern:nn and others.)

```

\__draw_backend_cm:nnnn
\__draw_backend_cm_aux:nnnn

```

Another split here between LuaTeX/pdfTeX and dvipdfmx/X_YTeX. In the former, we have a direct method to maintain alignment: the backend can use a matrix itself. For dvipdfmx/X_YTeX, 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_YTeX, 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!

```

1411 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1412 {
1413   <*luatex | pdftex>
1414   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1415   </luatex | pdftex>
1416   <*dvipdfmx | xetex>
1417   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1418   \__draw_backend_cm_aux:nnnn
1419   </dvipdfmx | xetex>
1420 }
1421 <*dvipdfmx | xetex>
1422 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1423 {
1424   \__kernel_backend_literal:x
1425   {
1426     x:rotate~
1427     \fp_compare:nNnTF {#1} = \c_zero_fp
1428     { 0 }
1429     { \fp_eval:n { round ( -#1 , 5 ) } }
1430   }
1431   \__kernel_backend_literal:x
1432   {
1433     x:scale~
1434     \fp_eval:n { round ( #2 , 5 ) } ~
1435     \fp_eval:n { round ( #3 , 5 ) }
1436   }
1437   \__kernel_backend_literal:x
1438   {
1439     x:rotate~
1440     \fp_compare:nNnTF {#4} = \c_zero_fp
1441     { 0 }
1442     { \fp_eval:n { round ( -#4 , 5 ) } }
1443   }
1444 }
1445 </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.

```

1446 <*dvipdfmx | xetex>
1447 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1448 {
1449   \use:x
1450   {
1451     \__draw_backend_cm_decompose_auxi:nnnnN
1452     { \fp_eval:n { (#1 + #4) / 2 } }
1453     { \fp_eval:n { (#1 - #4) / 2 } }
1454     { \fp_eval:n { (#3 + #2) / 2 } }
1455     { \fp_eval:n { (#3 - #2) / 2 } }
1456   }
1457   #5
1458 }
1459 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1460 {
1461   \use:x
1462   {
1463     \__draw_backend_cm_decompose_auxii:nnnnN
1464     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1465     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1466     { \fp_eval:n { atand ( #3 , #2 ) } }
1467     { \fp_eval:n { atand ( #4 , #1 ) } }
1468   }
1469   #5
1470 }
1471 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1472 {
1473   \use:x
1474   {

```

```

1475 \__draw_backend_cm_decompose_auxiii:nnnnN
1476 { \fp_eval:n { ( #4 - #3 ) / 2 } }
1477 { \fp_eval:n { ( #1 + #2 ) / 2 } }
1478 { \fp_eval:n { ( #1 - #2 ) / 2 } }
1479 { \fp_eval:n { ( #4 + #3 ) / 2 } }
1480 }
1481 #5
1482 }
1483 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1484 {
1485 \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1486 { #5 {#1} {#2} {#3} {#4} }
1487 { #5 {#1} {#3} {#2} {#4} }
1488 }
1489 </dvipdfmx | xetex>

```

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

`__draw_backend_box_use:Nnnnn`

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

```

1490 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1491 {
1492 \__kernel_backend_scope_begin:
1493 <*luatex | pdftex>
1494 \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1495 </luatex | pdftex>
1496 <*dvipdfmx | xetex>
1497 \__kernel_backend_literal:n
1498 { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1499 </dvipdfmx | xetex>
1500 \hbox_overlap_right:n { \box_use:N #1 }
1501 <*dvipdfmx | xetex>
1502 \__kernel_backend_literal:n { pdf:etrans }
1503 </dvipdfmx | xetex>
1504 \__kernel_backend_scope_end:
1505 }

```

(End definition for `__draw_backend_box_use:Nnnnn`.)

```

1506 </dvipdfmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1507 <*dvisvgm>

```

`__draw_backend_literal:n`

The same as the more general literal call.

`__draw_backend_literal:x`

```

1508 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1509 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for `__draw_backend_literal:n`.)

`__draw_backend_begin:` A drawing needs to be set up such that the co-ordinate system is translated. That is
`__draw_backend_end:` done inside a scope, which as described below

```

1510 \cs_new_protected:Npn \__draw_backend_begin:
1511 {
1512   \__kernel_backend_scope_begin:
1513   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1514 }
1515 \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
`__draw_backend_lineto:nn` values as they are given, the entire path needs to be collected up before being output
`__draw_backend_rectangle:nnnn` in one go. For that we use a dedicated storage routine, which adds spaces as required.
`__draw_backend_curveto:nnnnnn` Since paths should be fully expanded there is no need to worry about the internal x-type
`__draw_backend_add_to_path:n` expansion.
`\g__draw_draw_path_tl`

```

1516 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1517 {
1518   \__draw_backend_add_to_path:n
1519   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1520 }
1521 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1522 {
1523   \__draw_backend_add_to_path:n
1524   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1525 }
1526 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1527 {
1528   \__draw_backend_add_to_path:n
1529   {
1530     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1531     h ~ \dim_to_decimal:n {#3} ~
1532     v ~ \dim_to_decimal:n {#4} ~
1533     h ~ \dim_to_decimal:n { -#3 } ~
1534     Z
1535   }
1536 }
1537 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1538 {
1539   \__draw_backend_add_to_path:n
1540   {
1541     C ~
1542     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1543     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1544     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1545   }
1546 }
1547 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1548 {
1549   \tl_gset:Nx \g__draw_draw_path_tl
1550   {
1551     \g__draw_draw_path_tl
1552     \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1553     #1

```

```

1554     }
1555   }
1556   \tl_new:N \g__draw_draw_path_tl

(End definition for \__draw_backend_moveto:nn and others.)

```

__draw_backend_evenodd_rule: The fill rules here have to be handled as scopes.

```

\__draw_backend_nonzero_rule:
1557 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1558   { \__draw_backend_scope:n { fill-rule="evenodd" } }
1559 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1560   { \__draw_backend_scope:n { fill-rule="nonzero" } }

(End definition for \__draw_backend_evenodd_rule: and \__draw_backend_nonzero_rule:.)

```

__draw_backend_path:n Setting fill and stroke effects and doing clipping all has to be done using scopes. This 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.

```

\__draw_backend_closepath:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
1561 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_clip:
1562   { \__draw_backend_add_to_path:n { Z } }
\__draw_backend_discardpath:
1563 \cs_new_protected:Npn \__draw_backend_path:n #1
\g__draw_draw_clip_bool
1564   {
\g__draw_draw_path_int
1565     \bool_if:NTF \g__draw_draw_clip_bool
1566       {
1567         \int_gincr:N \g__draw_clip_path_int
1568         \__draw_backend_literal:x
1569           {
1570             < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1571               { ?nl }
1572             <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1573             < /clipPath > { ? nl }
1574             <
1575               use~xlink:href =
1576                 "\c_hash_str l3path \int_use:N \g__draw_path_int " ~
1577                 #1
1578             />
1579           }
1580         \__draw_backend_scope:x
1581           {
1582             clip-path =
1583               "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int )"
1584           }
1585         }
1586       {
1587         \__draw_backend_literal:x
1588           { <path ~ d=" \g__draw_draw_path_tl " ~ #1 /> }
1589         }
1590       \tl_gclear:N \g__draw_draw_path_tl
1591       \bool_gset_false:N \g__draw_draw_clip_bool
1592     }
1593   \int_new:N \g__draw_path_int
1594   \cs_new_protected:Npn \__draw_backend_stroke:
1595     { \__draw_backend_path:n { style="fill:none" } }

```

```

1596 \cs_new_protected:Npn \__draw_backend_closestroke:
1597 {
1598   \__draw_backend_closepath:
1599   \__draw_backend_stroke:
1600 }
1601 \cs_new_protected:Npn \__draw_backend_fill:
1602 { \__draw_backend_path:n { style="stroke:none" } }
1603 \cs_new_protected:Npn \__draw_backend_fillstroke:
1604 { \__draw_backend_path:n { } }
1605 \cs_new_protected:Npn \__draw_backend_clip:
1606 { \bool_gset_true:N \g__draw_draw_clip_bool }
1607 \bool_new:N \g__draw_draw_clip_bool
1608 \cs_new_protected:Npn \__draw_backend_discardpath:
1609 {
1610   \bool_if:NT \g__draw_draw_clip_bool
1611   {
1612     \int_gincr:N \g__draw_clip_path_int
1613     \__draw_backend_literal:x
1614     {
1615       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1616       { ?nl }
1617       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1618       < /clipPath >
1619     }
1620     \__draw_backend_scope:x
1621     {
1622       clip-path =
1623       "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1624     }
1625   }
1626   \tl_gclear:N \g__draw_draw_path_tl
1627   \bool_gset_false:N \g__draw_draw_clip_bool
1628 }

```

(End definition for __draw_backend_path:n and others.)

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_dash_aux:nn
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1629 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1630 {
1631   \use:x
1632   {
1633     \__draw_backend_dash_aux:nn
1634     { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1635     { \dim_to_decimal:n {#2} }
1636   }
1637 }
1638 \cs_new:Npn \__draw_backend_dash:n #1
1639 { , \dim_to_decimal_in_bp:n {#1} }
1640 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1641 {
1642   \__draw_backend_scope:x
1643   {
1644     stroke-dasharray =

```

```

1645         "
1646         \tl_if_empty:oTF { \use_none:n #1 }
1647         { none }
1648         { \use_none:n #1 }
1649     " ~
1650     stroke-offset=" #2 "
1651 }
1652 }
1653 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1654 { \__draw_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1655 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1656 { \__draw_backend_scope:x { stroke-miterlimit=" #1 " } }
1657 \cs_new_protected:Npn \__draw_backend_cap_but:
1658 { \__draw_backend_scope:n { stroke-linecap="butt" } }
1659 \cs_new_protected:Npn \__draw_backend_cap_round:
1660 { \__draw_backend_scope:n { stroke-linecap="round" } }
1661 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1662 { \__draw_backend_scope:n { stroke-linecap="square" } }
1663 \cs_new_protected:Npn \__draw_backend_join_miter:
1664 { \__draw_backend_scope:n { stroke-linejoin="miter" } }
1665 \cs_new_protected:Npn \__draw_backend_join_round:
1666 { \__draw_backend_scope:n { stroke-linejoin="round" } }
1667 \cs_new_protected:Npn \__draw_backend_join_bevel:
1668 { \__draw_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.

```

1669 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1670 {
1671     \__draw_backend_scope:n
1672     {
1673         transform =
1674         " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1675     }
1676 }

```

(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.

```

1677 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1678 {
1679     \__kernel_backend_scope_begin:
1680     \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1681     \__kernel_backend_literal_svg:n
1682     {
1683         < g~
1684         stroke="none"~
1685         transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1686         >
1687     }
1688     \box_set_wd:Nn #1 { Opt }

```

```

1689 \box_set_ht:Nn #1 { Opt }
1690 \box_set_dp:Nn #1 { Opt }
1691 \box_use:N #1
1692 \__kernel_backend_literal_svg:n { </g> }
1693 \__kernel_backend_scope_end:
1694 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```
1695 </dvisvgm>
```

```
1696 </package>
```

5 l3backend-graphics Implementation

```

1697 <*package>
1698 <@@=graphics>

```

5.1 dvips backend

```
1699 <*dvips>
```

_graphics_backend_getbb_eps:n Simply use the generic function.

```
1700 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n
```

(End definition for __graphics_backend_getbb_eps:n.)

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

```

1701 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1702 {
1703   \__kernel_backend_literal:x
1704   {
1705     PSfile = #1 \c_space_tl
1706     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1707     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1708     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1709     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1710   }
1711 }

```

(End definition for __graphics_backend_include_eps:n.)

```
1712 </dvips>
```

5.2 LuaTeX and pdfTeX backends

```
1713 <*luatex | pdftex>
```

\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.

```
1714 \tl_new:N \l_graphics_graphics_attr_tl
```

(End definition for \l_graphics_graphics_attr_tl.)

`_graphics_backend_getbb_jpg:n`
`_graphics_backend_getbb_pdf:n`
`_graphics_backend_getbb_png:n`
`_graphics_backend_getbb_auxi:n`
`_graphics_backend_getbb_auxii:n`

Getting the bounding box here requires us to box up the graphic and measure it. To deal with the difference in feature support in bitmap and vector graphics but keeping the common parts, there is a little work to do in terms of auxiliaries. The key here is to notice that we need two forms of the attributes: a “short” set to allow us to track for caching, and the full form to pass to the primitive.

```

1715 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1716 {
1717   \int_zero:N \l_graphics_page_int
1718   \tl_clear:N \l_graphics_pagebox_tl
1719   \tl_set:Nx \l__graphics_graphics_attr_tl
1720   {
1721     \tl_if_empty:NF \l_graphics_decodearray_tl
1722     { :D \l_graphics_decodearray_tl }
1723     \bool_if:NT \l_graphics_interpolate_bool
1724     { :I }
1725   }
1726   \tl_clear:N \l__graphics_graphics_attr_tl
1727   \_graphics_backend_getbb_auxi:n {#1}
1728 }
1729 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1730 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1731 {
1732   \tl_clear:N \l_graphics_decodearray_tl
1733   \bool_set_false:N \l_graphics_interpolate_bool
1734   \tl_set:Nx \l__graphics_graphics_attr_tl
1735   {
1736     : \l_graphics_pagebox_tl
1737     \int_compare:nNnT \l_graphics_page_int > 1
1738     { :P \int_use:N \l_graphics_page_int }
1739   }
1740   \_graphics_backend_getbb_auxi:n {#1}
1741 }
1742 \cs_new_protected:Npn \_graphics_backend_getbb_auxi:n #1
1743 {
1744   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1745   { \_graphics_backend_getbb_auxii:n {#1} }
1746 }

```

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

```

1747 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:n #1
1748 {
1749   \tex_immediate:D \tex_pdfximage:D
1750   \bool_lazy_or:nnT
1751   { \l_graphics_interpolate_bool }
1752   { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1753   {
1754     attr ~
1755     {
1756       \tl_if_empty:NF \l_graphics_decodearray_tl
1757       { /Decode~[ \l_graphics_decodearray_tl ] }
1758       \bool_if:NT \l_graphics_interpolate_bool
1759       { /Interpolate~true }

```

```

1760     }
1761   }
1762   \int_compare:nNnT \l_graphics_page_int > 0
1763     { page ~ \int_use:N \l_graphics_page_int }
1764   \tl_if_empty:NF \l_graphics_pagebox_tl
1765     { \l_graphics_pagebox_tl }
1766   {#1}
1767   \hbox_set:Nn \l__graphics_internal_box
1768     { \tex_pdfrefximage:D \tex_pdflastximage:D }
1769   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1770   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1771   \int_const:cn { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1772     { \tex_the:D \tex_pdflastximage:D }
1773   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1774 }

```

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

`__graphics_backend_include_jpg: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.

