

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

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

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

```

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

```

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

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

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

```

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

```

(End definition for `__kernel_backend_literal:e`.)

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

```

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

```

(End definition for `__kernel_backend_first_shipout:n`.)

1.1 dvips backend

```

60 ⟨*dvips⟩

```

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

```

61 \cs_new_protected:Npn __kernel_backend_literal_postscript:n #1
62 { __kernel_backend_literal:n { ps:: #1 } }
63 \cs_generate_variant:Nn __kernel_backend_literal_postscript:n { x }

```

(End definition for `_kernel_backend_literal_postscript:n`.)

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

```

64 \cs_new_protected:Npn \_kernel_backend_postscript:n #1
65   { \_kernel_backend_literal:n { ps: SDict ~ begin ~ #1 ~ end } }
66 \cs_generate_variant:Nn \_kernel_backend_postscript:n { x }

```

(End definition for `_kernel_backend_postscript:n`.)

PostScript for the header: a small saving but makes the code clearer. This is held until the start of shipout such that a document with no actual output does not write anything.

```

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

```

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

```

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

```

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

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

```

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

```

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

```

87 </dvips>

```

1.2 LuaTeX and pdfTeX backends

88 `<*luatex | pdftex>`

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

`_kernel_backend_literal_pdf:n`
`_kernel_backend_literal_pdf:x`

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

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

(End definition for `_kernel_backend_literal_pdf:n`.)

`_kernel_backend_literal_page:n`

Page literals are pretty simple. To avoid an expansion, we write out by hand.

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

(End definition for `_kernel_backend_literal_page:n`.)

`_kernel_backend_scope_begin:`

Higher-level interfaces for saving and restoring the graphic state.

`_kernel_backend_scope_end:`

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

```

126 </pdfTeX>
127 }

```

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

`_kernel_backend_matrix:n` Here the appropriate function is set up to insert an affine matrix into the PDF. With pdfTeX and LuaTeX in direct PDF output mode there is a primitive for this, which only needs the rotation/scaling/skew part.

```

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

```

(End definition for `_kernel_backend_matrix:n.`)

```

139 </luatex | pdfTeX>

```

1.3 dvipdfmx backend

```

140 <*dvipdfmx | xetex>

```

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

__box_backend_rotate_aux:Nn

\l__box_backend_cos_fp

\l__box_backend_sin_fp

```

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.

```

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` 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 $\langle *dvipdfmx | luatex | pdftex | xetex \rangle$

$\backslash l_color_backend_fill_tl$
 $\backslash l_color_backend_stroke_tl$

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 $\langle /dvipdfmx | luatex | pdftex | xetex \rangle$

3.3.3 dvipdfmx/X_YTeX

655 $\langle *dvipdfmx | xetex \rangle$

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 $\langle /dvipdfmx | xetex \rangle$

3.4 Separations

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

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

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

```
\g_color_backend_colorant_prop
670 \prop_new:N \g__color_backend_colorant_prop
(End definition for \g__color_backend_colorant_prop.)

\_color_backend_devicen_colorants:n
\_color_backend_devicen_colorants:w
671 \cs_new:Npx \_color_backend_devicen_colorants:n #1
672 {
673   \exp_not:N \tl_if_blank:nF {#1}
674   {
675     \c_space_tl
676     << ~
677     /Colorants ~
678     << ~
679     \exp_not:N \_color_backend_devicen_colorants:w #1 ~
680     \exp_not:N \q_recursion_tail \c_space_tl
681     \exp_not:N \q_recursion_stop
682     >> ~
683   }
684 }
685 }
686 \cs_new:Npn \_color_backend_devicen_colorants:w #1 ~
687 {
688   \quark_if_recursion_tail_stop:n {#1}
689   \prop_if_in:NnT \g__color_backend_colorant_prop {#1}
690   {
691     #1 ~
692     \prop_item:Nn \g__color_backend_colorant_prop {#1} ~
693   }
694   \_color_backend_devicen_colorants:w
695 }
(End definition for \_color_backend_devicen_colorants:n and \_color_backend_devicen_colorants:w.)
696 < /dvipdfmx | luatex | pdftex | xetex | dvips >
697 < *dvips >

\_color_backend_select_separation:nn
\_color_backend_select_devicen:nn
698 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
699 { \_color_backend_select:n { separation ~ #1 ~ #2 } }
700 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
(End definition for \_color_backend_select_separation:nn and \_color_backend_select_devicen:nn.)
```

```

\__color_backend_separation_init:nnnnn
\__color_backend_separation_init:nxxnn
\__color_backend_separation_init_aux:nnnnnn
lor_backend_separation_init_/DeviceCMYK:nnn
lor_backend_separation_init_/DeviceGray:nnn
lor_backend_separation_init_/DeviceRGB:nnn
\__color_backend_separation_init_Device:Nn
\__color_backend_separation_init:nnn
\__color_backend_separation_init_count:n
\__color_backend_separation_init_count:w
\__color_backend_separation_init:nnnn
\__color_backend_separation_init:w
\__color_backend_separation_init:n
\__color_backend_separation_init:nw
\__color_backend_separation_init_CIELAB:nnn

```

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.

```

701 \cs_new_protected:Npx \__color_backend_separation_init:nnnnn #1#2#3#4#5
702 {
703   \bool_if:NT \g__kernel_backend_header_bool
704   {
705     \exp_args:Nx \__kernel_backend_first_shipout:n
706     {
707       \exp_not:N \__color_backend_separation_init_aux:nnnnnn
708       { \exp_not:N \int_use:N \g__color_model_int }
709       {#1} {#2} {#3} {#4} {#5}
710     }
711     \prop_gput:Nxx \exp_not:N \g__color_backend_colorant_prop
712     { / \exp_not:N \str_convert_pdfname:n {#1} }
713     {
714       << ~
715       /setcolorspace ~ {} ~
716       >> ~ begin ~
717       color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
718       end
719     }
720   }
721 }
722 \cs_generate_variant:Nn \__color_backend_separation_init:nnnnn { nxx }
723 \cs_new_protected:Npn \__color_backend_separation_init_aux:nnnnnn #1#2#3#4#5#6
724 {
725   \__kernel_backend_literal:e
726   {
727     !
728     TeXDict ~ begin ~
729     /color #1
730     {
731       [ ~
732       /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
733       [ ~ #3 ~ ] ~
734       {
735         \cs_if_exist_use:cF { __color_backend_separation_init_ #3 :nnn }
736         { \__color_backend_separation_init:nnn }
737         {#4} {#5} {#6}
738       }
739       ] ~ setcolorspace
740     } ~ def ~
741     end
742   }
743 }
744 \cs_new:cpn { __color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
745 { \__color_backend_separation_init_Device:Nn 4 {#3} }
746 \cs_new:cpn { __color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
747 { \__color_backend_separation_init_Device:Nn 1 {#3} }
748 \cs_new:cpn { __color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
749 { \__color_backend_separation_init_Device:Nn 2 {#3} }
750 \cs_new:Npn \__color_backend_separation_init_Device:Nn #1#2

```

```

751 {
752   #2 ~
753   \prg_replicate:nn {#1}
754   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
755   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
756 }

```

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.

```

757 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3
758 {
759   \exp_args:Ne \__color_backend_separation_init:nnnn
760   { \__color_backend_separation_init_count:n {#2} }
761   {#1} {#2} {#3}
762 }
763 \cs_new:Npn \__color_backend_separation_init_count:n #1
764 { \int_eval:n { 0 \__color_backend_separation_init_count:w #1 ~ \s__color_stop } }
765 \cs_new:Npn \__color_backend_separation_init_count:w #1 ~ #2 \s__color_stop
766 {
767   +1
768   \tl_if_blank:nF {#2}
769   { \__color_backend_separation_init_count:w #2 \s__color_stop }
770 }

```

Now we implement the algorithm. In the terms in the PostScript manual, we have $N = 1$ 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.

```

771 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
772 {
773   \__color_backend_separation_init:w #3 ~ \s__color_stop #4 ~ \s__color_stop
774   \prg_replicate:nn {#1}
775   {
776     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
777     \int_eval:n { 3 * #1 } ~ index ~ mul ~
778     2 ~ index ~ add ~
779     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
780   }
781   \int_step_function:nnnN {#1} { -1 } { 1 }
782   \__color_backend_separation_init:n
783   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
784   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
785   \tl_if_blank:nF {#2}
786   { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
787 }

```

```

788 \cs_new:Npn \__color_backend_separation_init:w
789   #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
790   {
791     #1 ~ #3 ~ 0 ~
792     \tl_if_blank:nF {#2}
793     { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }
794   }
795 \cs_new:Npn \__color_backend_separation_init:n #1
796   { \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.

```

797 \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s__color_stop
798   {
799     #2 ~ #3 ~
800     2 ~ index ~ 2 ~ index ~ lt ~
801     { ~ pop ~ exch ~ pop ~ } ~
802     { ~
803       2 ~ index ~ 1 ~ index ~ gt ~
804       { ~ exch ~ pop ~ exch ~ pop ~ } ~
805       { ~ pop ~ pop ~ } ~
806       ifelse ~
807     }
808     ifelse ~
809     #1 ~ 1 ~ roll ~
810     \tl_if_blank:nF {#4}
811     { \__color_backend_separation_init:nw {#1} #4 \s__color_stop }
812   }

```

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.

```

813 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
814   {
815     \__color_backend_separation_init:nxxxnn
816     {#2}
817     {
818       /CIEBasedABC ~
819       << ~
820       /RangeABC ~ [ ~ \c__color_model_range_CIELAB_tl \c_space_tl ] ~
821       /DecodeABC ~
822       [ ~
823         { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
824         { ~ 500 ~ div ~ } ~ bind ~
825         { ~ 200 ~ div ~ } ~ bind ~
826       ] ~
827       /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
828       /DecodeLMN ~
829       [ ~
830         { ~
831           dup ~ 6 ~ 29 ~ div ~ ge ~
832           { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
833           { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
834           ifelse ~
835           0.9505 ~ mul ~

```

```

836         } ~ bind ~
837         { ~
838             dup ~ 6 ~ 29 ~ div ~ ge ~
839             { ~ dup ~ dup ~ mul ~ mul ~ } ~
840             { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
841             ifelse ~
842         } ~ bind ~
843         { ~
844             dup ~ 6 ~ 29 ~ div ~ ge ~
845             { ~ dup ~ dup ~ mul ~ mul ~ } ~
846             { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
847             ifelse ~
848             1.0890 ~ mul ~
849         } ~ bind
850     ] ~
851     /WhitePoint ~
852     [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
853     >>
854 }
855 { \c__color_model_range_CIELAB_tl }
856 { 100 ~ 0 ~ 0 }
857 {#3}
858 }

```

(End definition for _color_backend_separation_init:nnnnn and others.)

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

```

859 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
860 {
861     \_kernel_backend_literal:e
862     {
863         !
864         TeXDict ~ begin ~
865         /color \int_use:N \g__color_model_int
866         {
867             [ ~
868             /DeviceN ~
869             [ ~ #1 ~ ] ~
870             #2 ~
871             { ~ #3 ~ } ~
872             \_color_backend_devicen_colorants:n {#1}
873             ] ~ setcolorspace
874         } ~ def ~
875     end
876 }
877 }

```

(End definition for _color_backend_devicen_init:nnn.)