```

1775 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1776 {
1777   \tex_pdfrefximage:D
1778   \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1779 }
1780 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1781 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

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

`__graphics_backend_getbb_eps:n`
`__graphics_backend_getbb_eps:nm`
`__graphics_backend_include_eps:n`

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

```

\l__graphics_backend_dir_str 1782 \sys_if_shell:T
\l__graphics_backend_name_str 1783 {
\l__graphics_backend_ext_str 1784   \str_new:N \l__graphics_backend_dir_str
1785   \str_new:N \l__graphics_backend_name_str
1786   \str_new:N \l__graphics_backend_ext_str
1787   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1788   {
1789     \file_parse_full_name:nNNN {#1}
1790     \l__graphics_backend_dir_str
1791     \l__graphics_backend_name_str
1792     \l__graphics_backend_ext_str
1793     \exp_args:Nx \__graphics_backend_getbb_eps:nn
1794     {
1795       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1796       -converted-to.pdf
1797     }
1798     {#1}
1799   }
1800   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2

```

```

1801 {
1802   \file_compare_timestamp:nNnT {#2} > {#1}
1803   {
1804     \sys_shell_now:n
1805     { repstopdf ~ #2 ~ #1 }
1806   }
1807   \tl_set:Nn \l_graphics_name_tl {#1}
1808   \__graphics_backend_getbb_pdf:n {#1}
1809 }
1810 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1811 {
1812   \file_parse_full_name:nNNN {#1}
1813   \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1814   \exp_args:Nx \__graphics_backend_include_pdf:n
1815   {
1816     \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1817     -converted-to.pdf
1818   }
1819 }
1820 }

```

(End definition for __graphics_backend_getbb_eps:n and others.)

1821 </luatex | pdftex>

5.3 dvipdfmx backend

1822 <*dvipdfmx | xetex>

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

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
1823 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n
1824 <*dvipdfmx>
1825 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1826 {
1827   \int_zero:N \l_graphics_page_int
1828   \tl_clear:N \l_graphics_pagebox_tl
1829   \graphics_extract_bb:n {#1}
1830 }
1831 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1832 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1833 {
1834   \tl_clear:N \l_graphics_decodearray_tl
1835   \bool_set_false:N \l_graphics_interpolate_bool
1836   \graphics_extract_bb:n {#1}
1837 }
1838 </dvipdfmx>

```

(End definition for __graphics_backend_getbb_eps:n and others.)

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

1839 \int_new:N \g__graphics_track_int

(End definition for \g__graphics_track_int.)

`_graphics_backend_include_eps:n`
`_graphics_backend_include_jpg:n`
`_graphics_backend_include_pdf:n`
`_graphics_backend_include_png:n`
`_graphics_backend_include_auxi:nn`
`_graphics_backend_include_auxii:nnn`
`_graphics_backend_include_auxiii:nnn`

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

```

1840 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1841 {
1842   \__kernel_backend_literal:x
1843   {
1844     PSfile = #1 \c_space_tl
1845     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1846     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1847     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1848     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1849   }
1850 }
1851 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1852 { \_graphics_backend_include_auxi:nn {#1} { image } }
1853 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1854 <dvipdfmx>
1855 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1856 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1857 </dvipdfmx>

```

Graphic inclusion is set up to use the fact that each image is stored in the PDF as an XObject. This means that we can include repeated images only once and refer to them. To allow that, track the nature of each image: much the same as for the direct PDF mode case.

```

1858 \cs_new_protected:Npn \_graphics_backend_include_auxi:nn #1#2
1859 {
1860   \_graphics_backend_include_auxii:xnn
1861   {
1862     \tl_if_empty:NF \l_graphics_pagebox_tl
1863     { : \l_graphics_pagebox_tl }
1864     \int_compare:nNnT \l_graphics_page_int > 1
1865     { :P \int_use:N \l_graphics_page_int }
1866     \tl_if_empty:NF \l_graphics_decodearray_tl
1867     { :D \l_graphics_decodearray_tl }
1868     \bool_if:NT \l_graphics_interpolate_bool
1869     { :I }
1870   }
1871   {#1} {#2}
1872 }
1873 \cs_new_protected:Npn \_graphics_backend_include_auxii:nnn #1#2#3
1874 {
1875   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1876   {
1877     \__kernel_backend_literal:x
1878     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1879   }
1880   { \_graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1881 }
1882 \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!

```

1883 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1884 {
1885   \int_gincr:N \g__graphics_track_int
1886   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1887   \__kernel_backend_literal:x
1888   {
1889     pdf:#3~
1890     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1891     \int_compare:nNnT \l_graphics_page_int > 1
1892     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1893     \tl_if_empty:NF \l_graphics_pagebox_tl
1894     {
1895       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1896       bbox ~
1897         \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1898         \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1899         \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1900         \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1901     }
1902     (#1)
1903     \bool_lazy_or:nnT
1904     { \l_graphics_interpolate_bool }
1905     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1906     {
1907       <<
1908       \tl_if_empty:NF \l_graphics_decodearray_tl
1909       { /Decode~[ \l_graphics_decodearray_tl ] }
1910       \bool_if:NT \l_graphics_interpolate_bool
1911       { /Interpolate~true> }
1912     }
1913     >>
1914   }
1915 }

```

(End definition for __graphics_backend_include_eps:n and others.)

```

1916 </dvipdfmx | xetex>

```

5.4 Xe_{La}TeX backend

```

1917 <*xetex>

```

5.4.1 Images

For Xe_{La}TeX, 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 Xe_{La}TeX primitive omits the text box from the page box specification, so there is also some “trimming” to do here.

```

1918 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1919 {
1920   \int_zero:N \l_graphics_page_int
1921   \tl_clear:N \l_graphics_pagebox_tl
1922   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1923 }

```

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
\__graphics_backend_getbb_auxi:nN
\__graphics_backend_getbb_auxii:nnN
\__graphics_backend_getbb_auxiii:nNnN
\__graphics_backend_getbb_auxiv:nnNnN
\__graphics_backend_getbb_auxiv:NnNnN
\__graphics_backend_getbb_auxv:nNnN
\__graphics_backend_getbb_auxv:NnNnN
\__graphics_backend_getbb_pagebox:w

```

```

1924 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1925 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1926 {
1927   \tl_clear:N \l_graphics_decodearray_tl
1928   \bool_set_false:N \l_graphics_interpolate_bool
1929   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1930 }
1931 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
1932 {
1933   \int_compare:nNnTF \l_graphics_page_int > 1
1934     { \__graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
1935     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1936 }
1937 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
1938 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1939 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
1940 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1941 {
1942   \tl_if_empty:NTF \l_graphics_pagebox_tl
1943     { \__graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1944     { \__graphics_backend_getbb_auxv:nNnn {#1} #2 {#3} {#4} }
1945 }
1946 }
1947 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
1948 {
1949   \use:x
1950   {
1951     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1952     { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1953   }
1954 }
1955 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1956 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1957 {
1958   \graphics_bb_restore:nF {#1#3}
1959   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1960 }
1961 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1962 {
1963   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1964   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1965   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1966   \graphics_bb_save:n {#1#3}
1967 }
1968 \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 Xe_{La}TeX is best done using
__graphics_backend_include_bitmap_quote:w 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.

```

1969 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1970 {

```

```

1971 \tex_XeTeXpdffile:D
1972 \__graphics_backend_include_pdf_quote:w #1 "#1" \s__graphics_stop \c_space_tl
1973 \int_compare:nNnT \l_graphics_page_int > 0
1974 { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1975 \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1976 }
1977 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \s__graphics_stop
1978 { " #2 " }

```

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

```

1979 </xetex>

```

5.5 dvisvgm backend

```

1980 <*dvisvgm>

```

_graphics_backend_getbb_eps:n Simply use the generic function.

```

1981 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n

```

(End definition for __graphics_backend_getbb_eps:n.)

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

```

1982 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1983 {
1984   \int_zero:N \l_graphics_page_int
1985   \tl_clear:N \l_graphics_pagebox_tl
1986   \graphics_extract_bb:n {#1}
1987 }
1988 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n

```

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

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

```

1989 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1990 {
1991   \tl_clear:N \l_graphics_decodearray_tl
1992   \bool_set_false:N \l_graphics_interpolate_bool
1993   \graphics_extract_bb:n {#1}
1994 }

```

(End definition for __graphics_backend_getbb_pdf:n.)

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

```

1995 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1996 { __graphics_backend_include:nn { PSfile } {#1} }
1997 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1998 { __graphics_backend_include:nn { pdffile } {#1} }
1999 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
2000 {
2001   \__kernel_backend_literal:x
2002   {
2003     #1 = #2 \c_space_tl
2004     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl

```

```

2005         lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2006         urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2007         ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
2008     }
2009 }

```

(End definition for `__graphics_backend_include_eps:n`, `__graphics_backend_include_pdf:n`, and `__graphics_backend_include:nn`.)

```

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

```

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

```

2010 \cs_new_protected:Npn \__graphics_backend_include_png:n #1
2011 {
2012     \__kernel_backend_literal:x
2013     {
2014         dvisvgm:img~
2015         \dim_to_decimal:n { \l_graphics_ury_dim } ~
2016         \dim_to_decimal:n { \l_graphics_ury_dim } ~
2017         \__graphics_backend_include_bitmap_quote:w #1 " #1 " \s__graphics_stop
2018     }
2019 }
2020 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_png:n
2021 \cs_new:Npn \__graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \s__graphics_stop
2022 { " #2 " }

```

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

```

2023 </dvisvgm>
2024 </package>

```

6 l3backend-pdf Implementation

```

2025 <*package>
2026 <@@=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
2027 \box_new:N \l__pdf_internal_box
(End definition for \l__pdf_internal_box.)

```

6.2 dvips backend

2028 `<*dvips>`

`__pdf_backend_pdfmark:n`
`__pdf_backend_pdfmark:x`

Used often enough it should be a separate function.

2029 `\cs_new_protected:Npn __pdf_backend_pdfmark:n #1`
 2030 `{ __kernel_backend_postscript:n { mark #1 ~ pdfmark } }`
 2031 `\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`

2032 `\cs_new_protected:Npn __pdf_backend_catalog_gput:nn #1#2`
 2033 `{ __pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }`
 2034 `\cs_new_protected:Npn __pdf_backend_info_gput:nn #1#2`
 2035 `{ __pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }`

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

6.2.2 Objects

`\g__pdf_backend_object_int`
`\g__pdf_backend_object_prop`

For tracking objects to allow finalisation.

2036 `\int_new:N \g__pdf_backend_object_int`
 2037 `\prop_new:N \g__pdf_backend_object_prop`

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

`__pdf_backend_object_new:nn`
`__pdf_backend_object_ref:n`

Tracking objects is similar to `dvipdfmx`.

2038 `\cs_new_protected:Npn __pdf_backend_object_new:nn #1#2`
 2039 `{`
 2040 `\int_gincr:N \g__pdf_backend_object_int`
 2041 `\int_const:cn`
 2042 `{ c__pdf_backend_object_ \tl_to_str:n {#1} _int }`
 2043 `{ \g__pdf_backend_object_int }`
 2044 `\prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}`
 2045 `}`
 2046 `\cs_new:Npn __pdf_backend_object_ref:n #1`
 2047 `{ { pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } } }`

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

`_pdf_backend_object_write:nn`
`_pdf_backend_object_write:nx`
`__pdf_backend_object_write_array:nn`
`_pdf_backend_object_write_dict:nn`
`__pdf_backend_object_write_fstream:nn`
`_pdf_backend_object_write_stream:nn`
`_pdf_backend_object_write_stream:nnn`

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

2048 `\cs_new_protected:Npn __pdf_backend_object_write:nn #1#2`
 2049 `{`
 2050 `__pdf_backend_pdfmark:x`
 2051 `{`
 2052 `/_objdef ~ __pdf_backend_object_ref:n {#1}`
 2053 `/type`
 2054 `\str_case_e:nn`
 2055 `{ \prop_item:Nn \g__pdf_backend_object_prop {#1} }`
 2056 `{`
 2057 `{ array } { /array }`
 2058 `{ dict } { /dict }`

```

2059         { fstream } { /stream }
2060         { stream } { /stream }
2061     }
2062     /OBJ
2063 }
2064 \use:c
2065 { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2066 { __pdf_backend_object_ref:n {#1} } {#2}
2067 }
2068 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2069 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2070 {
2071     \__pdf_backend_pdfmark:x
2072     { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2073 }
2074 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2075 {
2076     \__pdf_backend_pdfmark:x
2077     { #1 << \exp_not:n {#2} >> /PUT }
2078 }
2079 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2080 {
2081     \exp_args:Nx
2082     \__pdf_backend_object_write_fstream:nnn {#1} #2
2083 }
2084 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2085 {
2086     \__kernel_backend_postscript:n
2087     {
2088         SDict ~ begin ~
2089         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2090         mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2091         end
2092     }
2093 }
2094 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2095 {
2096     \exp_args:Nx
2097     \__pdf_backend_object_write_stream:nnn {#1} #2
2098 }
2099 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2100 {
2101     \__kernel_backend_postscript:n
2102     {
2103         mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2104         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2105     }
2106 }

```

(End definition for __pdf_backend_object_write:nn and others.)

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

```

\__pdf_backend_object_now:nx
2107 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2108 {

```

```

2109 \int_gincr:N \g__pdf_backend_object_int
2110 \__pdf_backend_pdfmark:x
2111 {
2112   /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2113   /type
2114   \str_case:nn
2115     {#1}
2116     {
2117       { array } { /array }
2118       { dict } { /dict }
2119       { fstream } { /stream }
2120       { stream } { /stream }
2121     }
2122   /OBJ
2123 }
2124 \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2125 { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2126 }
2127 \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.

```

2128 \cs_new:Npn \__pdf_backend_object_last:
2129   { { 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.

```

2130 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2131   { { 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.

```

2132 \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.

```

2133 \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.

```

2134 \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.

```
2135 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2136 {
2137   \exp_args:Nf \_pdf_backend_annotation_aux:nnnn
2138   { \dim_eval:n {#1} } {#2} {#3} {#4}
2139 }
2140 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2141 {
2142   \box_move_down:nn {#3}
2143   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2144   \box_move_up:nn {#2}
2145   {
2146     \hbox:n
2147     {
2148       \_kernel_kern:n {#1}
2149       \_kernel_backend_postscript:n { pdf.save.ur }
2150       \_kernel_kern:n { -#1 }
2151     }
2152   }
2153   \int_gincr:N \g__pdf_backend_object_int
2154   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2155   \_pdf_backend_pdfmark:x
2156   {
2157     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2158     pdf.rect
2159     #4 ~
2160     /ANN
2161   }
2162 }
```

(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.

```
2163 \cs_new:Npn \_pdf_backend_annotation_last:
2164 { { 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.

```
2165 \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.

```
2166 \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.

```
2167 \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.