```

878 </dvips>
879 <*dvisvgm>

```

_color_backend_select_separation:nn No support at present.

```

\_color_backend_select_devicen:nn
880 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2 { }
881 \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.

`_color_backend_separation_init:nnnnn`
`_color_backend_separation_init_CIELAB:nnn`

```
882 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }
883 \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:nnn`.)

```
884 </dvisvgm>
885 <*dvipdfmx | luatex | pdfTeX | xetex>
```

`_color_backend_select_separation:nn`
`_color_backend_select_devicen:nn`

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.

```
886 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
887 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
888 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
```

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

`_color_backend_separation_init:nnnnn`
`_color_backend_separation_init:nn`
`_color_backend_separation_init_CIELAB:nnn`

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.

```
889 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5
890 {
891   \pdf_object_unnamed_write:nx { dict }
892   {
893     /FunctionType ~ 2
894     /Domain ~ [0 ~ 1]
895     \tl_if_blank:nF {#3} { /Range ~ [#3] }
896     /C0 ~ [#4] ~
897     /C1 ~ [#5] /N ~ 1
898   }
899   \exp_args:Nx \_color_backend_separation_init:nn
900   { \str_convert_pdfname:n {#1} } {#2}
901   \bool_lazy_and:nnT
902   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
903   { \pdfmanagement_if_active_p: }
904   {
905     \use:x
906     {
907       \pdfmanagement_add:nnn
908       { Page / Resources / ColorSpace }
909       { color \int_use:N \g__color_model_int }
910       { \pdf_object_ref_last: }
911     }
912   }
913 }
914 \cs_new_protected:Npn \_color_backend_separation_init:nn #1#2
915 {
916   \pdf_object_unnamed_write:nx { array }
917   { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
918   \prop_gput:Nnx \g__color_backend_colorant_prop { /#1 }
919   { \pdf_object_ref_last: }
920 }
```

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

```

921 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
922 {
923   \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
924   {
925     \pdf_object_new:nn { __color_illuminant_CIELAB_ #1 } { array }
926     \pdf_object_write:nx { __color_illuminant_CIELAB_ #1 }
927     {
928       /Lab ~
929       <<
930       /WhitePoint ~
931       [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
932       /Range ~ [ \c__color_model_range_CIELAB_tl ]
933       >>
934     }
935   }
936   \__color_backend_separation_init:nnnnn
937   {#2}
938   { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
939   { \c__color_model_range_CIELAB_tl }
940   { 100 ~ 0 ~ 0 }
941   {#3}
942 }

```

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

__color_backend_devicen_init:nnn
 __color_backend_devicen_init:w

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

```

943 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
944 {
945   \pdf_object_unnamed_write:nx { stream }
946   {
947     {
948       /FunctionType ~ 4 ~
949       /Domain ~
950       [ ~
951         \prg_replicate:nn
952         { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
953         { 0 ~ 1 ~ }
954       ] ~
955       /Range ~
956       [ ~
957         \str_case:nn {#2}
958         {
959           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
960           { /DeviceGray } { 0 ~ 1 }
961           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
962         } ~
963       ]
964     }
965     { {#3} }
966   }

```

```

967 \pdf_object_unnamed_write:nx { array }
968 {
969   /DeviceN ~
970   [ ~ #1 ~ ] ~
971   #2 ~
972   \pdf_object_ref_last:
973   \__color_backend_devicen_colorants:n {#1}
974 }
975 \bool_lazy_and:nnT
976 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
977 { \pdfmanagement_if_active_p: }
978 {
979   \use:x
980   {
981     \pdfmanagement_add:nnn
982     { Page / Resources / ColorSpace }
983     { color \int_use:N \g__color_model_int }
984     { \pdf_object_ref_last: }
985   }
986 }
987 }
988 \cs_new:Npn \__color_backend_devicen_init:w #1 ~ #2 \s__color_stop
989 {
990   + 1
991   \tl_if_blank:nF {#2}
992   { \__color_backend_devicen_init:w #2 \s__color_stop }
993 }

(End definition for \__color_backend_devicen_init:nnn and \__color_backend_devicen_init:w.)

994 </dvipdfmx | luatex | pdftex | xetex>
995 <*dvipdfmx | xetex>

```

__color_backend_select_separation:nn
 __color_backend_select_devicen: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.

```

996 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
997 {
998   \cs_gset_protected:Npn \__color_backend_select_separation:nn #1#2 { }
999   \cs_gset_eq:NN \__color_backend_select_devicen:nn
1000   \__color_backend_select_separation:nn
1001 }

```

(End definition for __color_backend_select_separation:nn and __color_backend_select_devicen:nn.)

```
1002 </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.

```
1003 <*dvipdfmx | luatex | pdftex | xetex>
```

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

```

1004 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1005 { \_color_backend_fill:n { #1 ~ k } }
1006 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1007 { \_color_backend_fill:n { #1 ~ g } }
1008 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1009 { \_color_backend_fill:n { #1 ~ rg } }
1010 \cs_new_protected:Npn \_color_backend_fill:n #1
1011 {
1012   \tl_set:Nn \l__color_backend_fill_tl {#1}
1013   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1014   { #1 ~ \l__color_backend_stroke_tl }
1015   \group_insert_after:N \_color_backend_reset:
1016 }
1017 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1018 { \_color_backend_stroke:n { #1 ~ K } }
1019 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1020 { \_color_backend_stroke:n { #1 ~ G } }
1021 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1022 { \_color_backend_stroke:n { #1 ~ RG } }
1023 \cs_new_protected:Npn \_color_backend_stroke:n #1
1024 {
1025   \tl_set:Nn \l__color_backend_stroke_tl {#1}
1026   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1027   { \l__color_backend_fill_tl \c_space_tl #1 }
1028   \group_insert_after:N \_color_backend_reset:
1029 }

```

(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
1030 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1031 { \_color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
1032 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1033 { \_color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
1034 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1035 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

```

1036 </dvipdfmx | luatex | pdftex | xetex>
1037 <*dvipdfmx | xetex>

```

`_color_backend_fill_cmyk:n` Deal with older (x)dvipdfmx.

```

\_color_backend_fill_gray:n
\_color_backend_fill_rgb:n
\_color_backend_reset:
\_color_backend_stroke:n
\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
1038 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
1039 {
1040   \cs_gset_protected:Npn \_color_backend_fill_cmyk:n #1
1041   {
1042     \__kernel_backend_literal:n { pdf: bc ~ [#1] }
1043     \group_insert_after:N \_color_backend_reset:
1044   }

```

```

1045 \cs_gset_eq:NN \_color_backend_fill_gray:n \_color_backend_fill_cmyk:n
1046 \cs_gset_eq:NN \_color_backend_fill_rgb:n \_color_backend_fill_cmyk:n
1047 \cs_gset_protected:Npn \_color_backend_reset:
1048 { \_kernel_backend_literal:n { pdf: ec } }
1049 \cs_gset_protected:Npn \_color_backend_stroke:n #1
1050 { \_kernel_backend_literal:n {#1} }
1051 \cs_gset_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1052 \cs_gset_eq:NN \_color_backend_fill_devicen:nn
1053 \_color_backend_fill_separation:nn
1054 \cs_gset_eq:NN \_color_backend_stroke_separation:nn
1055 \_color_backend_fill_separation:nn
1056 \cs_gset_eq:NN \_color_backend_stroke_devicen:nn
1057 \_color_backend_stroke_separation:nn
1058 }

```

(End definition for _color_backend_fill_cmyk:n and others.)

```

1059 </dvipdfmx | xetex>

```

```

1060 <*dvips>

```

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

```

\_color_backend_fill_gray:n
\_color_backend_fill_rgb:n
\_color_backend_fill:n
  \_color_backend_stroke_cmyk:n
  \_color_backend_stroke_gray:n
  \_color_backend_stroke_rgb:n
1061 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1062 { \_color_backend_fill:n { cmyk ~ #1 } }
1063 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1064 { \_color_backend_fill:n { gray ~ #1 } }
1065 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1066 { \_color_backend_fill:n { rgb ~ #1 } }
1067 \cs_new_protected:Npn \_color_backend_fill:n #1
1068 {
1069   \_kernel_backend_literal:n { color~push~ #1 }
1070   \group_insert_after:N \_color_backend_reset:
1071 }
1072 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1073 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1074 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1075 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1076 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1077 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }

```

(End definition for _color_backend_fill_cmyk:n and others.)

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
1078 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1079 { \_color_backend_fill:n { separation ~ #1 ~ #2 } }
1080 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1081 { \_kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1082 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1083 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

(End definition for _color_backend_fill_separation:nn and others.)

```

1084 </dvips>

```

```

1085 <*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 1086 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
\_color_backend_fill_rgb:n 1087 { \_color_backend_fill:n { cmyk ~ #1 } }
\_color_backend_fill:n 1088 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1089 { \_color_backend_fill:n { gray ~ #1 } }
1090 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1091 { \_color_backend_fill:n { rgb ~ #1 } }
1092 \cs_new_protected:Npn \_color_backend_fill:n #1
1093 {
1094 \_kernel_backend_literal:n { color~push~ #1 }
1095 \group_insert_after:N \_color_backend_reset:
1096 }

```

(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 1097 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
\_color_backend_stroke_gray:n 1098 { \_color_backend_cmyk:w #1 \s__color_stop }
\_color_backend_stroke_gray_aux:n 1099 \cs_new_protected:Npn \_color_backend_stroke_cmyk:w
\_color_backend_stroke_rgb:n 1100 #1 ~ #2 ~ #3 ~ #4 \s__color_stop
\_color_backend_stroke_rgb:w 1101 {
\_color_backend:nnn 1102 \use:x
1103 {
1104 \_color_backend:nnn
1105 { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1106 { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1107 { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }
1108 }
1109 }
1110 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1111 {
1112 \use:x
1113 {
1114 \_color_backend_stroke_gray_aux:n
1115 { \fp_eval:n { 100 * (#1) } }
1116 }
1117 }
1118 \cs_new_protected:Npn \_color_backend_stroke_gray_aux:n #1
1119 { \_color_backend:nnn {#1} {#1} {#1} }
1120 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1121 { \_color_backend_rgb:w #1 \s__color_stop }
1122 \cs_new_protected:Npn \_color_backend_stroke_rgb:w
1123 #1 ~ #2 ~ #3 \s__color_stop
1124 {
1125 \use:x
1126 {
1127 \_color_backend:nnn
1128 { \fp_eval:n { 100 * (#1) } }
1129 { \fp_eval:n { 100 * (#2) } }
1130 { \fp_eval:n { 100 * (#3) } }
1131 }
1132 }
1133 \cs_new_protected:Npx \_color_backend:nnn #1#2#3

```

```

1134 {
1135   \_kernel_backend_scope:n
1136   {
1137     stroke =
1138     "
1139     rgb
1140     (
1141       #1 \c_percent_str ,
1142       #2 \c_percent_str ,
1143       #3 \c_percent_str
1144     )
1145     "
1146   }
1147 }

```

(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
1148 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1149 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2 { }
1150 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1151 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

(End definition for _color_backend_fill_separation:nn and others.)