2168 `\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.

2169 `\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.

2170 `\tl_new:N \l__pdf_breaklink_pdfmark_tl`

2171 `\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.

2172 `\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.

2173 `\cs_new_eq:NN __pdf_breaklink_usebox:N \box_use:N`

(End definition for `__pdf_breaklink_usebox:N`.)

`__pdf_backend_link_begin_goto:nnw` 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`.

`__pdf_backend_link_begin_user:nnw`

`__pdf_backend_link:nw` 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).

`__pdf_backend_link_aux:nw`

`__pdf_backend_link_end:`

`__pdf_backend_link_end_aux:`

`__pdf_backend_link_minima:` 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.

`__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`

2174 `\cs_new_protected:Npn __pdf_backend_link_begin_goto:nnw #1#2`

2175 `{`

2176 `__pdf_backend_link_begin:nw`

2177 `{ #1 /Subtype /Link /Action << /S /GoTo /D (#2) >> }`

2178 `}`

2179 `\cs_new_protected:Npn __pdf_backend_link_begin_user:nnw #1#2`

2180 `{ __pdf_backend_link_begin:nw {#1#2} }`

2181 `\cs_new_protected:Npn __pdf_backend_link_begin:nw #1`

2182 `{`

```

2183 \bool_if:NF \g__pdf_backend_link_bool
2184 { \__pdf_backend_link_begin_aux:nw {#1} }
2185 }

```

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.

```

2186 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2187 {
2188   \bool_gset_true:N \g__pdf_backend_link_bool
2189   \__kernel_backend_postscript:n
2190   { /pdf.link.dict ( #1 ) def }
2191   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2192   \__pdf_backend_link_sf_save:
2193   \mode_if_math:TF
2194     { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2195     { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2196   \hbox_set:Nw \l__pdf_backend_content_box
2197   \__pdf_backend_link_sf_restore:
2198   \bool_if:NT \g__pdf_backend_link_math_bool
2199     { \c_math_toggle_token }
2200 }
2201 \cs_new_protected:Npn \__pdf_backend_link_end:
2202 {
2203   \bool_if:NT \g__pdf_backend_link_bool
2204     { \__pdf_backend_link_end_aux: }
2205 }
2206 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2207 {
2208   \bool_if:NT \g__pdf_backend_link_math_bool
2209     { \c_math_toggle_token }
2210   \__pdf_backend_link_sf_save:
2211   \hbox_set_end:
2212   \__pdf_backend_link_minima:
2213   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2214   \exp_args:Nx \__pdf_backend_link_outerbox:n
2215   {
2216     \int_if_odd:nTF { \value { page } }
2217       { \oddsidemargin }
2218       { \evensidemargin }
2219   }
2220   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2221   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2222   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2223   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2224   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2225   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2226   {
2227     \hbox:n
2228       { \__kernel_backend_postscript:n { pdf.save.linkur } }
2229   }
2230   \int_gincr:N \g__pdf_backend_object_int
2231   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2232   \__kernel_backend_postscript:x
2233   {

```

```

2234     mark
2235     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2236     \g__pdf_backend_link_dict_tl \c_space_tl
2237     pdf.rect
2238     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2239   }
2240   \__pdf_backend_link_sf_restore:
2241   \bool_gset_false:N \g__pdf_backend_link_bool
2242 }
2243 \cs_new_protected:Npn \__pdf_backend_link_minima:
2244 {
2245   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2246   \__kernel_backend_postscript:x
2247   {
2248     /pdf.linkdp.pad ~
2249     \dim_to_decimal:n
2250     {
2251       \dim_max:nn
2252       {
2253         \box_dp:N \l__pdf_backend_model_box
2254         - \box_dp:N \l__pdf_backend_content_box
2255       }
2256       { Opt }
2257     } ~
2258     pdf.pt.dvi ~ def
2259   /pdf.linkht.pad ~
2260   \dim_to_decimal:n
2261   {
2262     \dim_max:nn
2263     {
2264       \box_ht:N \l__pdf_backend_model_box
2265       - \box_ht:N \l__pdf_backend_content_box
2266     }
2267     { Opt }
2268   } ~
2269   pdf.pt.dvi ~ def
2270 }
2271 }
2272 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2273 {
2274   \__kernel_backend_postscript:x
2275   {
2276     /pdf.outerbox
2277     [
2278       \dim_to_decimal:n {#1} ~
2279       \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2280       \dim_to_decimal:n { #1 + \textwidth } ~
2281       \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2282     ]
2283     [ exch { pdf.pt.dvi } forall ] def
2284   /pdf.baselineskip ~
2285   \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2286   { pdf.pt.dvi ~ def }
2287   { pop ~ pop }

```

```

2288         ifelse
2289     }
2290 }
2291 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2292 {
2293     \int_gset:Nn \g__pdf_backend_link_sf_int
2294     {
2295         \mode_if_horizontal:TF
2296         { \tex_spacefactor:D }
2297         { 0 }
2298     }
2299 }
2300 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2301 {
2302     \mode_if_horizontal:T
2303     {
2304         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2305         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2306     }
2307 }

```

(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.

```

2308 \use_none:n
2309 {
2310     \cs_if_exist:NT \@makecol@hook
2311     {
2312         \tl_put_right:Nn \@makecol@hook
2313         {
2314             \box_if_empty:NF \@cclv
2315             {
2316                 \vbox_set:Nn \@cclv
2317                 {
2318                     \__kernel_backend_postscript:n
2319                     {
2320                         pdf.globaldict /pdf.brokenlink.rect ~ known
2321                         { pdf.bordertracking.continue }
2322                         if
2323                     }
2324                     \vbox_unpack_drop:N \@cclv
2325                     \__kernel_backend_postscript:n
2326                     { pdf.bordertracking.endpage }
2327                 }
2328             }
2329         }
2330         \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2331         \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2332         \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2333     }
2334 }

```

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

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

```
2335 \cs_new:Npn \_pdf_backend_link_last:
2336 { { 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.

```
2337 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2338 {
2339   \__kernel_backend_postscript:x
2340   {
2341     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2342   }
2343 }
```

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

```
2344 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2345 {
2346   \__kernel_backend_postscript:n { pdf.dest.anchor }
2347   \_pdf_backend_pdfmark:x
2348   {
2349     /View
2350     [
2351       \str_case:nnF {#2}
2352       {
2353         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2354         { fit } { /Fit }
2355         { fitb } { /FitB }
2356         { fitbh } { /FitBH ~ pdf.dest.y }
2357         { fitbv } { /FitBV ~ pdf.dest.x }
2358         { fith } { /FitH ~ pdf.dest.y }
2359         { fitv } { /FitV ~ pdf.dest.x }
2360         { fitr } { /Fit }
2361       }
2362       {
2363         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2364       }
2365     ]
2366     /Dest ( \exp_not:n {#1} ) cvn
2367     /DEST
2368   }
2369 }
2370 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2371 {
2372   \exp_args:Ne \_pdf_backend_destination_aux:nnnn
2373   { \dim_eval:n {#2} } {#1} {#3} {#4}
2374 }
```

```

2375 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2376 {
2377   \vbox_to_zero:n
2378   {
2379     \__kernel_kern:n {#4}
2380     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2381     \tex_vss:D
2382   }
2383   \__kernel_kern:n {#1}
2384   \vbox_to_zero:n
2385   {
2386     \__kernel_kern:n { -#3 }
2387     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2388     \tex_vss:D
2389   }
2390   \__kernel_kern:n { -#1 }
2391   \__pdf_backend_pdfmark:n
2392   {
2393     /View
2394     [
2395       /FitR ~
2396       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2397       pdf.urx ~ pdf.ury ~ pdf.dest2device
2398     ]
2399     /Dest ( #2 ) cvn
2400     /DEST
2401   }
2402 }

```

(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
2403 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2404 {
2405   \int_compare:nNnT {#1} = 0
2406   {
2407     \__kernel_backend_literal_postscript:n
2408     {
2409       /setdistillerparams ~ where
2410       { pop << /CompressPages ~ false >> setdistillerparams }
2411       if
2412     }
2413   }
2414 }
2415 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2416 {
2417   \bool_if:nF {#1}
2418   {
2419     \__kernel_backend_literal_postscript:n
2420     {
2421       /setdistillerparams ~ where

```

```

2422         { pop << /CompressStreams ~ false >> setdistillerparams }
2423     if
2424     }
2425 }
2426 }

```

(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`

```

2427 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2428 {
2429     \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2430 }
2431 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2432 {
2433     \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2434 }

```

(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:`

```

2435 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2436 \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`

`_pdf_backend_emc:`

Simple wrappers.

```

2437 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2438 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2439 \cs_new_protected:Npn \_pdf_backend_emc:
2440 { \_pdf_backend_pdfmark:n { /EMC } }

```

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

2441 \langle /dvips \rangle

6.3 LuaTeX and pdfTeX backend

2442 \langle *luatex | pdftex \rangle

6.3.1 Annotations

`_pdf_backend_annotation:nnnn`

Simply pass the raw data through, just dealing with evaluation of dimensions.

```

2443 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2444 {
2445      $\langle$ *luatex $\rangle$ 
2446     \tex_pdfextension:D annot ~
2447      $\langle$ /luatex $\rangle$ 
2448      $\langle$ *pdftex $\rangle$ 
2449     \tex_pdfannot:D
2450      $\langle$ /pdftex $\rangle$ 
2451     width ~ \dim_eval:n {#1} ~
2452     height ~ \dim_eval:n {#2} ~
2453     depth ~ \dim_eval:n {#3} ~

```



```

2454     {#4}
2455 }

```

(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.

```

2456 \cs_new:Npx \_pdf_backend_annotation_last:
2457 {
2458     \exp_not:N \int_value:w
2459 <*luatex>
2460     \exp_not:N \tex_pdffeedback:D lastannot ~
2461 </luatex>
2462 <*pdftex>
2463     \exp_not:N \tex_pdflastannot:D
2464 </pdftex>
2465     \c_space_tl 0 ~ R
2466 }

```

(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` 2467 `\cs_new_protected:Npn _pdf_backend_link_begin_goto:nnw #1#2`

`_pdf_backend_link_begin:nnnw` 2468 `{ _pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }`

`_pdf_backend_link_end:` 2469 `\cs_new_protected:Npn _pdf_backend_link_begin_user:nnw #1#2`

2470 `{ _pdf_backend_link_begin:nnnw {#1} { user } {#2} }`

2471 `\cs_new_protected:Npn _pdf_backend_link_begin:nnnw #1#2#3`

2472 `{`

2473 `<*luatex>`

2474 `\tex_pdfextension:D startlink ~`

2475 `</luatex>`

2476 `<*pdftex>`

2477 `\tex_pdfstartlink:D`

2478 `</pdftex>`

2479 `attr {#1}`

2480 `#2 {#3}`

2481 `}`

2482 `\cs_new_protected:Npn _pdf_backend_link_end:`

2483 `{`

2484 `<*luatex>`

2485 `\tex_pdfextension:D endlink \scan_stop:`

2486 `</luatex>`

2487 `<*pdftex>`

2488 `\tex_pdfendlink:D`

2489 `</pdftex>`

2490 `}`

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

`_pdf_backend_link_last:` Formatted for direct use.

```

2491 \cs_new:Npx \_pdf_backend_link_last:
2492 {
2493     \exp_not:N \int_value:w
2494 <*luatex>

```

```

2495 \exp_not:N \tex_pdffeedback:D lastlink ~
2496 </luatex>
2497 <*pdftex>
2498 \exp_not:N \tex_pdflastlink:D
2499 </pdftex>
2500 \c_space_tl 0 ~ R
2501 }

```

(End definition for _pdf_backend_link_last:.)

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

```

2502 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2503 {
2504 <*luatex>
2505 \tex_pdfvariable:D linkmargin
2506 </luatex>
2507 <*pdftex>
2508 \tex_pdflinkmargin:D
2509 </pdftex>
2510 \dim_eval:n {#1} \scan_stop:
2511 }

```

(End definition for _pdf_backend_link_margin:n.)

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

_pdf_backend_destination:nnnn

```

2512 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2513 {
2514 <*luatex>
2515 \tex_pdfextension:D dest ~
2516 </luatex>
2517 <*pdftex>
2518 \tex_pdfdest:D
2519 </pdftex>
2520 name {#1}
2521 \str_case:nnF {#2}
2522 {
2523 { xyz } { xyz }
2524 { fit } { fit }
2525 { fitb } { fitb }
2526 { fitbh } { fitbh }
2527 { fitbv } { fitbv }
2528 { fith } { fith }
2529 { fitv } { fitv }
2530 { fitr } { fitr }
2531 }
2532 { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2533 \scan_stop:
2534 }
2535 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2536 {
2537 <*luatex>
2538 \tex_pdfextension:D dest ~

```

```

2539 </luatex>
2540 <*pdftex>
2541   \tex_pdfdest:D
2542 </pdftex>
2543   name {#1}
2544   fitr ~
2545   width \dim_eval:n {#2} ~
2546   height \dim_eval:n {#3} ~
2547   depth \dim_eval:n {#4} \scan_stop:
2548 }

```

(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
2549 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2550 {
2551 <*luatex>
2552   \tex_pdfextension:D catalog
2553 </luatex>
2554 <*pdftex>
2555   \tex_pdfcatalog:D
2556 </pdftex>
2557   { / #1 ~ #2 }
2558 }
2559 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2560 {
2561 <*luatex>
2562   \tex_pdfextension:D info
2563 </luatex>
2564 <*pdftex>
2565   \tex_pdfinfo:D
2566 </pdftex>
2567   { / #1 ~ #2 }
2568 }

```

(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.

```

2569 \prop_new:N \g_pdf_backend_object_prop

```

(End definition for \g_pdf_backend_object_prop.)

_pdf_backend_object_new:nn Declaring objects means reserving at the PDF level plus starting tracking.

_pdf_backend_object_ref:n

```

2570 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2
2571 {
2572 <*luatex>
2573   \tex_pdfextension:D obj ~
2574 </luatex>
2575 <*pdftex>
2576   \tex_pdfobj:D

```

```

2577 </pdfTeX>
2578     reserveobjnum ~
2579     \int_const:cn
2580     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2581 <*luatex>
2582     { \tex_pdffeedback:D lastobj }
2583 </luatex>
2584 <*pdfTeX>
2585     { \tex_pdflastobj:D }
2586 </pdfTeX>
2587     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2588 }
2589 \cs_new:Npn \__pdf_backend_object_ref:n #1
2590 { \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

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

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

```

\__pdf_backend_object_write:nn 2591 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
\__pdf_backend_object_write:nx 2592 {
\__pdf_exp_not_i:nn 2593 <*luatex>
\__pdf_exp_not_ii:nn 2594     \tex_immediate:D \tex_pdfextension:D obj ~
2595 </luatex>
2596 <*pdfTeX>
2597     \tex_immediate:D \tex_pdfobj:D
2598 </pdfTeX>
2599     useobjnum ~
2600     \int_use:c
2601     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2602     \str_case:e:nn
2603     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2604     {
2605         { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2606         { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2607         { fstream }
2608         {
2609             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2610             file ~ { \__pdf_exp_not_ii:nn #2 }
2611         }
2612         { stream }
2613         {
2614             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2615             { \__pdf_exp_not_ii:nn #2 }
2616         }
2617     }
2618 }
2619 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2620 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2621 \cs_new:Npn \__pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

```

(End definition for __pdf_backend_object_write:nn, __pdf_exp_not_i:nn, and __pdf_exp_not_ii:nn.)

__pdf_backend_object_now:nn Much like writing, but direct creation.