```

1152 </dvisvgm>
1153 </package>

```

4 l3backend-draw Implementation

```

1154 <*package>
1155 <@@=draw>

```

4.1 dvips backend

```

1156 <*dvips>

```

```

\_draw_backend_literal:n The same as literal PostScript: same arguments about positioning apply her.
\_draw_backend_literal:x
1157 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
1158 \cs_generate_variant:Nn \_draw_backend_literal:n { x }

```

(End definition for _draw_backend_literal:n.)

```

\_draw_backend_begin: The ps::[begin] special here deals with positioning but allows us to continue on to a
  \_draw_backend_end: matching ps::[end]: contrast with ps:, which positions but where we can't split mater-
                        ial 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 transla-
                        tions (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.

```

```

1159 \cs_new_protected:Npn \_draw_backend_begin:
1160 {

```

```

1161     \__kernel_backend_literal:n { ps::[begin] }
1162     \__draw_backend_literal:n { @beginspecial }
1163   }
1164   \cs_new_protected:Npn \__draw_backend_end:
1165   {
1166     \__draw_backend_literal:n { @endspecial }
1167     \__kernel_backend_literal:n { ps::[end] }
1168   }

```

(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
 __draw_backend_scope_end: the graphic state has to be sent to the stack.

```

1169   \cs_new_protected:Npn \__draw_backend_scope_begin:
1170   { \__draw_backend_literal:n { save } }
1171   \cs_new_protected:Npn \__draw_backend_scope_end:
1172   { \__draw_backend_literal:n { restore } }

```

(End definition for __draw_backend_scope_begin: and __draw_backend_scope_end:.)

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

```

1173   \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1174   {
1175     \__draw_backend_literal:x
1176     {
1177       \dim_to_decimal_in_bp:n {#1} ~
1178       \dim_to_decimal_in_bp:n {#2} ~ moveto
1179     }
1180   }
1181   \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1182   {
1183     \__draw_backend_literal:x
1184     {
1185       \dim_to_decimal_in_bp:n {#1} ~
1186       \dim_to_decimal_in_bp:n {#2} ~ lineto
1187     }
1188   }
1189   \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1190   {
1191     \__draw_backend_literal:x
1192     {
1193       \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1194       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1195       moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1196     }
1197   }
1198   \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1199   {
1200     \__draw_backend_literal:x
1201     {

```



```

1202         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1203         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1204         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1205         curveto
1206     }
1207 }

```

(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
1208 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1209 { \bool_gset_true:N \g__draw_draw_eor_bool }
1210 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1211 { \bool_gset_false:N \g__draw_draw_eor_bool }
1212 \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
1213 \cs_new_protected:Npn \__draw_backend_closepath:
1214 { \__draw_backend_literal:n { closepath } }
1215 \cs_new_protected:Npn \__draw_backend_stroke:
1216 {
1217     \__draw_backend_literal:n { gsave }
1218     \__draw_backend_literal:n { color.sc }
1219     \__draw_backend_literal:n { stroke }
1220     \__draw_backend_literal:n { grestore }
1221     \bool_if:NT \g__draw_draw_clip_bool
1222     {
1223         \__draw_backend_literal:x
1224         {
1225             \bool_if:NT \g__draw_draw_eor_bool { eo }
1226             clip
1227         }
1228     }
1229     \__draw_backend_literal:n { newpath }
1230     \bool_gset_false:N \g__draw_draw_clip_bool
1231 }
1232 \cs_new_protected:Npn \__draw_backend_closestroke:
1233 {
1234     \__draw_backend_closepath:
1235     \__draw_backend_stroke:
1236 }
1237 \cs_new_protected:Npn \__draw_backend_fill:
1238 {
1239     \__draw_backend_literal:x
1240     {
1241         \bool_if:NT \g__draw_draw_eor_bool { eo }

```

```

1242         fill
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 }
1255 \cs_new_protected:Npn \__draw_backend_fillstroke:
1256 {
1257     \__draw_backend_literal:x
1258     {
1259         \bool_if:NT \g__draw_draw_eor_bool { eo }
1260         fill
1261     }
1262     \__draw_backend_literal:n { gsave }
1263     \__draw_backend_literal:n { color.sc }
1264     \__draw_backend_literal:n { stroke }
1265     \__draw_backend_literal:n { grestore }
1266     \bool_if:NT \g__draw_draw_clip_bool
1267     {
1268         \__draw_backend_literal:x
1269         {
1270             \bool_if:NT \g__draw_draw_eor_bool { eo }
1271             clip
1272         }
1273     }
1274     \__draw_backend_literal:n { newpath }
1275     \bool_gset_false:N \g__draw_draw_clip_bool
1276 }
1277 \cs_new_protected:Npn \__draw_backend_clip:
1278 { \bool_gset_true:N \g__draw_draw_clip_bool }
1279 \bool_new:N \g__draw_draw_clip_bool
1280 \cs_new_protected:Npn \__draw_backend_discardpath:
1281 {
1282     \bool_if:NT \g__draw_draw_clip_bool
1283     {
1284         \__draw_backend_literal:x
1285         {
1286             \bool_if:NT \g__draw_draw_eor_bool { eo }
1287             clip
1288         }
1289     }
1290     \__draw_backend_literal:n { newpath }
1291     \bool_gset_false:N \g__draw_draw_clip_bool
1292 }

```

(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_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1293 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1294 {
1295   \__draw_backend_literal:x
1296   {
1297     [
1298       \exp_args:Nf \use:n
1299       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1300     ] ~
1301     \dim_to_decimal_in_bp:n {#2} ~ setdash
1302   }
1303 }
1304 \cs_new:Npn \__draw_backend_dash:n #1
1305 { ~ \dim_to_decimal_in_bp:n {#1} }
1306 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1307 {
1308   \__draw_backend_literal:x
1309   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1310 }
1311 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1312 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1313 \cs_new_protected:Npn \__draw_backend_cap_butt:
1314 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1315 \cs_new_protected:Npn \__draw_backend_cap_round:
1316 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1317 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1318 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1319 \cs_new_protected:Npn \__draw_backend_join_miter:
1320 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1321 \cs_new_protected:Npn \__draw_backend_join_round:
1322 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1323 \cs_new_protected:Npn \__draw_backend_join_bevel:
1324 { \__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.

```

1325 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1326 {
1327   \__draw_backend_literal:n
1328   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1329 }

(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).

```

1330 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1331 {
1332   \__draw_backend_literal:n { @endspecial }
1333   \__draw_backend_literal:n { [end] }
1334   \__draw_backend_literal:n { [begin] }
1335   \__draw_backend_literal:n { save }
1336   \__draw_backend_literal:n { currentpoint }
1337   \__draw_backend_literal:n { currentpoint~translate }
1338   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1339   \__draw_backend_cm:nnnn { #2 } { #3 } { #4 } { #5 }
1340   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1341   \__draw_backend_literal:n { neg~exch~neg~exch~translate }
1342   \__draw_backend_literal:n { [end] }
1343   \hbox_overlap_right:n { \box_use:N #1 }
1344   \__draw_backend_literal:n { [begin] }
1345   \__draw_backend_literal:n { restore }
1346   \__draw_backend_literal:n { [end] }
1347   \__draw_backend_literal:n { [begin] }
1348   \__draw_backend_literal:n { @beginspecial }
1349 }

```

(End definition for `__draw_backend_box_use:Nnnnn`.)

```

1350 </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.

```

1351 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

```

\__draw_backend_literal:n Pass data through using a dedicated interface.
\__draw_backend_literal:x
1352 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1353 \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:
1354 \cs_new_protected:Npn \__draw_backend_begin:
1355 { \__draw_backend_scope_begin: }
1356 \cs_new_protected:Npn \__draw_backend_end:
1357 { \__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:`

```

1358 \cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:
1359 \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`

```

1360 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1361 {
1362   \_draw_backend_literal:x
1363   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1364 }
1365 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1366 {
1367   \_draw_backend_literal:x
1368   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1369 }
1370 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1371 {
1372   \_draw_backend_literal:x
1373   {
1374     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1375     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1376     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1377     c
1378   }
1379 }
1380 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1381 {
1382   \_draw_backend_literal:x
1383   {
1384     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1385     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1386     re
1387   }
1388 }

```

(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`

```

1389 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1390 { \bool_gset_true:N \g__draw_draw_eor_bool }
1391 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1392 { \bool_gset_false:N \g__draw_draw_eor_bool }
1393 \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:`

```

1397 { \_draw_backend_literal:n { S } }
1398 \cs_new_protected:Npn \_draw_backend_closestroke:
1399 { \_draw_backend_literal:n { s } }
1400 \cs_new_protected:Npn \_draw_backend_fill:
1401 {
1402   \_draw_backend_literal:x
1403   { f \bool_if:NT \g__draw_draw_eor_bool * }
1404 }
1405 \cs_new_protected:Npn \_draw_backend_fillstroke:
1406 {
1407   \_draw_backend_literal:x
1408   { B \bool_if:NT \g__draw_draw_eor_bool * }
1409 }
1410 \cs_new_protected:Npn \_draw_backend_clip:
1411 {
1412   \_draw_backend_literal:x
1413   { W \bool_if:NT \g__draw_draw_eor_bool * }
1414 }
1415 \cs_new_protected:Npn \_draw_backend_discardpath:
1416 { \_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:
1417 \cs_new_protected:Npn \_draw_backend_dash_pattern:nn #1#2
1418 {
1419   \_draw_backend_literal:x
1420   {
1421     [
1422       \exp_args:Nf \use:n
1423       { \clist_map_function:nN {#1} \_draw_backend_dash:n }
1424     ] ~
1425     \dim_to_decimal_in_bp:n {#2} ~ d
1426   }
1427 }
1428 \cs_new:Npn \_draw_backend_dash:n #1
1429 { ~ \dim_to_decimal_in_bp:n {#1} }
1430 \cs_new_protected:Npn \_draw_backend_linewidth:n #1
1431 {
1432   \_draw_backend_literal:x
1433   { \dim_to_decimal_in_bp:n {#1} ~ w }
1434 }
1435 \cs_new_protected:Npn \_draw_backend_miterlimit:n #1
1436 { \_draw_backend_literal:x { #1 ~ M } }
1437 \cs_new_protected:Npn \_draw_backend_cap_butt:
1438 { \_draw_backend_literal:n { 0 ~ J } }
1439 \cs_new_protected:Npn \_draw_backend_cap_round:
1440 { \_draw_backend_literal:n { 1 ~ J } }
1441 \cs_new_protected:Npn \_draw_backend_cap_rectangle:
1442 { \_draw_backend_literal:n { 2 ~ J } }
1443 \cs_new_protected:Npn \_draw_backend_join_miter:
1444 { \_draw_backend_literal:n { 0 ~ j } }
1445 \cs_new_protected:Npn \_draw_backend_join_round:
1446 { \_draw_backend_literal:n { 1 ~ j } }

```

```

1447 \cs_new_protected:Npn \__draw_backend_join_bevel:
1448 { \__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!

```

1449 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1450 {
1451   <*luatex | pdftex>
1452   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1453   </luatex | pdftex>
1454   <*dvipdfmx | xetex>
1455   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1456   \__draw_backend_cm_aux:nnnn
1457   </dvipdfmx | xetex>
1458 }
1459 <*dvipdfmx | xetex>
1460 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1461 {
1462   \__kernel_backend_literal:x
1463   {
1464     x:rotate~
1465     \fp_compare:nNnTF {#1} = \c_zero_fp
1466     { 0 }
1467     { \fp_eval:n { round ( -#1 , 5 ) } }
1468   }
1469   \__kernel_backend_literal:x
1470   {
1471     x:scale~
1472     \fp_eval:n { round ( #2 , 5 ) } ~
1473     \fp_eval:n { round ( #3 , 5 ) }
1474   }
1475   \__kernel_backend_literal:x
1476   {
1477     x:rotate~
1478     \fp_compare:nNnTF {#4} = \c_zero_fp
1479     { 0 }
1480     { \fp_eval:n { round ( -#4 , 5 ) } }
1481   }
1482 }
1483 </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.

```

1484 <*dvipdfmx | xetex>
1485 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1486 {
1487   \use:x
1488   {
1489     \__draw_backend_cm_decompose_auxi:nnnnN
1490     { \fp_eval:n { (#1 + #4) / 2 } }
1491     { \fp_eval:n { (#1 - #4) / 2 } }
1492     { \fp_eval:n { (#3 + #2) / 2 } }
1493     { \fp_eval:n { (#3 - #2) / 2 } }
1494   }
1495   #5
1496 }
1497 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1498 {
1499   \use:x
1500   {
1501     \__draw_backend_cm_decompose_auxii:nnnnN
1502     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1503     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1504     { \fp_eval:n { atand ( #3 , #2 ) } }
1505     { \fp_eval:n { atand ( #4 , #1 ) } }
1506   }
1507   #5
1508 }
1509 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1510 {
1511   \use:x
1512   {

```



```

1513 \__draw_backend_cm_decompose_auxiii:nnnnN
1514 { \fp_eval:n { ( #4 - #3 ) / 2 } }
1515 { \fp_eval:n { ( #1 + #2 ) / 2 } }
1516 { \fp_eval:n { ( #1 - #2 ) / 2 } }
1517 { \fp_eval:n { ( #4 + #3 ) / 2 } }
1518 }
1519 #5
1520 }
1521 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1522 {
1523 \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1524 { #5 {#1} {#2} {#3} {#4} }
1525 { #5 {#1} {#3} {#2} {#4} }
1526 }
1527 </dviptdpmx | xetex>

```

(End definition for __draw_backend_cm_decompose:nnnnN and others.)

__draw_backend_box_use:Nnnnn

Inserting a T_EX box transformed to the requested position and using the current matrix is done using a mixture of T_EX and low-level manipulation. The offset can be handled by T_EX, 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.

```

1528 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1529 {
1530 \__kernel_backend_scope_begin:
1531 <*luatex | pdftex>
1532 \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1533 </luatex | pdftex>
1534 <*dviptdpmx | xetex>
1535 \__kernel_backend_literal:n
1536 { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1537 </dviptdpmx | xetex>
1538 \hbox_overlap_right:n { \box_use:N #1 }
1539 <*dviptdpmx | xetex>
1540 \__kernel_backend_literal:n { pdf:etrans }
1541 </dviptdpmx | xetex>
1542 \__kernel_backend_scope_end:
1543 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1544 </dviptdpmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1545 <*dvisvgm>

```

__draw_backend_literal:n
__draw_backend_literal:x

The same as the more general literal call.

```

1546 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1547 \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

```

1548 \cs_new_protected:Npn \__draw_backend_begin:
1549 {
1550   \__kernel_backend_scope_begin:
1551   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1552 }
1553 \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`

```

1554 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1555 {
1556   \__draw_backend_add_to_path:n
1557   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1558 }
1559 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1560 {
1561   \__draw_backend_add_to_path:n
1562   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1563 }
1564 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1565 {
1566   \__draw_backend_add_to_path:n
1567   {
1568     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1569     h ~ \dim_to_decimal:n {#3} ~
1570     v ~ \dim_to_decimal:n {#4} ~
1571     h ~ \dim_to_decimal:n { -#3 } ~
1572     Z
1573   }
1574 }
1575 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1576 {
1577   \__draw_backend_add_to_path:n
1578   {
1579     C ~
1580     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1581     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1582     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1583   }
1584 }
1585 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1586 {
1587   \tl_gset:Nx \g__draw_draw_path_tl
1588   {
1589     \g__draw_draw_path_tl
1590     \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1591     #1

```

```

1592     }
1593   }
1594   \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:

```

1595 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1596 { \__draw_backend_scope:n { fill-rule="evenodd" } }
1597 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1598 { \__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:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int

1599 \cs_new_protected:Npn \__draw_backend_closepath:
1600 { \__draw_backend_add_to_path:n { Z } }
1601 \cs_new_protected:Npn \__draw_backend_path:n #1
1602 {
1603   \bool_if:NTF \g__draw_draw_clip_bool
1604   {
1605     \int_gincr:N \g__draw_clip_path_int
1606     \__draw_backend_literal:x
1607     {
1608       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1609       { ?nl }
1610       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1611       < /clipPath > { ? nl }
1612       <
1613         use~xlink:href =
1614         "\c_hash_str l3path \int_use:N \g__draw_path_int " ~
1615         #1
1616       />
1617     }
1618     \__draw_backend_scope:x
1619     {
1620       clip-path =
1621       "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int )"
1622     }
1623   }
1624   {
1625     \__draw_backend_literal:x
1626     { <path ~ d=" \g__draw_draw_path_tl " ~ #1 /> }
1627   }
1628   \tl_gclear:N \g__draw_draw_path_tl
1629   \bool_gset_false:N \g__draw_draw_clip_bool
1630 }
1631 \int_new:N \g__draw_path_int
1632 \cs_new_protected:Npn \__draw_backend_stroke:
1633 { \__draw_backend_path:n { style="fill:none" } }

```

```

1634 \cs_new_protected:Npn \__draw_backend_closestroke:
1635 {
1636   \__draw_backend_closepath:
1637   \__draw_backend_stroke:
1638 }
1639 \cs_new_protected:Npn \__draw_backend_fill:
1640 { \__draw_backend_path:n { style="stroke:none" } }
1641 \cs_new_protected:Npn \__draw_backend_fillstroke:
1642 { \__draw_backend_path:n { } }
1643 \cs_new_protected:Npn \__draw_backend_clip:
1644 { \bool_gset_true:N \g__draw_draw_clip_bool }
1645 \bool_new:N \g__draw_draw_clip_bool
1646 \cs_new_protected:Npn \__draw_backend_discardpath:
1647 {
1648   \bool_if:NT \g__draw_draw_clip_bool
1649   {
1650     \int_gincr:N \g__draw_clip_path_int
1651     \__draw_backend_literal:x
1652     {
1653       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1654       { ?nl }
1655       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1656       < /clipPath >
1657     }
1658     \__draw_backend_scope:x
1659     {
1660       clip-path =
1661       "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1662     }
1663   }
1664   \tl_gclear:N \g__draw_draw_path_tl
1665   \bool_gset_false:N \g__draw_draw_clip_bool
1666 }

```

(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:
1667 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1668 {
1669   \use:x
1670   {
1671     \__draw_backend_dash_aux:nn
1672     { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1673     { \dim_to_decimal:n {#2} }
1674   }
1675 }
1676 \cs_new:Npn \__draw_backend_dash:n #1
1677 { , \dim_to_decimal_in_bp:n {#1} }
1678 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1679 {
1680   \__draw_backend_scope:x
1681   {
1682     stroke-dasharray =

```

```

1683         "
1684         \tl_if_empty:oTF { \use_none:n #1 }
1685         { none }
1686         { \use_none:n #1 }
1687     " ~
1688     stroke-offset=" #2 "
1689 }
1690 }
1691 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1692 { \__draw_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1693 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1694 { \__draw_backend_scope:x { stroke-miterlimit=" #1 " } }
1695 \cs_new_protected:Npn \__draw_backend_cap_but:
1696 { \__draw_backend_scope:n { stroke-linecap="butt" } }
1697 \cs_new_protected:Npn \__draw_backend_cap_round:
1698 { \__draw_backend_scope:n { stroke-linecap="round" } }
1699 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1700 { \__draw_backend_scope:n { stroke-linecap="square" } }
1701 \cs_new_protected:Npn \__draw_backend_join_miter:
1702 { \__draw_backend_scope:n { stroke-linejoin="miter" } }
1703 \cs_new_protected:Npn \__draw_backend_join_round:
1704 { \__draw_backend_scope:n { stroke-linejoin="round" } }
1705 \cs_new_protected:Npn \__draw_backend_join_bevel:
1706 { \__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.

```

1707 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1708 {
1709     \__draw_backend_scope:n
1710     {
1711         transform =
1712         " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1713     }
1714 }

```

(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.

```

1715 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1716 {
1717     \__kernel_backend_scope_begin:
1718     \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1719     \__kernel_backend_literal_svg:n
1720     {
1721         < g~
1722         stroke="none"~
1723         transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1724         >
1725     }
1726     \box_set_wd:Nn #1 { Opt }

```

```

1727 \box_set_ht:Nn #1 { Opt }
1728 \box_set_dp:Nn #1 { Opt }
1729 \box_use:N #1
1730 \__kernel_backend_literal_svg:n { </g> }
1731 \__kernel_backend_scope_end:
1732 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```
1733 </dvisvgm>
```

```
1734 </package>
```

5 l3backend-graphics Implementation

```

1735 <*package>
1736 <@@=graphics>

```

5.1 dvips backend

```
1737 <*dvips>
```

_graphics_backend_getbb_eps:n Simply use the generic function.

```
1738 \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.

```

1739 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1740 {
1741   \__kernel_backend_literal:x
1742   {
1743     PSfile = #1 \c_space_tl
1744     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1745     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1746     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1747     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1748   }
1749 }

```

(End definition for __graphics_backend_include_eps:n.)

```
1750 </dvips>
```

5.2 LuaTeX and pdfTeX backends

```
1751 <*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.

```
1752 \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.

```

1753 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1754 {
1755   \int_zero:N \l_graphics_page_int
1756   \tl_clear:N \l_graphics_pagebox_tl
1757   \tl_set:Nx \l__graphics_graphics_attr_tl
1758   {
1759     \tl_if_empty:NF \l_graphics_decodearray_tl
1760     { :D \l_graphics_decodearray_tl }
1761     \bool_if:NT \l_graphics_interpolate_bool
1762     { :I }
1763   }
1764   \tl_clear:N \l__graphics_graphics_attr_tl
1765   \_graphics_backend_getbb_auxi:n {#1}
1766 }
1767 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1768 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1769 {
1770   \tl_clear:N \l_graphics_decodearray_tl
1771   \bool_set_false:N \l_graphics_interpolate_bool
1772   \tl_set:Nx \l__graphics_graphics_attr_tl
1773   {
1774     : \l_graphics_pagebox_tl
1775     \int_compare:nNnT \l_graphics_page_int > 1
1776     { :P \int_use:N \l_graphics_page_int }
1777   }
1778   \_graphics_backend_getbb_auxi:n {#1}
1779 }
1780 \cs_new_protected:Npn \_graphics_backend_getbb_auxi:n #1
1781 {
1782   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1783   { \_graphics_backend_getbb_auxii:n {#1} }
1784 }

```

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.

```

1785 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:n #1
1786 {
1787   \tex_immediate:D \tex_pdfximage:D
1788   \bool_lazy_or:nnT
1789   { \l_graphics_interpolate_bool }
1790   { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1791   {
1792     attr ~
1793     {
1794       \tl_if_empty:NF \l_graphics_decodearray_tl
1795       { /Decode~[ \l_graphics_decodearray_tl ] }
1796       \bool_if:NT \l_graphics_interpolate_bool
1797       { /Interpolate~true }

```

```

1798     }
1799   }
1800   \int_compare:nNnT \l_graphics_page_int > 0
1801     { page ~ \int_use:N \l_graphics_page_int }
1802   \tl_if_empty:NF \l_graphics_pagebox_tl
1803     { \l_graphics_pagebox_tl }
1804   {#1}
1805   \hbox_set:Nn \l__graphics_internal_box
1806     { \tex_pdfrefximage:D \tex_pdflastximage:D }
1807   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1808   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1809   \int_const:cn { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1810     { \tex_the:D \tex_pdflastximage:D }
1811   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1812 }

```

(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.

```

1813 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1814 {
1815   \tex_pdfrefximage:D
1816   \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1817 }
1818 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1819 \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`
`\l__graphics_backend_dir_str`
`\l__graphics_backend_name_str`
`\l__graphics_backend_ext_str`

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.