```

\__pdf_backend_object_now:nx 2622 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2

```

```

2623 {
2624 <*luatex>
2625   \tex_immediate:D \tex_pdfextension:D obj ~
2626 </luatex>
2627 <*pdftex>
2628   \tex_immediate:D \tex_pdfobj:D
2629 </pdftex>
2630   \str_case:nn
2631     {#1}
2632   {
2633     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2634     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2635     { fstream }
2636     {
2637       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2638       file ~ { \__pdf_exp_not_ii:nn #2 }
2639     }
2640     { stream }
2641     {
2642       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2643       { \__pdf_exp_not_ii:nn #2 }
2644     }
2645   }
2646 }
2647 \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.

```

2648 \cs_new:Npx \__pdf_backend_object_last:
2649 {
2650   \exp_not:N \int_value:w
2651 <*luatex>
2652   \exp_not:N \tex_pdffeedback:D lastobj ~
2653 </luatex>
2654 <*pdftex>
2655   \exp_not:N \tex_pdflastobj:D
2656 </pdftex>
2657   \c_space_tl 0 ~ R
2658 }

```

(End definition for __pdf_backend_object_last:.)

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

```

2659 \cs_new:Npx \__pdf_backend_pageobject_ref:n #1
2660 {
2661   \exp_not:N \int_value:w
2662 <*luatex>
2663   \exp_not:N \tex_pdffeedback:D pageref
2664 </luatex>
2665 <*pdftex>
2666   \exp_not:N \tex_pdfpageref:D
2667 </pdftex>
2668   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2669 }

```

(End definition for `_pdf_backend_pageobject_ref:n`.)

6.3.4 Structure

Simply pass data to the engine.

```

\_pdf_backend_compresslevel:n
\_pdf_backend_compress_objects:n
\_pdf_backend_objcompresslevel:n
2670 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2671 {
2672   \tex_global:D
2673   \*luatex
2674   \tex_pdfvariable:D compresslevel
2675   \*luatex
2676   \*pdftex
2677   \tex_pdfcompresslevel:D
2678   \*pdftex
2679   \int_value:w \int_eval:n {#1} \scan_stop:
2680 }
2681 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2682 {
2683   \bool_if:nTF {#1}
2684   { \_pdf_backend_objcompresslevel:n { 2 } }
2685   { \_pdf_backend_objcompresslevel:n { 0 } }
2686 }
2687 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2688 {
2689   \tex_global:D
2690   \*luatex
2691   \tex_pdfvariable:D objcompresslevel
2692   \*luatex
2693   \*pdftex
2694   \tex_pdfobjcompresslevel:D
2695   \*pdftex
2696   #1 \scan_stop:
2697 }

```

(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.

```

2698 \cs_new_protected:Npx \_pdf_backend_version_major_gset:n #1
2699 {
2700   \*luatex
2701   \int_compare:nNnT \tex_luatexversion:D > { 106 }
2702   {
2703     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2704     \exp_not:N \int_eval:n {#1} \scan_stop:
2705   }
2706   \*luatex
2707   \*pdftex
2708   \cs_if_exist:NT \tex_pdfmajorversion:D
2709   {
2710     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2711     \exp_not:N \int_eval:n {#1} \scan_stop:
2712   }
2713   \*pdftex

```

```

2714 }
2715 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2716 {
2717   \tex_global:D
2718   \*luatex
2719   \tex_pdfvariable:D minorversion
2720   \*luatex
2721   \*pdftex
2722   \tex_pdfminorversion:D
2723   \*pdftex
2724   \int_eval:n {#1} \scan_stop:
2725 }

```

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

__pdf_backend_version_major: As above.

```

\__pdf_backend_version_minor:
2726 \cs_new:Npx \__pdf_backend_version_major:
2727 {
2728   \*luatex
2729   \int_compare:nNnTF \tex luatexversion:D > { 106 }
2730   { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2731   { 1 }
2732   \*luatex
2733   \*pdftex
2734   \cs_if_exist:NTF \tex_pdfmajorversion:D
2735   { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2736   { 1 }
2737   \*pdftex
2738 }
2739 \cs_new:Npn \__pdf_backend_version_minor:
2740 {
2741   \tex_the:D
2742   \*luatex
2743   \tex_pdfvariable:D minorversion
2744   \*luatex
2745   \*pdftex
2746   \tex_pdfminorversion:D
2747   \*pdftex
2748 }

```

(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>.

```

2749 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2750 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2751 \cs_new_protected:Npn \__pdf_backend_emc:
2752 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2753 \*luatex | \*pdftex

```

6.4 dvipdfmx backend

2754 `*dvipdfmx|xetex`

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

`__pdf_backend:x`

```

2755 \cs_new_protected:Npx \__pdf_backend:n #1
2756 { \__kernel_backend_literal:n { pdf: #1 } }
2757 \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
2758 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2759 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2760 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2761 { \__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`

```

2762 \int_new:N \g__pdf_backend_object_int
2763 \prop_new:N \g__pdf_backend_object_prop

```

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

`__pdf_backend_object_new:nn` Objects are tracked at the macro level, but we don't have to do anything at this stage.

`__pdf_backend_object_ref:n`

```

2764 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2765 {
2766   \int_gincr:N \g__pdf_backend_object_int
2767   \int_const:cn
2768   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2769   { \g__pdf_backend_object_int }
2770   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2771 }
2772 \cs_new:Npn \__pdf_backend_object_ref:n #1
2773 { @pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } }

```

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

`__pdf_backend_object_write:nn` This is where we choose the actual type.

```

\__pdf_backend_object_write:nx
\__pdf_backend_object_write:nnn
\__pdf_backend_object_write_array:nn
\__pdf_backend_object_write_dict:nn
\__pdf_backend_object_write_fstream:nn
\__pdf_backend_object_write_stream:nn
\__pdf_backend_object_write_stream:nnnn
2774 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2775 {
2776   \exp_args:Nx \__pdf_backend_object_write:nnn
2777   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2778 }
2779 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2780 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2781 {
2782   \use:c { \__pdf_backend_object_write_ #1 :nn }
2783   { \__pdf_backend_object_ref:n {#2} } {#3}
2784 }

```



```

2785 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2786 {
2787   \__pdf_backend:x
2788   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2789 }
2790 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2791 {
2792   \__pdf_backend:x
2793   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2794 }
2795 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2796 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2797 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2798 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2799 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2800 {
2801   \__pdf_backend:x
2802   {
2803     #1 stream ~ #2 ~
2804     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2805   }
2806 }

```

(End definition for __pdf_backend_object_write:nn and others.)

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

```

\__pdf_backend_object_now:nx
2807 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2808 {
2809   \int_gincr:N \g__pdf_backend_object_int
2810   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2811   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2812   {#2}
2813 }
2814 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2815 \cs_new:Npn \__pdf_backend_object_last:
2816 { @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_TTeX.

```

2817 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2818 { @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.

```
2819 \int_new:N \g_pdf_backend_annotation_int
```

(End definition for `\g_pdf_backend_annotation_int`.)

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

```
2820 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2821 {
2822   \int_gincr:N \g_pdf_backend_object_int
2823   \int_gset_eq:NN \g_pdf_backend_annotation_int \g_pdf_backend_object_int
2824   \_pdf_backend:x
2825   {
2826     ann ~ @pdf.obj \int_use:N \g_pdf_backend_object_int \c_space_tl
2827     width ~ \dim_eval:n {#1} ~
2828     height ~ \dim_eval:n {#2} ~
2829     depth ~ \dim_eval:n {#3} ~
2830     << /Type /Annot #4 >>
2831   }
2832 }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

`_pdf_backend_annotation_last:`

```
2833 \cs_new:Npn \_pdf_backend_annotation_last:
2834 { @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.

```
2835 \int_new:N \g_pdf_backend_link_int
```

(End definition for `\g_pdf_backend_link_int`.)

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

`_pdf_backend_link_begin_user:nnw`

`_pdf_backend_link_begin:n`

`_pdf_backend_link_end:`

```
2836 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2837 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2838 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2839 { \_pdf_backend_link_begin:n {#1#2} }
2840 \cs_new_protected:Npx \_pdf_backend_link_begin:n #1
2841 {
2842   \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2843   {
2844     \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int
2845   }
2846   \_pdf_backend:x
2847   {
2848     bann ~
2849     \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2850     {
2851       @pdf.lnk
2852       \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int
2853       \c_space_tl
2854     }
2855   }
2856 }
```

```

2855         <<
2856         /Type /Annot
2857         #1
2858         >>
2859     }
2860 }
2861 \cs_new_protected:Npn \__pdf_backend_link_end:
2862 { \__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.

```

2863 \cs_new:Npx \__pdf_backend_link_last:
2864 {
2865     \int_compare:nNnF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
2866     {
2867         @pdf.lnk
2868         \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
2869     }
2870 }

```

(End definition for __pdf_backend_link_last:.)

__pdf_backend_link_margin:n Pass to dvipdfmx.

```

2871 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2872 { \__kernel_backend_literal:x { dvipdfmx:config~g~ \dim_eval:n {#1} } }

```

(End definition for __pdf_backend_link_margin:n.)

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

```

2873 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2874 {
2875     \__pdf_backend:x
2876     {
2877         dest ~ ( \exp_not:n {#1} )
2878         [
2879             @thispage
2880             \str_case:nnF {#2}
2881             {
2882                 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2883                 { fit } { /Fit }
2884                 { fitb } { /FitB }
2885                 { fitbh } { /FitBH }
2886                 { fitbv } { /FitBV ~ @xpos }
2887                 { fith } { /FitH ~ @ypos }
2888                 { fitv } { /FitV ~ @xpos }
2889                 { fitr } { /Fit }
2890             }
2891             { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2892         ]
2893     }

```

```

2894 }
2895 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2896 {
2897   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2898   { \dim_eval:n {#2} } {#1} {#3} {#4}
2899 }
2900 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2901 {
2902   \vbox_to_zero:n
2903   {
2904     \__kernel_kern:n {#4}
2905     \hbox:n
2906     {
2907       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2908       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2909     }
2910     \tex_vss:D
2911   }
2912   \__kernel_kern:n {#1}
2913   \vbox_to_zero:n
2914   {
2915     \__kernel_kern:n { -#3 }
2916     \hbox:n
2917     {
2918       \__pdf_backend:n
2919       {
2920         dest ~ (#2)
2921         [
2922           @thispage
2923           /FitR ~
2924           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2925           @xpos ~ @ypos
2926         ]
2927       }
2928     }
2929     \tex_vss:D
2930   }
2931   \__kernel_kern:n { -#1 }
2932 }

```

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

6.4.4 Structure

Pass data to the backend: these are a one-shot.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2933 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2934 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2935 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2936 {
2937   \bool_if:nF {#1}
2938   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2939 }

```

(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.

```

2940 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2941 {
2942   \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
2943   \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
2944 }
2945 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2946 {
2947   \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
2948   \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
2949 }

```

(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.

```

2950 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2951 \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>.

```

2952 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2953 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2954 \cs_new_protected:Npn \__pdf_backend_emc:
2955 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2956 </dviPDFmx | xetex>

```

6.5 dvisvgm backend

```

2957 <*dvisvgm>

```

6.5.1 Catalogue entries

_pdf_backend_catalog_gput:nn No-op.

```

2958 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2 { }
2959 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2 { }

```

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

6.5.2 Objects

_pdf_backend_object_new:nn All no-ops here.

```

2960 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2 { }
2961 \cs_new:Npn \__pdf_backend_object_ref:n #1 { }
2962 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2 { }
2963 \cs_new_protected:Npn \__pdf_backend_object_write:nx #1#2 { }
2964 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2 { }
2965 \cs_new_protected:Npn \__pdf_backend_object_now:nx #1#2 { }
2966 \cs_new:Npn \__pdf_backend_object_last: { }
2967 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1 { }

```

(End definition for __pdf_backend_object_new:nn and others.)

6.5.3 Structure

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2968 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2969 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1 { }

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

\__pdf_backend_version_major_gset:n
\__pdf_backend_version_minor_gset:n
2970 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2971 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1 { }

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

\__pdf_backend_version_major:
\__pdf_backend_version_minor:
2972 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2973 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

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

\__pdf_backend_bdc:nn
\__pdf_backend_emc:
2974 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2975 \cs_new_protected:Npn \__pdf_backend_emc: { }

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

2976 </dvisvgm>
2977 </package>

```

7 l3backend-opacity Implementation

```

2978 <*package>
2979 <@@=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.

```

2980 <*dvips>

```

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:n
\__opacity_backend_select_aux:n
\__opacity_backend_fill:n
\__opacity_backend_stroke:n
\__opacity_backend:nnn
\__opacity_backend:xnn
2981 \cs_new_protected:Npn \__opacity_backend_select:n #1
2982 {
2983   \exp_args:Nx \__opacity_backend_select_aux:n
2984   { \fp_eval:n { min(max(0,#1),1) } }
2985 }
2986 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
2987 {
2988   \__opacity_backend:nnn {#1} { fill } { ca }

```

```

2989   \_opacity_backend:nnn {#1} { stroke } { CA }
2990 }
2991 \cs_new_protected:Npn \_opacity_backend_fill:n #1
2992 {
2993   \_opacity_backend:xnn
2994   { \fp_eval:n { min(max(0,#1),1) } }
2995   { fill }
2996   { ca }
2997 }
2998 \cs_new_protected:Npn \_opacity_backend_stroke:n #1
2999 {
3000   \_opacity_backend:xnn
3001   { \fp_eval:n { min(max(0,#1),1) } }
3002   { stroke }
3003   { CA }
3004 }
3005 \cs_new_protected:Npn \_opacity_backend:nnn #1#2#3
3006 {
3007   \_kernel_backend_postscript:n
3008   {
3009     product ~ (Ghostscript) ~ search
3010     {
3011       pop ~ pop ~ pop ~
3012       #1 ~ .set #2 constantalpha
3013     }
3014     {
3015       pop ~
3016       mark ~
3017       /#3 ~ #1
3018       /SetTransparency ~
3019       pdfmark
3020     }
3021     ifelse
3022   }
3023 }
3024 \cs_generate_variant:Nn \_opacity_backend:nnn { x }

```

(End definition for _opacity_backend_select:n and others.)