```

1820 \sys_if_shell:T
1821 {
1822   \str_new:N \l__graphics_backend_dir_str
1823   \str_new:N \l__graphics_backend_name_str
1824   \str_new:N \l__graphics_backend_ext_str
1825   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1826   {
1827     \file_parse_full_name:nNNN {#1}
1828     \l__graphics_backend_dir_str
1829     \l__graphics_backend_name_str
1830     \l__graphics_backend_ext_str
1831     \exp_args:Nx \__graphics_backend_getbb_eps:nn
1832     {
1833       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1834       -converted-to.pdf
1835     }
1836     {#1}
1837   }
1838   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2

```



```

1839     {
1840       \file_compare_timestamp:nNnT {#2} > {#1}
1841       {
1842         \sys_shell_now:n
1843         { repstopdf ~ #2 ~ #1 }
1844       }
1845       \tl_set:Nn \l_graphics_name_tl {#1}
1846       \__graphics_backend_getbb_pdf:n {#1}
1847     }
1848     \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1849     {
1850       \file_parse_full_name:nNNN {#1}
1851       \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1852       \exp_args:Nx \__graphics_backend_include_pdf:n
1853       {
1854         \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1855         -converted-to.pdf
1856       }
1857     }
1858   }

```

(End definition for __graphics_backend_getbb_eps:n and others.)

1859 </luatex | pdftex>

5.3 dvipdfmx backend

1860 <*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
1861 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n
1862 <*dvipdfmx>
1863 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1864 {
1865   \int_zero:N \l_graphics_page_int
1866   \tl_clear:N \l_graphics_pagebox_tl
1867   \graphics_extract_bb:n {#1}
1868 }
1869 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1870 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1871 {
1872   \tl_clear:N \l_graphics_decodearray_tl
1873   \bool_set_false:N \l_graphics_interpolate_bool
1874   \graphics_extract_bb:n {#1}
1875 }
1876 </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.

1877 \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.

```

1878 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1879 {
1880   \_kernel_backend_literal:x
1881   {
1882     PSfile = #1 \c_space_tl
1883     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1884     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1885     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1886     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1887   }
1888 }
1889 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1890 { \_graphics_backend_include_auxi:nn {#1} { image } }
1891 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1892 <*/dvipdfmx>
1893 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1894 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1895 </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.

```

1896 \cs_new_protected:Npn \_graphics_backend_include_auxi:nn #1#2
1897 {
1898   \_graphics_backend_include_auxii:xnn
1899   {
1900     \tl_if_empty:NF \l_graphics_pagebox_tl
1901     { : \l_graphics_pagebox_tl }
1902     \int_compare:nNnT \l_graphics_page_int > 1
1903     { :P \int_use:N \l_graphics_page_int }
1904     \tl_if_empty:NF \l_graphics_decodearray_tl
1905     { :D \l_graphics_decodearray_tl }
1906     \bool_if:NT \l_graphics_interpolate_bool
1907     { :I }
1908   }
1909   {#1} {#2}
1910 }
1911 \cs_new_protected:Npn \_graphics_backend_include_auxii:nnn #1#2#3
1912 {
1913   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1914   {
1915     \_kernel_backend_literal:x
1916     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1917   }
1918   { \_graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1919 }
1920 \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!

```

1921 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1922 {
1923   \int_gincr:N \g__graphics_track_int
1924   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1925   \__kernel_backend_literal:x
1926   {
1927     pdf:#3~
1928     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1929     \int_compare:nNnT \l_graphics_page_int > 1
1930     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1931     \tl_if_empty:NF \l_graphics_pagebox_tl
1932     {
1933       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1934       bbox ~
1935         \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1936         \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1937         \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1938         \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1939     }
1940     (#1)
1941     \bool_lazy_or:nnT
1942     { \l_graphics_interpolate_bool }
1943     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1944     {
1945       <<
1946       \tl_if_empty:NF \l_graphics_decodearray_tl
1947       { /Decode~[ \l_graphics_decodearray_tl ] }
1948       \bool_if:NT \l_graphics_interpolate_bool
1949       { /Interpolate~true> }
1950     }
1951     >>
1952   }
1953 }

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

1954 </dviPDFmx|xetex>

```

5.4 X_YTeX backend

```

1955 <*xetex>

```

5.4.1 Images

For X_YTeX, there are two primitives that allow us to obtain the bounding box without needing `extractbb`. The only complexity is passing the various minor variations to a common core process. The X_YTeX primitive omits the text box from the page box specification, so there is also some “trimming” to do here.

```

1956 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1957 {
1958   \int_zero:N \l_graphics_page_int
1959   \tl_clear:N \l_graphics_pagebox_tl
1960   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1961 }

```

```

\__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:nnN
\__graphics_backend_getbb_auxiv:nnN
\__graphics_backend_getbb_auxiv:nnN
\__graphics_backend_getbb_auxv:nnN
\__graphics_backend_getbb_auxv:nnN
\__graphics_backend_getbb_pagebox:w

```

```

1962 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1963 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1964 {
1965   \tl_clear:N \l_graphics_decodearray_tl
1966   \bool_set_false:N \l_graphics_interpolate_bool
1967   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1968 }
1969 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
1970 {
1971   \int_compare:nNnTF \l_graphics_page_int > 1
1972     { \__graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
1973     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1974 }
1975 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
1976 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1977 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
1978 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1979 {
1980   \tl_if_empty:NTF \l_graphics_pagebox_tl
1981     { \__graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1982     { \__graphics_backend_getbb_auxv:nNnn {#1} #2 {#3} {#4} }
1983 }
1984 }
1985 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
1986 {
1987   \use:x
1988   {
1989     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1990     { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1991   }
1992 }
1993 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1994 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1995 {
1996   \graphics_bb_restore:nF {#1#3}
1997   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1998 }
1999 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2000 {
2001   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
2002   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2003   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2004   \graphics_bb_save:n {#1#3}
2005 }
2006 \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

__graphics_backend_include_pdf:n
 __graphics_backend_include_bitmap_quote:w

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

```

2007 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2008 {

```

```

2009 \tex_XeTeXpdffile:D
2010 \__graphics_backend_include_pdf_quote:w #1 "#1" \s__graphics_stop \c_space_tl
2011 \int_compare:nNnT \l_graphics_page_int > 0
2012 { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
2013 \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
2014 }
2015 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \s__graphics_stop
2016 { " #2 " }

```

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

```

2017 </xetex>

```

5.5 dvisvgm backend

```

2018 <*dvisvgm>

```

`__graphics_backend_getbb_eps:n` Simply use the generic function.

```

2019 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n

```

(End definition for `__graphics_backend_getbb_eps:n`.)

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

```

\__graphics_backend_getbb_jpg:n
2020 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2021 {
2022   \int_zero:N \l_graphics_page_int
2023   \tl_clear:N \l_graphics_pagebox_tl
2024   \graphics_extract_bb:n {#1}
2025 }
2026 \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

```

2027 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2028 {
2029   \tl_clear:N \l_graphics_decodearray_tl
2030   \bool_set_false:N \l_graphics_interpolate_bool
2031   \graphics_extract_bb:n {#1}
2032 }

```

(End definition for `__graphics_backend_getbb_pdf:n`.)

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

```

\__graphics_backend_include_pdf:n
\__graphics_backend_include:nn
2033 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2034 { __graphics_backend_include:nn { PSfile } {#1} }
2035 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2036 { __graphics_backend_include:nn { pdffile } {#1} }
2037 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
2038 {
2039   \__kernel_backend_literal:x
2040   {
2041     #1 = #2 \c_space_tl
2042     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl

```

```

2043         lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2044         urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2045         ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
2046     }
2047 }

```

(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.

```

2048 \cs_new_protected:Npn \__graphics_backend_include_png:n #1
2049 {
2050     \__kernel_backend_literal:x
2051     {
2052         dvisvgm:img~
2053         \dim_to_decimal:n { \l_graphics_ury_dim } ~
2054         \dim_to_decimal:n { \l_graphics_ury_dim } ~
2055         \__graphics_backend_include_bitmap_quote:w #1 " #1 " \s__graphics_stop
2056     }
2057 }
2058 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_png:n
2059 \cs_new:Npn \__graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \s__graphics_stop
2060 { " #2 " }

```

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

```

2061 \enddvisvgm
2062 \endpackage

```

6 l3backend-pdf Implementation

```

2063 \*package
2064 \@@=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

2065 \box_new:N \l__pdf_internal_box

(End definition for \l__pdf_internal_box.)

```

6.2 dvips backend

2066 `<*dvips>`

`__pdf_backend_pdfmark:n`
`__pdf_backend_pdfmark:x`

Used often enough it should be a separate function.

2067 `\cs_new_protected:Npn __pdf_backend_pdfmark:n #1`
 2068 `{ __kernel_backend_postscript:n { mark #1 ~ pdfmark } }`
 2069 `\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`

2070 `\cs_new_protected:Npn __pdf_backend_catalog_gput:nn #1#2`
 2071 `{ __pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }`
 2072 `\cs_new_protected:Npn __pdf_backend_info_gput:nn #1#2`
 2073 `{ __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.

2074 `\int_new:N \g__pdf_backend_object_int`
 2075 `\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`.

2076 `\cs_new_protected:Npn __pdf_backend_object_new:nn #1#2`
 2077 `{`
 2078 `\int_gincr:N \g__pdf_backend_object_int`
 2079 `\int_const:cn`
 2080 `{ c__pdf_backend_object_ \tl_to_str:n {#1} _int }`
 2081 `{ \g__pdf_backend_object_int }`
 2082 `\prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}`
 2083 `}`
 2084 `\cs_new:Npn __pdf_backend_object_ref:n #1`
 2085 `{ { 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:nmm`

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

2086 `\cs_new_protected:Npn __pdf_backend_object_write:nn #1#2`
 2087 `{`
 2088 `_pdf_backend_pdfmark:x`
 2089 `{`
 2090 `/_objdef ~ __pdf_backend_object_ref:n {#1}`
 2091 `/type`
 2092 `\str_case_e:nn`
 2093 `{ \prop_item:Nn \g__pdf_backend_object_prop {#1} }`
 2094 `{`
 2095 `{ array } { /array }`
 2096 `{ dict } { /dict }`

```

2097         { fstream } { /stream }
2098         { stream } { /stream }
2099     }
2100     /OBJ
2101 }
2102 \use:c
2103 { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2104 { __pdf_backend_object_ref:n {#1} } {#2}
2105 }
2106 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2107 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2108 {
2109     \__pdf_backend_pdfmark:x
2110     { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2111 }
2112 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2113 {
2114     \__pdf_backend_pdfmark:x
2115     { #1 << \exp_not:n {#2} >> /PUT }
2116 }
2117 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2118 {
2119     \exp_args:Nx
2120     \__pdf_backend_object_write_fstream:nnn {#1} #2
2121 }
2122 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2123 {
2124     \__kernel_backend_postscript:n
2125     {
2126         SDict ~ begin ~
2127         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2128         mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2129         end
2130     }
2131 }
2132 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2133 {
2134     \exp_args:Nx
2135     \__pdf_backend_object_write_stream:nnn {#1} #2
2136 }
2137 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2138 {
2139     \__kernel_backend_postscript:n
2140     {
2141         mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2142         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2143     }
2144 }

```

(End definition for __pdf_backend_object_write:nn and others.)

__pdf_backend_object_now:nn
__pdf_backend_object_now:nx

No anonymous objects, so things are done manually.