```

3025 </dvips>
3026 <*dvipdfmx | luatex | pdftex | xetex>

```

\c_opacity_backend_stack_int Set up a stack.

```

3027 \bool_lazy_and:nnT
3028 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3029 { \pdfmanagement_if_active_p:}
3030 {
3031   \_kernel_color_backend_stack_init:Nnn \c_opacity_backend_stack_int
3032   { page ~ direct } { /opacity 1 ~ gs }
3033   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3034   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3035 }

```

(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`

```

3036 \tl_new:N \l__opacity_backend_fill_tl
3037 \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:`

```

3038 \cs_new_protected:Npn \__opacity_backend_select:n #1
3039 {
3040   \exp_args:Nx \__opacity_backend_select_aux:n
3041   { \fp_eval:n { min(max(0,#1),1) } }
3042 }
3043 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3044 {
3045   \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3046   \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3047   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3048   { opacity #1 }
3049   { << /ca ~ #1 /CA ~ #1 >> }
3050   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3051   { /opacity #1 ~ gs }
3052   \group_insert_after:N \__opacity_backend_reset:
3053 }
3054 \bool_lazy_and:nnF
3055 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3056 { \pdfmanagement_if_active_p:}
3057 {
3058   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1 { }
3059 }
3060 \cs_new_protected:Npn \__opacity_backend_reset:
3061 { \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int }

```

(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:nn`

`__opacity_backend_fillstroke:xx`

```

3062 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3063 {
3064   \__opacity_backend_fill_stroke:xx
3065   { \fp_eval:n { min(max(0,#1),1) } }
3066   \l__opacity_backend_stroke_tl
3067 }
3068 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3069 {
3070   \__opacity_backend_fill_stroke:xx
3071   \l__opacity_backend_fill_tl
3072   { \fp_eval:n { min(max(0,#1),1) } }
3073 }
3074 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3075 {
3076   \str_if_eq:nnTF {#1} {#2}
3077   { \__opacity_backend_select_aux:n {#1} }
3078   {

```



```

3079 \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3080 \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3081 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3082 { opacity.fill #1 }
3083 { << /ca ~ #1 >> }
3084 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3085 { opacity.stroke #1 }
3086 { << /CA ~ #2 >> }
3087 \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3088 { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3089 \group_insert_after:N \__opacity_backend_reset:
3090 }
3091 }
3092 \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.)

3093 </dvipdfmx | luatex | pdftex | xetex>
3094 <*dvipdfmx | xdvipdfmx>

```

__opacity_backend_select:n Older backends have no stack support, so everything is done directly.

```

3095 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
3096 {
3097   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1
3098   {
3099     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3100     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3101     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3102     { opacity #1 }
3103     { << /ca ~ #1 /CA ~ #1 >> }
3104     \__kernel_backend_literal_pdf:n { /opacity #1 ~ gs }
3105   }
3106   \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3107   {
3108     \str_if_eq:nnTF {#1} {#2}
3109     { \__opacity_backend_select_aux:n {#1} }
3110     {
3111       \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3112       \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3113       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3114       { opacity.fill #1 }
3115       { << /ca ~ #1 >> }
3116       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3117       { opacity.stroke #1 }
3118       { << /CA ~ #2 >> }
3119       \__kernel_backend_literal_pdf:n
3120       { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3121     }
3122   }
3123 }

```

(End definition for __opacity_backend_select:n.)

```

3124 </dvipdfmx | xdvipdfmx>

```

```
3125 <*dvisvgm>
```

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.

```
3126 \cs_new_protected:Npn \__opacity_backend_select:n #1
3127   { \__opacity_backend:nn {#1} { } }
3128 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3129   { \__opacity_backend:nn {#1} { fill- } }
3130 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3131   { \__opacity_backend:nn { {#1} } { stroke- } }
3132 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3133   { \__kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }
```

(End definition for __opacity_backend_select:n and others.)

```
3134 </dvisvgm>
```

```
3135 </package>
```

8 l3backend-header Implementation

```
3136 <*dvips & header>
```

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

```
3137 /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.

```
3138 TeXDict begin
3139 /TeXcolorseparation { setcolor } def
3140 end
```

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

pdf.globaldict A small global dictionary for backend use.

```
3141 true setglobal
3142 /pdf.globaldict 4 dict def
3143 false setglobal
```

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

pdf.cvs pdf.dvi.pt pdf.pt.dvi pdf.rect.ht 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.

```
3144 /pdf.cvs { 65534 string cvs } def
3145 /pdf.dvi.pt { 72.27 mul Resolution div } def
3146 /pdf.pt.dvi { 72.27 div Resolution mul } def
3147 /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 3148 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad 3149 /pdf.linkdp.pad { 0 } def
               3150 /pdf.linkht.pad { 0 } def

(End definition for pdf.linkmargin, pdf.linkdp.pad, and pdf.linkht.pad. These functions are docu-
mented on page ??.)

pdf.rect Functions for marking the limits of an annotation/link, plus drawing the border. We
pdf.save.ll separate links for generic annotations to support adding a margin and setting a minimal
pdf.save.ur size.
pdf.save.linkll 3151 /pdf.rect
pdf.save.linkur 3152 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.llx 3153 /pdf.save.ll
pdf.lly 3154 {
pdf.urx 3155     currentpoint
pdf.ury 3156     /pdf.lly exch def
               3157     /pdf.llx exch def
               3158 }
               3159 def
               3160 /pdf.save.ur
               3161 {
               3162     currentpoint
               3163     /pdf.ury exch def
               3164     /pdf.urx exch def
               3165 }
               3166 def
               3167 /pdf.save.linkll
               3168 {
               3169     currentpoint
               3170     pdf.linkmargin add
               3171     pdf.linkdp.pad add
               3172     /pdf.lly exch def
               3173     pdf.linkmargin sub
               3174     /pdf.llx exch def
               3175 }
               3176 def
               3177 /pdf.save.linkur
               3178 {
               3179     currentpoint
               3180     pdf.linkmargin sub
               3181     pdf.linkht.pad sub
               3182     /pdf.ury exch def
               3183     pdf.linkmargin add
               3184     /pdf.urx exch def
               3185 }
               3186 def

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

pdf.dest.anchor For finding the anchor point of a destination link. We make the use case a separate
pdf.dest.x function as it comes up a lot, and as this makes it easier to adjust if we need additional
pdf.dest.y effects. We also need a more complex approach to convert a co-ordinate pair correctly
pdf.dest.point
pdf.dest2device
pdf.dev.x
pdf.dev.y
pdf.tmpa
pdf.tmpb
pdf.tmpc
pdf.tmpd

```

when defining a rectangle: this can otherwise be out when using a landscape page.
(Thanks to Alexander Grahn for the approach here.)

```

3187 /pdf.dest.anchor
3188 {
3189     currentpoint exch
3190     pdf.dvi.pt 72 add
3191     /pdf.dest.x exch def
3192     pdf.dvi.pt
3193     vsize 72 sub exch sub
3194     /pdf.dest.y exch def
3195 }
3196 def
3197 /pdf.dest.point
3198 { pdf.dest.x pdf.dest.y } def
3199 /pdf.dest2device
3200 {
3201     /pdf.dest.y exch def
3202     /pdf.dest.x exch def
3203     matrix currentmatrix
3204     matrix defaultmatrix
3205     matrix invertmatrix
3206     matrix concatmatrix
3207     cvx exec
3208     /pdf.dev.y exch def
3209     /pdf.dev.x exch def
3210     /pdf.tmpd exch def
3211     /pdf.tmpc exch def
3212     /pdf.tmpb exch def
3213     /pdf.tmpa exch def
3214     pdf.dest.x pdf.tmpa mul
3215     pdf.dest.y pdf.tmpc mul add
3216     pdf.dev.x add
3217     pdf.dest.x pdf.tmpb mul
3218     pdf.dest.y pdf.tmpd mul add
3219     pdf.dev.y add
3220 }
3221 def

```

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

pdf.bordertracking	To know where a breakable link can go, we need to track the boundary rectangle. That
pdf.bordertracking.begin	can be done by hooking into a and x operations: those names have to be retained. The
pdf.bordertracking.end	boundary is stored at the end of the operation. Special effort is needed at the start and
pdf.leftboundary	end of pages (or rather galleys), such that everything works properly.
pdf.rightboundary	
pdf.brokenlink.rect	3222 /pdf.bordertracking false def
pdf.brokenlink.skip	3223 /pdf.bordertracking.begin
pdf.brokenlink.dict	3224 {
pdf.bordertracking.endpage	3225 SDict /pdf.bordertracking true put
pdf.bordertracking.continue	3226 SDict /pdf.leftboundary undef
pdf.originx	3227 SDict /pdf.rightboundary undef
pdf.originy	3228 /a where
	3229 {
	3230 /a
	3231 {

```

3232         currentpoint pop
3233         SDict /pdf.rightboundary known dup
3234         {
3235             SDict /pdf.rightboundary get 2 index lt
3236             { not }
3237             if
3238         }
3239         if
3240         { pop }
3241         { SDict exch /pdf.rightboundary exch put }
3242         ifelse
3243         moveto
3244         currentpoint pop
3245         SDict /pdf.leftboundary known dup
3246         {
3247             SDict /pdf.leftboundary get 2 index gt
3248             { not }
3249             if
3250         }
3251         if
3252         { pop }
3253         { SDict exch /pdf.leftboundary exch put }
3254         ifelse
3255     }
3256     put
3257 }
3258 if
3259 }
3260 def
3261 /pdf.bordertracking.end
3262 {
3263     /a where { /a { moveto } put } if
3264     /x where { /x { 0 exch rmoveto } put } if
3265     SDict /pdf.leftboundary known
3266     { pdf.outerbox 0 pdf.leftboundary put }
3267     if
3268     SDict /pdf.rightboundary known
3269     { pdf.outerbox 2 pdf.rightboundary put }
3270     if
3271     SDict /pdf.bordertracking false put
3272 }
3273 def
3274 /pdf.bordertracking.endpage
3275 {
3276     pdf.bordertracking
3277     {
3278         pdf.bordertracking.end
3279         true setglobal
3280         pdf.globaldict
3281         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3282         pdf.globaldict
3283         /pdf.brokenlink.skip pdf.baselineskip put
3284         pdf.globaldict
3285         /pdf.brokenlink.dict

```

```

3286         pdf.link.dict pdf.cvs put
3287     false setglobal
3288     mark pdf.link.dict cvx exec /Rect
3289     [
3290         pdf.llx
3291         pdf.lly
3292         pdf.outerbox 2 get pdf.linkmargin add
3293         currentpoint exch pop
3294         pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3295     ]
3296     /ANN pdf.pdfmark
3297 }
3298 if
3299 }
3300 def
3301 /pdf.bordertracking.continue
3302 {
3303     /pdf.link.dict pdf.globaldict
3304     /pdf.brokenlink.dict get def
3305     /pdf.outerbox pdf.globaldict
3306     /pdf.brokenlink.rect get def
3307     /pdf.baselineskip pdf.globaldict
3308     /pdf.brokenlink.skip get def
3309     pdf.globaldict dup dup
3310     /pdf.brokenlink.dict undef
3311     /pdf.brokenlink.skip undef
3312     /pdf.brokenlink.rect undef
3313     currentpoint
3314     /pdf.originy exch def
3315     /pdf.originx exch def
3316     /a where
3317     {
3318         /a
3319         {
3320             moveto
3321             SDict
3322             begin
3323                 currentpoint pdf.originy ne exch
3324                 pdf.originx ne or
3325                 {
3326                     pdf.save.linkll
3327                     /pdf.lly
3328                     pdf.lly pdf.outerbox 1 get sub def
3329                     pdf.bordertracking.begin
3330                 }
3331                 if
3332                 end
3333             }
3334             put
3335         }
3336         if
3337         /x where
3338         {
3339             /x

```

```

3340     {
3341         0 exch rmoveto
3342         SDict
3343         begin
3344         currentpoint
3345         pdf.originy ne exch pdf.originx ne or
3346         {
3347             pdf.save.linkll
3348             /pdf.lly
3349             pdf.lly pdf.outerbox 1 get sub def
3350             pdf.bordertracking.begin
3351         }
3352         if
3353         end
3354     }
3355     put
3356 }
3357 if
3358 }
3359 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
pdf.breaklink.write in the dictionary, looping over key-value pairs. The first line is handled first, adjusting
 pdf.count the rectangle to stay inside the text area. The second phase is a loop over the height of
pdf.currentrect the bulk of the link area, done on the basis of a number of baselines. Finally, the end of
 the link area is tidied up, again from the boundary of the text area.

```

3360 /pdf.breaklink
3361 {
3362     pop
3363     counttomark 2 mod 0 eq
3364     {
3365         counttomark /pdf.count exch def
3366         {
3367             pdf.count 0 eq { exit } if
3368             counttomark 2 roll
3369             1 index /Rect eq
3370             {
3371                 dup 4 array copy
3372                 dup dup
3373                 1 get
3374                 pdf.outerbox pdf.rect.ht
3375                 pdf.linkmargin 2 mul add sub
3376                 3 exch put
3377                 dup
3378                 pdf.outerbox 2 get
3379                 pdf.linkmargin add
3380                 2 exch put
3381                 dup dup
3382                 3 get
3383                 pdf.outerbox pdf.rect.ht
3384                 pdf.linkmargin 2 mul add add
3385                 1 exch put

```

```

3386      /pdf.currentrect exch def
3387      pdf.breaklink.write
3388      {
3389          pdf.currentrect
3390          dup
3391              pdf.outerbox 0 get
3392              pdf.linkmargin sub
3393              0 exch put
3394          dup
3395              pdf.outerbox 2 get
3396              pdf.linkmargin add
3397              2 exch put
3398          dup dup
3399              1 get
3400              pdf.baselineskip add
3401              1 exch put
3402          dup dup
3403              3 get
3404              pdf.baselineskip add
3405              3 exch put
3406          /pdf.currentrect exch def
3407          pdf.breaklink.write
3408      }
3409      1 index 3 get
3410      pdf.linkmargin 2 mul add
3411      pdf.outerbox pdf.rect.ht add
3412      2 index 1 get sub
3413      pdf.baselineskip div round cvi 1 sub
3414      exch
3415      repeat
3416      pdf.currentrect
3417      dup
3418          pdf.outerbox 0 get
3419          pdf.linkmargin sub
3420          0 exch put
3421      dup dup
3422          1 get
3423          pdf.baselineskip add
3424          1 exch put
3425      dup dup
3426          3 get
3427          pdf.baselineskip add
3428          3 exch put
3429      dup 2 index 2 get 2 exch put
3430      /pdf.currentrect exch def
3431      pdf.breaklink.write
3432      SDict /pdf.pdfmark.good false put
3433      exit
3434      }
3435      { pdf.count 2 sub /pdf.count exch def }
3436      ifelse
3437      }
3438      loop
3439      }

```



```

3440   if
3441   /ANN
3442 }
3443 def
3444 /pdf.breaklink.write
3445 {
3446   counttomark 1 sub
3447   index /_objdef eq
3448   {
3449     counttomark -2 roll
3450     dup wcheck
3451     {
3452       readonly
3453       counttomark 2 roll
3454     }
3455     { pop pop }
3456   } ifelse
3457 }
3458 if
3459 counttomark 1 add copy
3460 pop pdf.currentrect
3461 /ANN pdfmark
3462 }
3463 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.