```

2145 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2146 {

```



```

2147 \int_gincr:N \g__pdf_backend_object_int
2148 \__pdf_backend_pdfmark:x
2149 {
2150   /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2151   /type
2152   \str_case:nn
2153     {#1}
2154     {
2155       { array } { /array }
2156       { dict } { /dict }
2157       { fstream } { /stream }
2158       { stream } { /stream }
2159     }
2160   /OBJ
2161 }
2162 \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2163 { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2164 }
2165 \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.

```

2166 \cs_new:Npn \__pdf_backend_object_last:
2167 { { 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.

```

2168 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2169 { { 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.

```

2170 \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.

```

2171 \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.

```

2172 \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.

```
2173 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2174 {
2175   \exp_args:Nf \_pdf_backend_annotation_aux:nnnn
2176   { \dim_eval:n {#1} } {#2} {#3} {#4}
2177 }
2178 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2179 {
2180   \box_move_down:nn {#3}
2181   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2182   \box_move_up:nn {#2}
2183   {
2184     \hbox:n
2185     {
2186       \_kernel_kern:n {#1}
2187       \_kernel_backend_postscript:n { pdf.save.ur }
2188       \_kernel_kern:n { -#1 }
2189     }
2190   }
2191   \int_gincr:N \g__pdf_backend_object_int
2192   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2193   \_pdf_backend_pdfmark:x
2194   {
2195     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2196     pdf.rect
2197     #4 ~
2198     /ANN
2199   }
2200 }
```

(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.

```
2201 \cs_new:Npn \_pdf_backend_annotation_last:
2202 { { 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.

```
2203 \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.

```
2204 \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.

```
2205 \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.

2206 `\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.

2207 `\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.

2208 `\tl_new:N \l__pdf_breaklink_pdfmark_tl`

2209 `\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.

2210 `\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.

2211 `\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`

2212 `\cs_new_protected:Npn __pdf_backend_link_begin_goto:nnw #1#2`

2213 `{`

2214 `__pdf_backend_link_begin:nw`

2215 `{ #1 /Subtype /Link /Action << /S /GoTo /D (#2) >> }`

2216 `}`

2217 `\cs_new_protected:Npn __pdf_backend_link_begin_user:nnw #1#2`

2218 `{ __pdf_backend_link_begin:nw {#1#2} }`

2219 `\cs_new_protected:Npn __pdf_backend_link_begin:nw #1`

2220 `{`

```

2221 \bool_if:NF \g__pdf_backend_link_bool
2222 { \__pdf_backend_link_begin_aux:nw {#1} }
2223 }

```

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.

```

2224 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2225 {
2226   \bool_gset_true:N \g__pdf_backend_link_bool
2227   \__kernel_backend_postscript:n
2228   { /pdf.link.dict ( #1 ) def }
2229   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2230   \__pdf_backend_link_sf_save:
2231   \mode_if_math:TF
2232   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2233   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2234   \hbox_set:Nw \l__pdf_backend_content_box
2235   \__pdf_backend_link_sf_restore:
2236   \bool_if:NT \g__pdf_backend_link_math_bool
2237   { \c_math_toggle_token }
2238 }
2239 \cs_new_protected:Npn \__pdf_backend_link_end:
2240 {
2241   \bool_if:NT \g__pdf_backend_link_bool
2242   { \__pdf_backend_link_end_aux: }
2243 }
2244 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2245 {
2246   \bool_if:NT \g__pdf_backend_link_math_bool
2247   { \c_math_toggle_token }
2248   \__pdf_backend_link_sf_save:
2249   \hbox_set_end:
2250   \__pdf_backend_link_minima:
2251   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2252   \exp_args:Nx \__pdf_backend_link_outerbox:n
2253   {
2254     \int_if_odd:nTF { \value { page } }
2255     { \oddsidemargin }
2256     { \evensidemargin }
2257   }
2258   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2259   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2260   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2261   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2262   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2263   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2264   {
2265     \hbox:n
2266     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2267   }
2268   \int_gincr:N \g__pdf_backend_object_int
2269   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2270   \__kernel_backend_postscript:x
2271   {

```

```

2272     mark
2273     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2274     \g__pdf_backend_link_dict_tl \c_space_tl
2275     pdf.rect
2276     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2277   }
2278   \__pdf_backend_link_sf_restore:
2279   \bool_gset_false:N \g__pdf_backend_link_bool
2280 }
2281 \cs_new_protected:Npn \__pdf_backend_link_minima:
2282 {
2283   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2284   \__kernel_backend_postscript:x
2285   {
2286     /pdf.linkdp.pad ~
2287     \dim_to_decimal:n
2288     {
2289       \dim_max:nn
2290       {
2291         \box_dp:N \l__pdf_backend_model_box
2292         - \box_dp:N \l__pdf_backend_content_box
2293       }
2294       { Opt }
2295     } ~
2296     pdf.pt.dvi ~ def
2297   /pdf.linkht.pad ~
2298   \dim_to_decimal:n
2299   {
2300     \dim_max:nn
2301     {
2302       \box_ht:N \l__pdf_backend_model_box
2303       - \box_ht:N \l__pdf_backend_content_box
2304     }
2305     { Opt }
2306   } ~
2307   pdf.pt.dvi ~ def
2308 }
2309 }
2310 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2311 {
2312   \__kernel_backend_postscript:x
2313   {
2314     /pdf.outerbox
2315     [
2316       \dim_to_decimal:n {#1} ~
2317       \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2318       \dim_to_decimal:n { #1 + \textwidth } ~
2319       \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2320     ]
2321     [ exch { pdf.pt.dvi } forall ] def
2322   /pdf.baselineskip ~
2323   \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2324   { pdf.pt.dvi ~ def }
2325   { pop ~ pop }

```

```

2326         ifelse
2327     }
2328 }
2329 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2330 {
2331     \int_gset:Nn \g__pdf_backend_link_sf_int
2332     {
2333         \mode_if_horizontal:TF
2334         { \tex_spacefactor:D }
2335         { 0 }
2336     }
2337 }
2338 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2339 {
2340     \mode_if_horizontal:T
2341     {
2342         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2343         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2344     }
2345 }

```

(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.

```

2346 \use_none:n
2347 {
2348     \cs_if_exist:NT \@makecol@hook
2349     {
2350         \tl_put_right:Nn \@makecol@hook
2351         {
2352             \box_if_empty:NF \@cclv
2353             {
2354                 \vbox_set:Nn \@cclv
2355                 {
2356                     \__kernel_backend_postscript:n
2357                     {
2358                         pdf.globaldict /pdf.brokenlink.rect ~ known
2359                         { pdf.bordertracking.continue }
2360                         if
2361                     }
2362                     \vbox_unpack_drop:N \@cclv
2363                     \__kernel_backend_postscript:n
2364                     { pdf.bordertracking.endpage }
2365                 }
2366             }
2367         }
2368         \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2369         \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2370         \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2371     }
2372 }

```

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

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

```
2373 \cs_new:Npn \_pdf_backend_link_last:
2374 { { 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.

```
2375 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2376 {
2377   \__kernel_backend_postscript:x
2378   {
2379     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2380   }
2381 }
```

(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.

```
2382 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2383 {
2384   \__kernel_backend_postscript:n { pdf.dest.anchor }
2385   \_pdf_backend_pdfmark:x
2386   {
2387     /View
2388     [
2389       \str_case:nnF {#2}
2390       {
2391         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2392         { fit } { /Fit }
2393         { fitb } { /FitB }
2394         { fitbh } { /FitBH ~ pdf.dest.y }
2395         { fitbv } { /FitBV ~ pdf.dest.x }
2396         { fith } { /FitH ~ pdf.dest.y }
2397         { fitv } { /FitV ~ pdf.dest.x }
2398         { fitr } { /Fit }
2399       }
2400       {
2401         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2402       }
2403     ]
2404     /Dest ( \exp_not:n {#1} ) cvn
2405     /DEST
2406   }
2407 }
2408 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2409 {
2410   \exp_args:Ne \_pdf_backend_destination_aux:nnnn
2411   { \dim_eval:n {#2} } {#1} {#3} {#4}
2412 }
```

```

2413 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2414 {
2415   \vbox_to_zero:n
2416   {
2417     \__kernel_kern:n {#4}
2418     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2419     \tex_vss:D
2420   }
2421   \__kernel_kern:n {#1}
2422   \vbox_to_zero:n
2423   {
2424     \__kernel_kern:n { -#3 }
2425     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2426     \tex_vss:D
2427   }
2428   \__kernel_kern:n { -#1 }
2429   \__pdf_backend_pdfmark:n
2430   {
2431     /View
2432     [
2433       /FitR ~
2434       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2435       pdf.urx ~ pdf.ury ~ pdf.dest2device
2436     ]
2437     /Dest ( #2 ) cvn
2438     /DEST
2439   }
2440 }

```

(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
2441 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2442 {
2443   \int_compare:nNnT {#1} = 0
2444   {
2445     \__kernel_backend_literal_postscript:n
2446     {
2447       /setdistillerparams ~ where
2448       { pop << /CompressPages ~ false >> setdistillerparams }
2449       if
2450     }
2451   }
2452 }
2453 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2454 {
2455   \bool_if:nF {#1}
2456   {
2457     \__kernel_backend_literal_postscript:n
2458     {
2459       /setdistillerparams ~ where

```



```

2460         { pop << /CompressStreams ~ false >> setdistillerparams }
2461     if
2462     }
2463 }
2464 }

```

(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`

```

2465 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2466 {
2467     \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2468 }
2469 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2470 {
2471     \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2472 }

```

(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:`

```

2473 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2474 \cs_new:Npn \_pdf_backend_version_minor: { -1 }

```

(End definition for `_pdf_backend_version_major:` and `_pdf_backend_version_minor:.`)

6.2.5 Marked content

`_pdf_backend_bdc:nn`

Simple wrappers.

`_pdf_backend_emc:`

```

2475 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2476 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2477 \cs_new_protected:Npn \_pdf_backend_emc:
2478 { \_pdf_backend_pdfmark:n { /EMC } }

```

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

2479 \langle /dvips \rangle

6.3 LuaTeX and pdfTeX backend

2480 \langle *luatex | pdftex \rangle

6.3.1 Annotations

`_pdf_backend_annotation:nnnn`

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

```

2481 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2482 {
2483      $\langle$ *luatex $\rangle$ 
2484     \tex_pdfextension:D annot ~
2485      $\langle$ /luatex $\rangle$ 
2486      $\langle$ *pdftex $\rangle$ 
2487     \tex_pdfannot:D
2488      $\langle$ /pdftex $\rangle$ 
2489     width ~ \dim_eval:n {#1} ~
2490     height ~ \dim_eval:n {#2} ~
2491     depth ~ \dim_eval:n {#3} ~

```

```

2492     {#4}
2493 }

```

(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.