```

3464 /pdf.pdfmark
3465 {
3466   SDict /pdf.pdfmark.good true put
3467   dup /ANN eq
3468   {
3469     pdf.pdfmark.store
3470     pdf.pdfmark.dict
3471     begin
3472       Subtype /Link eq
3473       currentdict /Rect known and
3474       SDict /pdf.outerbox known and
3475       SDict /pdf.baselineskip known and
3476       {
3477         Rect 3 get
3478         pdf.linkmargin 2 mul add
3479         pdf.outerbox pdf.rect.ht add
3480         Rect 1 get sub
3481         pdf.baselineskip div round cvi 0 gt
3482         { pdf.breaklink }
3483       } if
3484     }
3485   } if

```

```

3486         end
3487         SDict /pdf.outerbox undef
3488         SDict /pdf.baselineskip undef
3489         currentdict /pdf.pdfmark.dict undef
3490     }
3491     if
3492     pdf.pdfmark.good
3493     { pdfmark }
3494     { cleartomark }
3495     ifelse
3496 }
3497 def
3498 /pdf.pdfmark.store
3499 {
3500     /pdf.pdfmark.dict 65534 dict def
3501     counttomark 1 add copy
3502     pop
3503     {
3504         dup mark eq
3505         {
3506             pop
3507             exit
3508         }
3509         {
3510             pdf.pdfmark.dict
3511             begin def end
3512         }
3513         ifelse
3514     }
3515     loop
3516 }
3517 def

```

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

```

3518 </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.

<code>_</code>	154	<code>_box_backend_rotate:Nn</code>	235, 283, 340, <u>419</u>
A			
<code>\AtBeginDvi</code>	57	<code>_box_backend_rotate_aux:Nn</code> ...	235, 283, <u>340</u>
B			
bool commands:			
<code>\bool_gset_false:N</code>	1173, 1192, 1215, 1237, 1253, 1354, 1591, 1627, 2195, 2241	<code>_box_backend_scale:Nnn</code>	252, <u>311</u> , 355, <u>432</u>
<code>\bool_gset_true:N</code>	1171, 1240, 1352, 1606, 2188, 2194	<code>\l_box_backend_sin_fp</code>	<u>283</u>
<code>\bool_if:NTF</code>	67, 675, 1183, 1187, 1203, 1206, 1210, 1221, 1228, 1232, 1244, 1248, 1365, 1370, 1375, 1565, 1610, 1723, 1758, 1868, 1910, 2183, 2198, 2203, 2208	<code>\g_box_clip_path_int</code>	<u>369</u>
<code>\bool_if:nTF</code>	2417, 2683, 2937	C	
<code>\bool_lazy_and:nnTF</code>	866, 936, 3027, 3054	char commands:	
<code>\bool_lazy_or:nnTF</code>	1750, 1903	<code>\char_set_catcode_space:n</code>	154
<code>\bool_new:N</code>	1174, 1241, 1355, 1607, 2168, 2169	clist commands:	
<code>\bool_set_false:N</code>	1733, 1835, 1928, 1992	<code>\clist_map_function:nN</code> ...	1261, 1385
box commands:			
<code>\box_dp:N</code>	224, 226, 274, 276, 331, 333, 380, 382, 384, 386, 2220, 2253, 2254, 2279	<code>\clist_map_function:nn</code>	1634
<code>\box_ht:N</code>	226, 276, 333, 384, 386, 1770, 1965, 2225, 2264, 2265, 2281	color internal commands:	
<code>\box_if_empty:N</code>	2314	<code>_color_backend:nnn</code>	1059
<code>\box_move_down:nn</code>	2142, 2220	<code>_color_backend_cmyk:w</code>	1060
<code>\box_move_up:nn</code>	2144, 2225	<code>_color_backend_devicen_init:n</code>	<u>905</u>
<code>\box_new:N</code>	2027, 2132, 2133	<code>_color_backend_devicen_-init:nnn</code>	821, <u>905</u>
<code>\box_set_dp:Nn</code>	1690	<code>_color_backend_devicen_init:w</code>	<u>905</u>
<code>\box_set_ht:Nn</code>	1689	<code>_color_backend_fill:n</code>	966, 993, <u>1023</u> , <u>1041</u> , <u>1048</u>
<code>\box_set_wd:Nn</code>	288, 1688	<code>_color_backend_fill_cmyk:n</code> ...	966, <u>1000</u> , <u>1023</u> , <u>1048</u>
<code>\box_use:N</code>	231, 249, 263, 279, 306, 320, 336, 352, 364, 415, 429, 448, 1305, 1500, 1691, 2173	<code>_color_backend_fill_devicen:nn</code>	992, <u>1014</u> , <u>1040</u> , <u>1110</u>
<code>\box_wd:N</code>	225, 233, 275, 281, 332, 338, 381, 383, 1769, 1964	<code>_color_backend_fill_gray:n</code> ...	966, <u>1000</u> , <u>1023</u> , <u>1048</u>
box internal commands:			
<code>_box_backend_clip:N</code>	213, 268, 325, <u>369</u>	<code>_color_backend_fill_rgb:n</code> ...	966, <u>1000</u> , <u>1023</u> , <u>1048</u>
<code>\l_box_backend_cos_fp</code>	<u>283</u>	<code>_color_backend_fill_separation:nn</code>	992, <u>1000</u> , <u>1040</u> , <u>1110</u>
		<code>\l_color_backend_fill_tl</code>	637, 647, 974, 989
		<code>\c_color_backend_main_stack_int</code>	516
		<code>_color_backend_pickup:N</code> ..	456, <u>479</u>
		<code>_color_backend_pickup:w</code> ..	14, 456, <u>479</u>
		<code>_color_backend_reset:</code>	619, 639, 656, 977, 990, <u>1000</u> , 1032, 1057
		<code>_color_backend_rgb:w</code>	1083
		<code>_color_backend_select:n</code> ..	619, 671
		<code>_color_backend_select:nn</code> ..	639, 848
		<code>_color_backend_select_cmyk:n</code> ..	619, 639, <u>656</u>
		<code>_color_backend_select_devicen:nn</code>	670, <u>841</u> , <u>847</u> , <u>958</u>

_color_backend_select_gray:n ..	_color_backend_stroke_separation:nn
..... 619 , 639 , 656 992 , 1000 , 1040 , 1110
_color_backend_select_rgb:n ...	\l_color_backend_stroke_tl
..... 619 , 639 , 656 637 , 648 , 976 , 987
_color_backend_select_separation:nn	\g_color_model_int 691 , 827 , 874 , 944
..... 670 , 841 , 847 , 958	\c_color_model_range_CIELAB_tl .
_color_backend_separation_- 782 , 817 , 894 , 901
init:n 673 , 850 , 929 , 955	color.sc 619 , 3137
_color_backend_separation_-	cs commands:
init:nnn 673	\cs_generate_variant:Nn .. 49 , 63 ,
_color_backend_separation_-	66 , 99 , 138 , 143 , 170 , 201 , 207 , 569 ,
init:nnnn 673	606 , 684 , 1120 , 1315 , 1509 , 1882 ,
_color_backend_separation_-	1939 , 1955 , 2031 , 2068 , 2127 , 2619 ,
init:nnnnn 673 , 843 , 850	2647 , 2757 , 2779 , 2814 , 3024 , 3092
_color_backend_separation_-	\cs_gset:Npx .. 2429 , 2433 , 2942 , 2947
init:nw 673	\cs_gset_eq:NN 663 ,
_color_backend_separation_-	664 , 961 , 1007 , 1008 , 1014 , 1016 , 1018
init:w 673	\cs_gset_protected:Npn
_color_backend_separation_- 551 , 658 , 665 , 960 , 1002 ,
init_/DeviceCMYK:nnn 673	1009 , 1011 , 1013 , 3058 , 3097 , 3106
_color_backend_separation_-	\cs_if_exist:NTF 27 ,
init_/DeviceGray:nnn 673	50 , 457 , 480 , 539 , 2310 , 2708 , 2734
_color_backend_separation_-	\cs_if_exist_p:N . 867 , 937 , 3028 , 3055
init_/DeviceRGB:nnn 673	\cs_if_exist_use:NTF 38 , 697
_color_backend_separation_-	\cs_new:Npn 706 , 708 , 710 ,
init_aux:nnnnn 673	712 , 719 , 725 , 727 , 733 , 750 , 757 ,
_color_backend_separation_-	759 , 949 , 1266 , 1390 , 1638 , 1968 ,
init_CIELAB:nnn 673 , 843 , 850	1977 , 2021 , 2046 , 2128 , 2130 , 2163 ,
_color_backend_separation_-	2335 , 2435 , 2436 , 2589 , 2620 , 2621 ,
init_CIELAB:nnnnnn 844	2739 , 2772 , 2815 , 2817 , 2833 , 2950 ,
_color_backend_separation_-	2951 , 2961 , 2966 , 2967 , 2972 , 2973
init_count:n 673	\cs_new:Npx
_color_backend_separation_-	.. 2456 , 2491 , 2648 , 2659 , 2726 , 2863
init_count:w 673	\cs_new_eq:NN
_color_backend_separation_- 46 , 57 , 59 , 672 , 849 , 955 ,
init_Device:Nn 673	996 , 997 , 1044 , 1045 , 1112 , 1113 ,
\g_color_backend_stack_int ... 516	1119 , 1314 , 1320 , 1321 , 1508 , 1515 ,
\l_color_backend_stack_int	1700 , 1729 , 1780 , 1781 , 1823 , 1831 ,
.... 513 , 541 , 547 , 649 , 653 , 975 , 988	1853 , 1924 , 1981 , 1988 , 2020 , 2173
_color_backend_stroke:n	\cs_new_protected:Npn ... 47 , 54 ,
..... 966 , 995 , 1000	61 , 64 , 72 , 78 , 83 , 85 , 89 , 100 , 110 ,
_color_backend_stroke_cmyk:n ..	119 , 128 , 141 , 144 , 146 , 148 , 168 ,
..... 966 , 1023 , 1059	173 , 182 , 192 , 202 , 213 , 235 , 237 ,
_color_backend_stroke_cmyk:w 1059	252 , 268 , 283 , 285 , 311 , 325 , 340 ,
_color_backend_stroke_devicen:nn	342 , 355 , 369 , 419 , 432 , 456 , 474 ,
..... 992 , 1018 , 1040 , 1110	479 , 487 , 517 , 560 , 570 , 582 , 596 ,
_color_backend_stroke_gray:n ..	607 , 619 , 621 , 623 , 625 , 633 , 639 ,
..... 966 , 1023 , 1059	641 , 643 , 645 , 652 , 670 , 685 , 775 ,
_color_backend_stroke_gray_-	821 , 841 , 842 , 843 , 844 , 847 , 850 ,
aux:n 1059	879 , 883 , 905 , 966 , 968 , 970 , 972 ,
_color_backend_stroke_rgb:n ...	979 , 981 , 983 , 985 , 992 , 994 , 1023 ,
..... 966 , 1023 , 1059	1025 , 1027 , 1029 , 1034 , 1036 , 1038 ,
_color_backend_stroke_rgb:w . 1059	1040 , 1042 , 1048 , 1050 , 1052 , 1054 ,
	1059 , 1061 , 1072 , 1080 , 1082 , 1084 ,

1110, 1111, 1121, 1126, 1131, 1133,
 1135, 1143, 1151, 1160, 1170, 1172,
 1175, 1177, 1194, 1199, 1217, 1239,
 1242, 1255, 1268, 1273, 1275, 1277,
 1279, 1281, 1283, 1285, 1287, 1292,
 1316, 1318, 1322, 1327, 1332, 1342,
 1351, 1353, 1356, 1358, 1360, 1362,
 1367, 1372, 1377, 1379, 1392, 1397,
 1399, 1401, 1403, 1405, 1407, 1409,
 1411, 1422, 1447, 1459, 1471, 1483,
 1490, 1510, 1516, 1521, 1526, 1537,
 1547, 1557, 1559, 1561, 1563, 1594,
 1596, 1601, 1603, 1605, 1608, 1629,
 1640, 1653, 1655, 1657, 1659, 1661,
 1663, 1665, 1667, 1669, 1677, 1701,
 1715, 1730, 1742, 1747, 1775, 1787,
 1800, 1810, 1825, 1832, 1840, 1851,
 1855, 1858, 1873, 1883, 1918, 1925,
 1931, 1937, 1940, 1947, 1956, 1961,
 1969, 1982, 1989, 1995, 1997, 1999,
 2010, 2029, 2032, 2034, 2038, 2048,
 2069, 2074, 2079, 2084, 2094, 2099,
 2107, 2135, 2140, 2172, 2174, 2179,
 2181, 2186, 2201, 2206, 2243, 2272,
 2291, 2300, 2337, 2344, 2370, 2375,
 2403, 2415, 2427, 2431, 2437, 2439,
 2443, 2467, 2469, 2471, 2482, 2502,
 2512, 2535, 2549, 2559, 2570, 2591,
 2622, 2670, 2681, 2687, 2715, 2749,
 2751, 2758, 2760, 2764, 2774, 2780,
 2785, 2790, 2795, 2797, 2799, 2807,
 2820, 2836, 2838, 2861, 2871, 2873,
 2895, 2900, 2933, 2935, 2940, 2945,
 2952, 2954, 2958, 2959, 2960, 2962,
 2963, 2964, 2965, 2968, 2969, 2970,
 2971, 2974, 2975, 2981, 2986, 2991,
 2998, 3005, 3038, 3043, 3060, 3062,
 3068, 3074, 3126, 3128, 3130, 3132
 \cs_new_protected:Npx
 520, 673, 1095, 2698, 2755, 2840
 \cs_set:Npn 152
 \cs_set_eq:NN 2331, 2332
 \cs_set_protected:Npn 459, 482

D

dim commands:

\dim_eval:n 2138, 2373,
 2451, 2452, 2453, 2510, 2545, 2546,
 2547, 2827, 2828, 2829, 2872, 2898
 \dim_max:nn 2251, 2262
 \dim_set:Nn ... 1769, 1770, 1964, 1965
 \dim_to_decimal:n .. 380, 381, 382,
 383, 384, 386, 1519, 1524, 1530,
 1531, 1532, 1533, 1542, 1543, 1544,