```

2494 \cs_new:Npx \_pdf_backend_annotation_last:
2495 {
2496     \exp_not:N \int_value:w
2497 <*luatex>
2498     \exp_not:N \tex_pdffeedback:D lastannot ~
2499 </luatex>
2500 <*pdfTeX>
2501     \exp_not:N \tex_pdflastannot:D
2502 </pdfTeX>
2503     \c_space_tl 0 ~ R
2504 }

```

(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 2505 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
\_pdf_backend_link_begin:nnnw 2506 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
\_pdf_backend_link_end: 2507 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2508 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2509 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2510 {
2511 <*luatex>
2512     \tex_pdfextension:D startlink ~
2513 </luatex>
2514 <*pdfTeX>
2515     \tex_pdfstartlink:D
2516 </pdfTeX>
2517     attr {#1}
2518     #2 {#3}
2519 }
2520 \cs_new_protected:Npn \_pdf_backend_link_end:
2521 {
2522 <*luatex>
2523     \tex_pdfextension:D endlink \scan_stop:
2524 </luatex>
2525 <*pdfTeX>
2526     \tex_pdfendlink:D
2527 </pdfTeX>
2528 }

```

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

`_pdf_backend_link_last:` Formatted for direct use.

```

2529 \cs_new:Npx \_pdf_backend_link_last:
2530 {
2531     \exp_not:N \int_value:w
2532 <*luatex>

```

```

2533     \exp_not:N \tex_pdffeedback:D lastlink ~
2534 </luatex>
2535 <*pdfTeX>
2536     \exp_not:N \tex_pdflastlink:D
2537 </pdfTeX>
2538     \c_space_tl 0 ~ R
2539 }

```

(End definition for _pdf_backend_link_last:.)

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

```

2540 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2541 {
2542 <*luatex>
2543     \tex_pdfvariable:D linkmargin
2544 </luatex>
2545 <*pdfTeX>
2546     \tex_pdflinkmargin:D
2547 </pdfTeX>
2548     \dim_eval:n {#1} \scan_stop:
2549 }

```

(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

```

2550 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2551 {
2552 <*luatex>
2553     \tex_pdfextension:D dest ~
2554 </luatex>
2555 <*pdfTeX>
2556     \tex_pdfdest:D
2557 </pdfTeX>
2558     name {#1}
2559     \str_case:nnF {#2}
2560     {
2561         { xyz } { xyz }
2562         { fit } { fit }
2563         { fitb } { fitb }
2564         { fitbh } { fitbh }
2565         { fitbv } { fitbv }
2566         { fith } { fith }
2567         { fitv } { fitv }
2568         { fitr } { fitr }
2569     }
2570     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2571     \scan_stop:
2572 }
2573 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2574 {
2575 <*luatex>
2576     \tex_pdfextension:D dest ~

```

```

2577 </luatex>
2578 <*pdftex>
2579   \tex_pdfdest:D
2580 </pdftex>
2581   name {#1}
2582   fitr ~
2583     width \dim_eval:n {#2} ~
2584     height \dim_eval:n {#3} ~
2585     depth \dim_eval:n {#4} \scan_stop:
2586 }

```

(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
2587 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2588 {
2589 <*luatex>
2590   \tex_pdfextension:D catalog
2591 </luatex>
2592 <*pdftex>
2593   \tex_pdfcatalog:D
2594 </pdftex>
2595   { / #1 ~ #2 }
2596 }
2597 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2598 {
2599 <*luatex>
2600   \tex_pdfextension:D info
2601 </luatex>
2602 <*pdftex>
2603   \tex_pdfinfo:D
2604 </pdftex>
2605   { / #1 ~ #2 }
2606 }

```

(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.

```

2607 \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

```

2608 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2
2609 {
2610 <*luatex>
2611   \tex_pdfextension:D obj ~
2612 </luatex>
2613 <*pdftex>
2614   \tex_pdfobj:D

```

```

2615 </pdfTeX>
2616     reserveobjnum ~
2617     \int_const:cn
2618     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2619 <*luatex>
2620     { \tex_pdffeedback:D lastobj }
2621 </luatex>
2622 <*pdfTeX>
2623     { \tex_pdflastobj:D }
2624 </pdfTeX>
2625     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2626 }
2627 \cs_new:Npn \__pdf_backend_object_ref:n #1
2628 { \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

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

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

```

\__pdf_backend_object_write:nx
\__pdf_exp_not_i:nn
\__pdf_exp_not_ii:nn
2629 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2630 {
2631 <*luatex>
2632     \tex_immediate:D \tex_pdfextension:D obj ~
2633 </luatex>
2634 <*pdfTeX>
2635     \tex_immediate:D \tex_pdfobj:D
2636 </pdfTeX>
2637     useobjnum ~
2638     \int_use:c
2639     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2640     \str_case:e:nn
2641     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2642     {
2643         { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2644         { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2645         { fstream }
2646         {
2647             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2648             file ~ { \__pdf_exp_not_ii:nn #2 }
2649         }
2650         { stream }
2651         {
2652             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2653             { \__pdf_exp_not_ii:nn #2 }
2654         }
2655     }
2656 }
2657 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2658 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2659 \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
2660 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2

```

```

2661 {
2662 <*luatex>
2663   \tex_immediate:D \tex_pdfextension:D obj ~
2664 </luatex>
2665 <*pdftex>
2666   \tex_immediate:D \tex_pdfobj:D
2667 </pdftex>
2668   \str_case:nn
2669     {#1}
2670   {
2671     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2672     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2673     { fstream }
2674     {
2675       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2676       file ~ { \__pdf_exp_not_ii:nn #2 }
2677     }
2678     { stream }
2679     {
2680       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2681       { \__pdf_exp_not_ii:nn #2 }
2682     }
2683   }
2684 }
2685 \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.

```

2686 \cs_new:Npx \__pdf_backend_object_last:
2687 {
2688   \exp_not:N \int_value:w
2689 <*luatex>
2690   \exp_not:N \tex_pdffeedback:D lastobj ~
2691 </luatex>
2692 <*pdftex>
2693   \exp_not:N \tex_pdflastobj:D
2694 </pdftex>
2695   \c_space_tl 0 ~ R
2696 }

```

(End definition for __pdf_backend_object_last:.)

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

```

2697 \cs_new:Npx \__pdf_backend_pageobject_ref:n #1
2698 {
2699   \exp_not:N \int_value:w
2700 <*luatex>
2701   \exp_not:N \tex_pdffeedback:D pageref
2702 </luatex>
2703 <*pdftex>
2704   \exp_not:N \tex_pdfpageref:D
2705 </pdftex>
2706   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2707 }

```

(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
2708 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2709 {
2710   \tex_global:D
2711   \*luatex
2712   \tex_pdfvariable:D compresslevel
2713   \*luatex
2714   \*pdftex
2715   \tex_pdfcompresslevel:D
2716   \*pdftex
2717   \int_value:w \int_eval:n {#1} \scan_stop:
2718 }
2719 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2720 {
2721   \bool_if:nTF {#1}
2722   { \__pdf_backend_objcompresslevel:n { 2 } }
2723   { \__pdf_backend_objcompresslevel:n { 0 } }
2724 }
2725 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2726 {
2727   \tex_global:D
2728   \*luatex
2729   \tex_pdfvariable:D objcompresslevel
2730   \*luatex
2731   \*pdftex
2732   \tex_pdfobjcompresslevel:D
2733   \*pdftex
2734   #1 \scan_stop:
2735 }

```

(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.

```

2736 \cs_new_protected:Npx \_pdf_backend_version_major_gset:n #1
2737 {
2738   \*luatex
2739   \int_compare:nNnT \tex_luatexversion:D > { 106 }
2740   {
2741     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2742     \exp_not:N \int_eval:n {#1} \scan_stop:
2743   }
2744   \*luatex
2745   \*pdftex
2746   \cs_if_exist:NT \tex_pdfmajorversion:D
2747   {
2748     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2749     \exp_not:N \int_eval:n {#1} \scan_stop:
2750   }
2751   \*pdftex

```

```

2752 }
2753 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2754 {
2755   \tex_global:D
2756   \*luatex
2757   \tex_pdfvariable:D minorversion
2758   \*pdfTeX
2759   \tex_pdfminorversion:D
2760   \*pdfTeX
2761   \int_eval:n {#1} \scan_stop:
2762 }
2763

```

(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:
2764 \cs_new:Npx \__pdf_backend_version_major:
2765 {
2766   \*luatex
2767   \int_compare:nNnTF \tex luatexversion:D > { 106 }
2768   { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2769   { 1 }
2770   \*pdfTeX
2771   \cs_if_exist:NTF \tex_pdfmajorversion:D
2772   { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2773   { 1 }
2774   \*pdfTeX
2775   }
2776   \cs_new:Npn \__pdf_backend_version_minor:
2777   {
2778     \tex_the:D
2779     \*luatex
2780     \tex_pdfvariable:D minorversion
2781     \*pdfTeX
2782     \tex_pdfminorversion:D
2783     \*pdfTeX
2784     }
2785   }
2786

```

(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>.

```

2787 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2788 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2789 \cs_new_protected:Npn \__pdf_backend_emc:
2790 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2791 \*luatex | pdfTeX

```


6.4 dvipdfmx backend

2792 `*dvipdfmx|xetex`

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

`__pdf_backend:x`

```

2793 \cs_new_protected:Npx \__pdf_backend:n #1
2794 { \__kernel_backend_literal:n { pdf: #1 } }
2795 \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
2796 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2797 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2798 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2799 { \__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`

```

2800 \int_new:N \g__pdf_backend_object_int
2801 \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
2802 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2803 {
2804   \int_gincr:N \g__pdf_backend_object_int
2805   \int_const:cn
2806   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2807   { \g__pdf_backend_object_int }
2808   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2809 }
2810 \cs_new:Npn \__pdf_backend_object_ref:n #1
2811 { @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`

```

2812 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2813 {
2814   \exp_args:Nx \__pdf_backend_object_write:nnn
2815   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2816 }
2817 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2818 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2819 {
2820   \use:c { \__pdf_backend_object_write_ #1 :nn }
2821   { \__pdf_backend_object_ref:n {#2} } {#3}
2822 }

```

```

2823 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2824 {
2825   \__pdf_backend:x
2826   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2827 }
2828 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2829 {
2830   \__pdf_backend:x
2831   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2832 }
2833 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2834 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2835 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2836 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2837 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2838 {
2839   \__pdf_backend:x
2840   {
2841     #1 stream ~ #2 ~
2842     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2843   }
2844 }

```

(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
2845 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2846 {
2847   \int_gincr:N \g__pdf_backend_object_int
2848   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2849   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2850   {#2}
2851 }
2852 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2853 \cs_new:Npn \__pdf_backend_object_last:
2854 { @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.

```

2855 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2856 { @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.