1635, 1654, 2015, 2016, 2249, 2260,
 2278, 2279, 2280, 2281, 2285, 2341
 \dim_to_decimal_in_bp:n
 224, 225, 226, 274, 275, 276,
 331, 332, 333, 1139, 1140, 1147,
 1148, 1155, 1156, 1164, 1165, 1166,
 1263, 1267, 1271, 1325, 1330, 1336,
 1337, 1338, 1346, 1347, 1387, 1391,
 1395, 1639, 1706, 1707, 1708, 1709,
 1845, 1846, 1847, 1848, 1897, 1898,
 1899, 1900, 2004, 2005, 2006, 2007
 draw internal commands:
 __draw_align_currentpoint... .. 35
 __draw_backend_add_to_path:n ...
 1516, 1562
 __draw_backend_begin:
 1121, 1316, 1510
 __draw_backend_box_use:Nnnnn ...
 30, 1292, 1490, 1677
 __draw_backend_cap_but:
 1255, 1379, 1629
 __draw_backend_cap_rectangle: ..
 1255, 1379, 1629
 __draw_backend_cap_round:
 1255, 1379, 1629
 __draw_backend_clip: 1175, 1356, 1561
 __draw_backend_closepath:
 1175, 1356, 1561
 __draw_backend_closestroke: ...
 1175, 1356, 1561
 __draw_backend_cm:nnnn 1287, 1300,
 1301, 1302, 1411, 1494, 1669, 1680
 __draw_backend_cm_aux:nnnn .. 1411
 __draw_backend_cm_decompose:nnnnN
 1417, 1446
 __draw_backend_cm_decompose_-
 auxi:nnnnN 1446
 __draw_backend_cm_decompose_-
 auxii:nnnnN 1446
 __draw_backend_cm_decompose_-
 auxiii:nnnnN 1446
 __draw_backend_curveto:nnnnnn ..
 1135, 1322, 1516
 __draw_backend_dash:n
 1255, 1379, 1629
 __draw_backend_dash_aux:nn .. 1629
 __draw_backend_dash_pattern:nn ..
 1255, 1379, 1629
 __draw_backend_discardpath: ...
 1175, 1356, 1561
 __draw_backend_end: 1121, 1316, 1510
 __draw_backend_evenodd_rule: ...
 1170, 1351, 1557
 __draw_backend_fill: 1175, 1356, 1561

<code>__draw_backend_fillstroke:</code>	
. 1175 , 1356 , 1561	
<code>__draw_backend_join_bevel:</code>	
. 1255 , 1379 , 1629	
<code>__draw_backend_join_miter:</code>	
. 1255 , 1379 , 1629	
<code>__draw_backend_join_round:</code>	
. 1255 , 1379 , 1629	
<code>__draw_backend_lineto:nn</code>	
. 1135 , 1322 , 1516	
<code>__draw_backend_linewidth:n</code>	
. 1255 , 1379 , 1629	
<code>__draw_backend_literal:n</code>	
. 1119 , 1124 , 1128 , 1132 ,	
1134 , 1137 , 1145 , 1153 , 1162 , 1176 ,	
1179 , 1180 , 1181 , 1182 , 1185 , 1191 ,	
1201 , 1208 , 1214 , 1219 , 1224 , 1225 ,	
1226 , 1227 , 1230 , 1236 , 1246 , 1252 ,	
1257 , 1270 , 1274 , 1276 , 1278 , 1280 ,	
1282 , 1284 , 1286 , 1289 , 1294 , 1295 ,	
1296 , 1297 , 1298 , 1299 , 1303 , 1304 ,	
1306 , 1307 , 1308 , 1309 , 1310 , 1314 ,	
1324 , 1329 , 1334 , 1344 , 1357 , 1359 ,	
1361 , 1364 , 1369 , 1374 , 1378 , 1381 ,	
1394 , 1398 , 1400 , 1402 , 1404 , 1406 ,	
1408 , 1410 , 1508 , 1568 , 1587 , 1613	
<code>__draw_backend_miterlimit:n</code> . . .	
. 1255 , 1379 , 1629	
<code>__draw_backend_moveto:nn</code>	
. 1135 , 1322 , 1516	
<code>__draw_backend_nonzero_rule:</code> . . .	
. 1170 , 1351 , 1557	
<code>__draw_backend_path:n</code> 1561	
<code>__draw_backend_rectangle:nmmn</code> . .	
. 1135 , 1322 , 1516	
<code>__draw_backend_scope:n</code> 1558 , 1560 ,	
1580 , 1620 , 1642 , 1654 , 1656 , 1658 ,	
1660 , 1662 , 1664 , 1666 , 1668 , 1671	
<code>__draw_backend_scope_begin:</code> . . .	
. 1131 , 1317 , 1320	
<code>__draw_backend_scope_end:</code>	
. 1131 , 1319 , 1320	
<code>__draw_backend_stroke:</code>	
. 1175 , 1356 , 1561	
<code>\g__draw_clip_path_int</code>	
. . . 1567 , 1570 , 1583 , 1612 , 1615 , 1623	
<code>\g__draw_draw_clip_bool</code> . . 1175 , 1561	
<code>\g__draw_draw_eor_bool</code>	
. . . 1170 , 1187 , 1203 , 1210 , 1221 ,	
1232 , 1248 , 1351 , 1365 , 1370 , 1375	
<code>\g__draw_draw_path_int</code> 1561	
<code>\g__draw_draw_path_tl</code>	
. . . 1516 , 1572 , 1588 , 1590 , 1617 , 1626	
<code>\g__draw_path_int</code> 1576 , 1593	
	E
	<code>\errmessage</code> 38
	<code>\evensidemargin</code> 2218
	exp commands:
	<code>\exp_after:wN</code> 159 , 465 , 1975
	<code>\exp_args:Ne</code> 721 , 2372 , 2897
	<code>\exp_args:Nf</code> 1260 , 1384 , 2137
	<code>\exp_args:NNf</code> 236 , 284 , 341
	<code>\exp_args:Nnx</code> 2124 , 2810
	<code>\exp_args:NV</code> 461
	<code>\exp_args:Nx</code> 1793 , 1814 ,
	2081 , 2096 , 2214 , 2776 , 2983 , 3040
	<code>\exp_last_unbraced:Nx</code> 470 , 484
	<code>\exp_not:N</code> 522 , 523 ,
	531 , 533 , 679 , 2458 , 2460 , 2463 ,
	2493 , 2495 , 2498 , 2650 , 2652 , 2655 ,
	2661 , 2663 , 2666 , 2703 , 2704 , 2710 ,
	2711 , 2730 , 2735 , 2844 , 2852 , 2868
	<code>\exp_not:n</code> . . . 48 , 97 , 108 , 136 , 2072 ,
	2077 , 2366 , 2605 , 2606 , 2620 , 2621 ,
	2633 , 2634 , 2788 , 2793 , 2804 , 2877
	<code>\ExplBackendFileDate</code> 1
	F
	file commands:
	<code>\file_compare_timestamp:nNnTF</code> . 1802
	<code>\file_parse_full_name:nNNN</code> 1789 , 1812
	<code>\fmtversion</code> 52
	fp commands:
	<code>\fp_compare:nNnTF</code>
	. . . 243 , 290 , 296 , 348 , 1427 , 1440 , 1485
	<code>\fp_eval:n</code> . . . 236 , 245 , 258 , 259 , 284 ,
	301 , 316 , 318 , 341 , 350 , 361 , 362 ,
	426 , 441 , 442 , 1067 , 1068 , 1069 ,
	1077 , 1090 , 1091 , 1092 , 1429 , 1434 ,
	1435 , 1442 , 1452 , 1453 , 1454 , 1455 ,
	1464 , 1465 , 1466 , 1467 , 1476 , 1477 ,
	1478 , 1479 , 2363 , 2532 , 2891 , 2984 ,
	2994 , 3001 , 3041 , 3065 , 3072 , 3133
	<code>\fp_new:N</code> 309 , 310
	<code>\fp_set:Nn</code> 289 , 292
	<code>\fp_use:N</code> 295 , 299 , 304
	<code>\fp_zero:N</code> 291
	<code>\c_zero_fp</code> 243 , 290 , 296 , 348 , 1427 , 1440
	G
	graphics commands:
	<code>\graphics_bb_restore:nTF</code> . 1744 , 1958
	<code>\graphics_bb_save:n</code> 1773 , 1966
	<code>\l_graphics_decodearray_tl</code>
 1721 , 1722 ,
	1732 , 1752 , 1756 , 1757 , 1834 , 1866 ,
	1867 , 1905 , 1908 , 1909 , 1927 , 1991
	<code>\graphics_extract_bb:n</code>
 1829 , 1836 , 1986 , 1993

\l_graphics_interpolate_bool . . .	__graphics_backend_include:nn 1995
. 1723 , 1733 , 1751 , 1758 ,	__graphics_backend_include_-
1835 , 1868 , 1904 , 1910 , 1928 , 1992	auxi:nn 1840
\l_graphics_llx_dim	__graphics_backend_include_-
. 1706 , 1845 , 1897 , 2004	auxii:nnn 1840
\l_graphics_lly_dim	__graphics_backend_include_-
. 1707 , 1846 , 1898 , 2005	auxiii:nnn 1840
\l_graphics_name_tl 1807	__graphics_backend_include_-
\l_graphics_page_int	bitmap_quote:w 1969 , 2010
. 1717 , 1737 , 1738 , 1762 ,	__graphics_backend_include_-
1763 , 1827 , 1864 , 1865 , 1891 , 1892 ,	eps:n 1701 , 1782 , 1840 , 1995
1920 , 1933 , 1934 , 1973 , 1974 , 1984	__graphics_backend_include_-
\l_graphics_pagebox_tl	jpg:n 1775 , 1840 , 2010
. 51 , 1718 , 1736 ,	__graphics_backend_include_-
1764 , 1765 , 1828 , 1862 , 1863 , 1893 ,	pdf:n 1775 , 1814 , 1840 , 1969 , 1995
1895 , 1921 , 1942 , 1943 , 1975 , 1985	__graphics_backend_include_pdf_-
\graphics_read_bb:n . 1700 , 1823 , 1981	quote:w 1972 , 1977
\l_graphics_urx_dim	__graphics_backend_include_-
. 1708 , 1769 , 1847 , 1899 , 1964 , 2006	png:n 1775 , 1840 , 2010
\l_graphics_ury_dim . . 1709 , 1770 ,	\l__graphics_backend_name_str . 1782
1848 , 1900 , 1965 , 2007 , 2015 , 2016	\l__graphics_graphics_attr_tl . . .
graphics internal commands: 1714 , 1719 ,
\l__graphics_backend_dir_str . 1782	1726 , 1734 , 1744 , 1771 , 1773 , 1778
\l__graphics_backend_ext_str . 1782	\l__graphics_internal_box
__graphics_backend_getbb_auxi:n 1767 , 1769 , 1770 , 1963 , 1964 , 1965
. 1715	\g__graphics_track_int
__graphics_backend_getbb_- 1839 , 1885 , 1886
auxi:nN 1918	group commands:
__graphics_backend_getbb_-	\group_begin: 151 , 179 , 198
auxii:n 1715	\group_end: 164 , 187
__graphics_backend_getbb_-	\group_insert_after:N 631 , 650 , 661 ,
auxii:nnN 1918	977 , 990 , 1005 , 1032 , 1057 , 3052 , 3089
__graphics_backend_getbb_-	
auxiii:nNnn 1918	H
__graphics_backend_getbb_-	hbox commands:
auxiv:nnNnn 1918	\hbox:n 2143 , 2146 ,
__graphics_backend_getbb_-	2221 , 2227 , 2380 , 2387 , 2905 , 2916
auxv:nNnn 1918	\hbox_overlap_right:n 231 ,
__graphics_backend_getbb_-	263 , 279 , 320 , 336 , 364 , 448 , 1305 , 1500
auxvi:nNnn 1959 , 1961	\hbox_set:Nn 1767 , 1963 , 2213 , 2245
__graphics_backend_getbb_eps:n .	\hbox_set:Nw 2196
. 1700 , 1782 , 1823 , 1981	\hbox_set_end: 2211
__graphics_backend_getbb_eps:nm	\hbox_unpack:N 2332
. 1782	hook commands:
__graphics_backend_getbb_eps:nn	\hook_gput_code:nnn 55
. 1793 , 1800	
__graphics_backend_getbb_jpg:n .	I
. 1715 , 1823 , 1918 , 1982	int commands:
__graphics_backend_getbb_-	\int_compare:nNnTF 516 ,
pagebox:w 1918 , 1975	558 , 656 , 958 , 1000 , 1737 , 1762 ,
__graphics_backend_getbb_pdf:n .	1864 , 1891 , 1933 , 1973 , 2304 , 2405 ,
. 1715 , 1808 , 1823 , 1918 , 1989	2701 , 2729 , 2842 , 2849 , 2865 , 3095
__graphics_backend_getbb_png:n .	\int_const:Nn 157 , 163 , 523 ,
. 1715 , 1823 , 1918 , 1982	549 , 584 , 1771 , 1886 , 2041 , 2579 , 2767

<code>\int_eval:n</code>	61, 75, 76, 80, 217, 218, 220, 221, 229, 241, 256, 1119, 2407, 2419
. 565, 575, 604, 615, 717, 726, 739, 741, 745, 758, 2429, 2433, 2679, 2704, 2711, 2724, 2934, 2942, 2947	
<code>\int_gincr:N</code>	205, 371, 522, 1567, 1612, 1885, 2040, 2109, 2153, 2230, 2766, 2809, 2822, 2844
<code>\int_gset:Nn</code>	180, 199, 2293
<code>\int_gset_eq:NN</code>	188, 2154, 2231, 2823
<code>\int_if_exist:NTF</code>	1875
<code>\int_if_odd:nTF</code>	2216
<code>\int_new:N</code>	171, 172, 418, 513, 519, 1593, 1839, 2036, 2134, 2165, 2167, 2762, 2819, 2835
<code>\int_set:Nn</code>	541
<code>\int_set_eq:NN</code> ...	176, 195, 547, 2305
<code>\int_step_function:nnnN</code>	743
<code>\int_use:N</code>	373, 404, 531, 542, 691, 827, 874, 944, 1570, 1576, 1583, 1615, 1623, 1738, 1763, 1778, 1865, 1878, 1890, 1892, 1974, 2047, 2112, 2125, 2129, 2157, 2164, 2235, 2336, 2590, 2600, 2773, 2811, 2816, 2826, 2834, 2852, 2868
<code>\int_value:w</code>	2458, 2493, 2650, 2661, 2679
<code>\int_zero:N</code> ...	1717, 1827, 1920, 1984
K	
kernel internal commands:	
<code>__kernel_backend_align_begin: ..</code>	72, 216, 240, 255
<code>__kernel_backend_align_end: ...</code>	72, 230, 248, 262
<code>__kernel_backend_first_shipout:n</code>	50, 69, 526, 677
<code>\g__kernel_backend_header_bool ..</code>	67, 675
<code>__kernel_backend_literal:n</code>	46, 62, 65, 70, 74, 81, 84, 86, 142, 145, 147, 149, 169, 345, 358, 528, 553, 554, 562, 572, 627, 634, 660, 666, 687, 823, 1004, 1010, 1012, 1031, 1056, 1123, 1129, 1424, 1431, 1437, 1497, 1502, 1703, 1842, 1877, 1887, 2001, 2012, 2756, 2872, 2934, 2938, 2943, 2948
<code>__kernel_backend_literal_page:n</code>	100, 144, 2750, 2752, 2953, 2955
<code>__kernel_backend_literal_pdf:n .</code>	89, 141, 271, 328, 1314, 3104, 3119
<code>__kernel_backend_literal_-</code>	
<code>postscript:n</code>	
<code>__kernel_backend_literal_svg:n .</code>	168, 175, 186, 194, 204, 372, 374, 391, 1508, 1681, 1692
<code>__kernel_backend_matrix:n</code>	128, 293, 314, 1414
<code>__kernel_backend_postscript:n ..</code>	64, 629, 1035, 1037, 1039, 1043, 2030, 2086, 2101, 2143, 2149, 2189, 2221, 2228, 2232, 2246, 2274, 2318, 2325, 2331, 2339, 2346, 2380, 2387, 3007
<code>__kernel_backend_scope:n</code>	173, 401, 406, 1097, 1513, 3133
<code>__kernel_backend_scope_begin: ..</code>	83, 110, 146, 173, 215, 239, 254, 270, 287, 313, 327, 344, 357, 1320, 1492, 1512, 1679
<code>__kernel_backend_scope_begin:n .</code>	173, 393, 421, 434
<code>__kernel_backend_scope_end: ...</code>	83, 110, 146, 173, 232, 250, 264, 280, 307, 321, 337, 353, 365, 416, 430, 449, 551, 1321, 1504, 1515, 1693
<code>\g__kernel_backend_scope_int ...</code>	171, 178, 180, 185, 189, 197, 199, 205
<code>\l__kernel_backend_scope_int ...</code>	171, 177, 190, 196
<code>__kernel_color_backend_stack_-</code>	
<code>init:Nnn</code>	516, 582, 3031
<code>__kernel_color_backend_stack_-</code>	
<code>pop:n</code>	558, 596, 653, 3061
<code>__kernel_color_backend_stack_-</code>	
<code>push:nn</code>	558, 596, 649, 975, 988, 3050, 3087
<code>__kernel_dependency_version_-</code>	
<code>check:Nn</code>	1
<code>__kernel_dependency_version_-</code>	
<code>check:nn</code>	27, 29
<code>__kernel_kern:n</code>	2148, 2150, 2379, 2383, 2386, 2390, 2904, 2912, 2915, 2931
<code>\c__kernel_sys_dvipdfmx_version_-</code>	
<code>int</code>	151, 516, 558, 656, 958, 1000, 2842, 2849, 2865, 3095
M	
<code>\MessageBreak</code>	40
mode commands:	
<code>\mode_if_horizontal:TF</code> ...	2295, 2302
<code>\mode_if_math:TF</code>	2193
O	
<code>\oddsidemargin</code>	2217

opacity internal commands:

[__opacity_backend:nn](#) [3126](#)
[__opacity_backend:nnn](#) [2981](#)
[__opacity_backend_fill:n](#)
 [2981](#), [3062](#), [3126](#)
[__opacity_backend_fill_stroke:nn](#)
 [3064](#), [3070](#), [3074](#), [3092](#), [3106](#)
[\l__opacity_backend_fill_tl](#)
 .. [3036](#), [3045](#), [3071](#), [3079](#), [3099](#), [3111](#)
[__opacity_backend_fillstroke:nn](#)
 [3062](#)
[__opacity_backend_reset:](#) [3038](#), [3089](#)
[__opacity_backend_select:n](#)
 [2981](#), [3038](#), [3095](#), [3126](#)
[__opacity_backend_select_aux:n](#) .
 [2981](#), [3038](#), [3077](#), [3097](#), [3109](#)
[\c__opacity_backend_stack_int](#) ...
 [3027](#), [3050](#), [3061](#), [3087](#)
[__opacity_backend_stroke:n](#)
 [2981](#), [3062](#), [3126](#)
[\l__opacity_backend_stroke_tl](#) ...
 .. [3036](#), [3046](#), [3066](#), [3080](#), [3100](#), [3112](#)

P

pdf commands:

[\pdf_object_if_exist:nTF](#) [885](#)
[\pdf_object_new:nn](#) [887](#)
[\pdf_object_ref:n](#) [900](#)
[\pdf_object_ref_last:](#)
 [864](#), [875](#), [934](#), [945](#)
[\pdf_object_unnamed_write:nn](#) ...
 [852](#), [881](#), [907](#)
[\pdf_object_write:nn](#) [888](#)

pdf internal commands:

[__pdf_backend:n](#) [2755](#),
[2759](#), [2761](#), [2787](#), [2792](#), [2801](#), [2824](#),
[2846](#), [2862](#), [2875](#), [2907](#), [2908](#), [2918](#)
[__pdf_backend_annotation:nnnn](#) ..
 [2135](#), [2443](#), [2820](#)
[__pdf_backend_annotation_-](#)
 aux:nnnn [2137](#), [2140](#)
[\g__pdf_backend_annotation_int](#) ..
 .. [2134](#), [2154](#), [2164](#), [2819](#), [2823](#), [2834](#)
[__pdf_backend_annotation_last:](#) .
 [2163](#), [2456](#), [2833](#)
[__pdf_backend_bdc:nn](#)
 [2437](#), [2749](#), [2952](#), [2974](#)
[__pdf_backend_catalog_gput:nn](#) ..
 [2032](#), [2549](#), [2758](#), [2958](#)
[__pdf_backend_compress_objects:n](#)
 [2403](#), [2670](#), [2933](#), [2968](#)
[__pdf_backend_compresslevel:n](#) ..
 [2403](#), [2670](#), [2933](#), [2968](#)

[\l__pdf_backend_content_box](#) [2132](#),
[2196](#), [2220](#), [2223](#), [2225](#), [2254](#), [2265](#)
[__pdf_backend_destination:nn](#) ...
 [2344](#), [2512](#), [2873](#)
[__pdf_backend_destination:nnnn](#) .
 [2344](#), [2512](#), [2873](#)
[__pdf_backend_destination_-](#)
 aux:nnnn [2344](#), [2873](#)
[__pdf_backend_emc:](#)
 [2437](#), [2749](#), [2952](#), [2974](#)
[__pdf_backend_info_gput:nn](#)
 [2032](#), [2549](#), [2758](#), [2958](#)
[__pdf_backend_link:nw](#) [2174](#)
[__pdf_backend_link_aux:nw](#) ... [2174](#)
[__pdf_backend_link_begin:n](#) .. [2836](#)
[__pdf_backend_link_begin:nnnw](#) [2467](#)
[__pdf_backend_link_begin:nw](#) ...
 [2176](#), [2180](#), [2181](#)
[__pdf_backend_link_begin_aux:nw](#)
 [2184](#), [2186](#)
[__pdf_backend_link_begin_-](#)
 goto:nnw [2174](#), [2467](#), [2836](#)
[__pdf_backend_link_begin_-](#)
 user:nnw [2174](#), [2467](#), [2836](#)
[\g__pdf_backend_link_bool](#)
 [2169](#), [2183](#), [2188](#), [2203](#), [2241](#)
[\g__pdf_backend_link_dict_tl](#) ...
 [2166](#), [2191](#), [2236](#)
[__pdf_backend_link_end:](#)
 [2174](#), [2467](#), [2836](#)
[__pdf_backend_link_end_aux:](#) . [2174](#)
[\g__pdf_backend_link_int](#)
 [2165](#), [2231](#),
[2235](#), [2336](#), [2835](#), [2844](#), [2852](#), [2868](#)
[__pdf_backend_link_last:](#)
 [2335](#), [2491](#), [2863](#)
[__pdf_backend_link_margin:n](#) ...
 [2337](#), [2502](#), [2871](#)
[\g__pdf_backend_link_math_bool](#) ..
 [2168](#), [2194](#), [2195](#), [2198](#), [2208](#)
[__pdf_backend_link_minima:](#) .. [2174](#)
[__pdf_backend_link_outerbox:n](#) [2174](#)
[\g__pdf_backend_link_sf_int](#)
 [2167](#), [2293](#), [2304](#), [2305](#)
[__pdf_backend_link_sf_restore:](#) [2174](#)
[__pdf_backend_link_sf_save:](#) . [2174](#)
[\l__pdf_backend_model_box](#) . [2133](#),
[2213](#), [2245](#), [2253](#), [2264](#), [2279](#), [2281](#)
[__pdf_backend_objcompresslevel:n](#)
 [2670](#)
[\g__pdf_backend_object_int](#)
 [2036](#), [2040](#), [2043](#),
[2109](#), [2112](#), [2125](#), [2129](#), [2153](#), [2154](#),

2157, 2230, 2231, 2762, 2766, 2769, 2809, 2811, 2816, 2822, 2823, 2826	pdf.baselineskip 2174, 3464
__pdf_backend_object_last: 2128, 2648, 2815, 2960	pdf.bordertracking 3222
__pdf_backend_object_new:nn 2038, 2570, 2764, 2960	pdf.bordertracking.begin 3222
__pdf_backend_object_now:nn 2107, 2622, 2807, 2960	pdf.bordertracking.continue 3222
\g__pdf_backend_object_prop 2036, 2044, 2055, 2065, 2569, 2587, 2603, 2762, 2770, 2777	pdf.bordertracking.end 3222
__pdf_backend_object_ref:n 2038, 2052, 2066, 2570, 2764, 2783, 2960	pdf.bordertracking.endpage 3222
__pdf_backend_object_write:nn 2048, 2591, 2774, 2960	pdf.breaklink 3360
__pdf_backend_object_write:nnn 2774	pdf.breaklink.write 3360
__pdf_backend_object_write_- array:nn 2048, 2774	pdf.brokenlink.dict 3222
__pdf_backend_object_write_- dict:nn 2048, 2774	pdf.brokenlink.rect 3222
__pdf_backend_object_write_- fstream:nn 2048, 2774	pdf.brokenlink.skip 3222
__pdf_backend_object_write_- fstream:nnn 2082, 2084	pdf.count 3360
__pdf_backend_object_write_- stream:nn 2048, 2774	pdf.currentrect 3360
__pdf_backend_object_write_- stream:nnn 2048	pdf.cvs 3144
__pdf_backend_object_write_- stream:nnnn 2774	pdf.dest.anchor 3187
__pdf_backend_pageobject_ref:n 2130, 2659, 2817, 2960	pdf.dest.point 3187
__pdf_backend_pdfmark:n 2029, 2033, 2035, 2050, 2071, 2076, 2110, 2155, 2347, 2391, 2438, 2440	pdf.dest.x 3187
__pdf_backend_version_major: 2429, 2435, 2726, 2942, 2943, 2950, 2972	pdf.dest.y 3187
__pdf_backend_version_major_- gset:n 2427, 2698, 2940, 2970	pdf.dest2device 3187
__pdf_backend_version_minor: 2433, 2435, 2726, 2947, 2948, 2950, 2972	pdf.dev.x 3187
__pdf_backend_version_minor_- gset:n 2427, 2698, 2940, 2970	pdf.dev.y 3187
\l__pdf_breaklink_pdfmark_tl 2170, 2238, 2330	pdf.dvi.pt 3144
__pdf_breaklink_postscript:n 2172, 2222, 2224, 2331	pdf.globaldict 3141
__pdf_breaklink_usebox:N 2173, 2223, 2332	pdf.leftboundary 3222
__pdf_exp_not_i:nn . 2591, 2637, 2642	pdf.link.dict 2174
__pdf_exp_not_ii:nn 2591, 2638, 2643	pdf.linkdp.pad 2174, 3148
\l__pdf_internal_box 2027	pdf.linkht.pad 2174, 3148
	pdf.linkmargin 3148
	pdf.llx 2174, 3151
	pdf.lly 2174, 3151
	pdf.originx 3222
	pdf.originy 3222
	pdf.outerbox 2174, 3464
	pdf.pdfmark 3464
	pdf.pdfmark.dict 3464
	pdf.pdfmark.good 3464
	pdf.pt.dvi 3144
	pdf.rect 3151
	pdf.rect.ht 3144
	pdf.rightboundary 3222
	pdf.save.linkll 3151
	pdf.save.linkur 3151
	pdf.save.ll 3151
	pdf.save.ur 3151
	pdf.tmpa 3187
	pdf.tmpb 3187
	pdf.tmpc 3187
	pdf.tmpd 3187
	pdf.urx 3151
	pdf.ury 2174, 3151
	pdfmanagement commands:
	\pdfmanagement_add:nnn 872, 942, 3033, 3047, 3081, 3084, 3101, 3113, 3116

tl commands:		501, 504, 647, 648, 974, 987, 1719,
\c_space_tl	295, 300, 303, 532,	1734, 1807, 2171, 2330, 3045, 3046,
	782, 989, 1552, 1705, 1706, 1707,	3079, 3080, 3099, 3100, 3111, 3112
	1708, 1844, 1845, 1846, 1847, 1892,	\tl_to_str:n 2042,
	1895, 1897, 1898, 1899, 1900, 1972,	2047, 2580, 2590, 2601, 2768, 2773
	1974, 2003, 2004, 2005, 2006, 2236,	\tl_use:N 814, 893
	2465, 2500, 2657, 2668, 2826, 2853	token commands:
\tl_clear:N	1718, 1726, 1732,	\c_math_toggle_token 2199, 2209
	1828, 1834, 1921, 1927, 1985, 1991	
\tl_gclear:N	1590, 1626	U
\tl_gset:Nn	1549, 2191	use commands:
\tl_if_blank:nTF		\use:N 43, 2064, 2124, 2782, 2810
	533, 592, 730, 747, 754, 772, 856, 952	\use:n 59, 465, 501, 524,
\tl_if_empty:NTF	1552, 1721, 1756,	870, 940, 1064, 1074, 1087, 1260,
	1764, 1862, 1866, 1893, 1908, 1942	1384, 1449, 1461, 1473, 1631, 1949
\tl_if_empty:nTF	1646	\use_none:n 1646, 1648, 2308
\tl_if_empty_p:N	1752, 1905	V
\tl_if_head_is_space:nTF	461	\value 2216
\tl_new:N	637,	vbox commands:
	638, 1556, 1714, 2166, 2170, 3036, 3037	\vbox_set:Nn 2316
\tl_put_right:Nn	2312	\vbox_to_zero:n 2377, 2384, 2902, 2913
\tl_set:Nn	463, 475, 491, 494, 497,	\vbox_unpack_drop:N 2324