```
2857 \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.

```
2858 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2859 {
2860   \int_gincr:N \g_pdf_backend_object_int
2861   \int_gset_eq:NN \g_pdf_backend_annotation_int \g_pdf_backend_object_int
2862   \_pdf_backend:x
2863   {
2864     ann ~ @pdf.obj \int_use:N \g_pdf_backend_object_int \c_space_tl
2865     width ~ \dim_eval:n {#1} ~
2866     height ~ \dim_eval:n {#2} ~
2867     depth ~ \dim_eval:n {#3} ~
2868     << /Type /Annot #4 >>
2869   }
2870 }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

`_pdf_backend_annotation_last:`

```
2871 \cs_new:Npn \_pdf_backend_annotation_last:
2872 { @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.

```
2873 \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:`

```
2874 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2875 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2876 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2877 { \_pdf_backend_link_begin:n {#1#2} }
2878 \cs_new_protected:Npx \_pdf_backend_link_begin:n #1
2879 {
2880   \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2881   {
2882     \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int
2883   }
2884   \_pdf_backend:x
2885   {
2886     bann ~
2887     \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2888     {
2889       @pdf.lnk
2890       \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int
2891       \c_space_tl
2892     }
2893   }
2894 }
```

```

2893         <<
2894         /Type /Annot
2895         #1
2896         >>
2897     }
2898 }
2899 \cs_new_protected:Npn \__pdf_backend_link_end:
2900 { \__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.

```

2901 \cs_new:Npx \__pdf_backend_link_last:
2902 {
2903     \int_compare:nNnF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
2904     {
2905         @pdf.lnk
2906         \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
2907     }
2908 }

```

(End definition for __pdf_backend_link_last:.)

__pdf_backend_link_margin:n Pass to dvipdfmx.

```

2909 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2910 { \__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.

```

2911 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2912 {
2913     \__pdf_backend:x
2914     {
2915         dest ~ ( \exp_not:n {#1} )
2916         [
2917             @thispage
2918             \str_case:nnF {#2}
2919             {
2920                 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2921                 { fit } { /Fit }
2922                 { fitb } { /FitB }
2923                 { fitbh } { /FitBH }
2924                 { fitbv } { /FitBV ~ @xpos }
2925                 { fith } { /FitH ~ @ypos }
2926                 { fitv } { /FitV ~ @xpos }
2927                 { fitr } { /Fit }
2928             }
2929             { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2930         ]
2931     }

```

```

2932 }
2933 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2934 {
2935   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2936   { \dim_eval:n {#2} } {#1} {#3} {#4}
2937 }
2938 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2939 {
2940   \vbox_to_zero:n
2941   {
2942     \__kernel_kern:n {#4}
2943     \hbox:n
2944     {
2945       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2946       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2947     }
2948     \tex_vss:D
2949   }
2950   \__kernel_kern:n {#1}
2951   \vbox_to_zero:n
2952   {
2953     \__kernel_kern:n { -#3 }
2954     \hbox:n
2955     {
2956       \__pdf_backend:n
2957       {
2958         dest ~ (#2)
2959         [
2960           @thispage
2961           /FitR ~
2962           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2963           @xpos ~ @ypos
2964         ]
2965       }
2966     }
2967     \tex_vss:D
2968   }
2969   \__kernel_kern:n { -#1 }
2970 }

```

(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
2971 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2972 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2973 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2974 {
2975   \bool_if:nF {#1}
2976   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2977 }

```

(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.

```

2978 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2979 {
2980   \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
2981   \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
2982 }
2983 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2984 {
2985   \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
2986   \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
2987 }

```

(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.

```

2988 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2989 \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>.

```

2990 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2991 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2992 \cs_new_protected:Npn \__pdf_backend_emc:
2993 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2994 </dviPDFmx | xetex>

```

6.5 dvisvgm backend

```

2995 <*dvisvgm>

```

6.5.1 Catalogue entries

_pdf_backend_catalog_gput:nn No-op.

```

2996 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2 { }
2997 \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.

```

2998 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2 { }
2999 \cs_new:Npn \__pdf_backend_object_ref:n #1 { }
3000 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2 { }
3001 \cs_new_protected:Npn \__pdf_backend_object_write:nx #1#2 { }
3002 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2 { }
3003 \cs_new_protected:Npn \__pdf_backend_object_now:nx #1#2 { }
3004 \cs_new:Npn \__pdf_backend_object_last: { }
3005 \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
3006 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
3007 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1 { }

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

\__pdf_backend_version_major_gset:n
\__pdf_backend_version_minor_gset:n
Data not available!
3008 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
3009 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1 { }

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

\__pdf_backend_version_major:
\__pdf_backend_version_minor:
Data not available!
3010 \cs_new:Npn \__pdf_backend_version_major: { -1 }
3011 \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:
More no-ops.
3012 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
3013 \cs_new_protected:Npn \__pdf_backend_emc: { }

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

3014 </dvisvgm>
3015 </package>

```

7 l3backend-opacity Implementation

```

3016 <*package>
3017 <@@=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.

```

3018 <*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
3019 \cs_new_protected:Npn \__opacity_backend_select:n #1
3020 {
3021   \exp_args:Nx \__opacity_backend_select_aux:n
3022   { \fp_eval:n { min(max(0,#1),1) } }
3023 }
3024 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3025 {
3026   \__opacity_backend:nnn {#1} { fill } { ca }

```

```

3027   \_opacity_backend:nnn {#1} { stroke } { CA }
3028 }
3029 \cs_new_protected:Npn \_opacity_backend_fill:n #1
3030 {
3031   \_opacity_backend:xnn
3032   { \fp_eval:n { min(max(0,#1),1) } }
3033   { fill }
3034   { ca }
3035 }
3036 \cs_new_protected:Npn \_opacity_backend_stroke:n #1
3037 {
3038   \_opacity_backend:xnn
3039   { \fp_eval:n { min(max(0,#1),1) } }
3040   { stroke }
3041   { CA }
3042 }
3043 \cs_new_protected:Npn \_opacity_backend:nnn #1#2#3
3044 {
3045   \_kernel_backend_postscript:n
3046   {
3047     product ~ (Ghostscript) ~ search
3048     {
3049       pop ~ pop ~ pop ~
3050       #1 ~ .set #2 constantalpha
3051     }
3052     {
3053       pop ~
3054       mark ~
3055       /#3 ~ #1
3056       /SetTransparency ~
3057       pdfmark
3058     }
3059     ifelse
3060   }
3061 }
3062 \cs_generate_variant:Nn \_opacity_backend:nnn { x }

```

(End definition for _opacity_backend_select:n and others.)

```

3063 </dvips>
3064 <*dvipdfmx | luatex | pdftex | xetex>

```

\c_opacity_backend_stack_int Set up a stack.

```

3065 \bool_lazy_and:nnT
3066 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3067 { \pdfmanagement_if_active_p:}
3068 {
3069   \_kernel_color_backend_stack_init:Nnn \c_opacity_backend_stack_int
3070   { page ~ direct } { /opacity 1 ~ gs }
3071   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3072   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3073 }

```

(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`

```

3074 \tl_new:N \l__opacity_backend_fill_tl
3075 \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:`

```

3076 \cs_new_protected:Npn \__opacity_backend_select:n #1
3077 {
3078   \exp_args:Nx \__opacity_backend_select_aux:n
3079   { \fp_eval:n { min(max(0,#1),1) } }
3080 }
3081 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3082 {
3083   \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3084   \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3085   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3086   { opacity #1 }
3087   { << /ca ~ #1 /CA ~ #1 >> }
3088   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3089   { /opacity #1 ~ gs }
3090   \group_insert_after:N \__opacity_backend_reset:
3091 }
3092 \bool_lazy_and:nnF
3093 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3094 { \pdfmanagement_if_active_p:}
3095 {
3096   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1 { }
3097 }
3098 \cs_new_protected:Npn \__opacity_backend_reset:
3099 { \__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`

```

3100 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3101 {
3102   \__opacity_backend_fill_stroke:xx
3103   { \fp_eval:n { min(max(0,#1),1) } }
3104   \l__opacity_backend_stroke_tl
3105 }
3106 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3107 {
3108   \__opacity_backend_fill_stroke:xx
3109   \l__opacity_backend_fill_tl
3110   { \fp_eval:n { min(max(0,#1),1) } }
3111 }
3112 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3113 {
3114   \str_if_eq:nnTF {#1} {#2}
3115   { \__opacity_backend_select_aux:n {#1} }
3116   {

```

```

3117 \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3118 \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3119 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3120 { opacity.fill #1 }
3121 { << /ca ~ #1 >> }
3122 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3123 { opacity.stroke #1 }
3124 { << /CA ~ #2 >> }
3125 \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3126 { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3127 \group_insert_after:N \__opacity_backend_reset:
3128 }
3129 }
3130 \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.)

3131 </dvipdfmx | luatex | pdftex | xetex>
3132 <*dvipdfmx | xdvipdfmx>

```

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

```

3133 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
3134 {
3135   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1
3136   {
3137     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3138     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3139     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3140     { opacity #1 }
3141     { << /ca ~ #1 /CA ~ #1 >> }
3142     \__kernel_backend_literal_pdf:n { /opacity #1 ~ gs }
3143   }
3144   \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3145   {
3146     \str_if_eq:nnTF {#1} {#2}
3147     { \__opacity_backend_select_aux:n {#1} }
3148     {
3149       \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3150       \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3151       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3152       { opacity.fill #1 }
3153       { << /ca ~ #1 >> }
3154       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3155       { opacity.stroke #1 }
3156       { << /CA ~ #2 >> }
3157       \__kernel_backend_literal_pdf:n
3158       { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3159     }
3160   }
3161 }

```

(End definition for __opacity_backend_select:n.)

```

3162 </dvipdfmx | xdvipdfmx>

```

```
3163 <*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.

```
3164 \cs_new_protected:Npn \__opacity_backend_select:n #1
3165   { \__opacity_backend:nn {#1} { } }
3166 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3167   { \__opacity_backend:nn {#1} { fill- } }
3168 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3169   { \__opacity_backend:nn { {#1} } { stroke- } }
3170 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3171   { \__kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

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

3172 </dvisvgm>
3173 </package>
```

8 l3backend-header Implementation

```
3174 <*dvips & header>
```

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

```
3175 /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.

```
3176 TeXDict begin
3177 /TeXcolorseparation { setcolor } def
3178 end
```

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

pdf.globaldict A small global dictionary for backend use.

```
3179 true setglobal
3180 /pdf.globaldict 4 dict def
3181 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.

```
3182 /pdf.cvs { 65534 string cvs } def
3183 /pdf.dvi.pt { 72.27 mul Resolution div } def
3184 /pdf.pt.dvi { 72.27 div Resolution mul } def
3185 /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 3186 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad 3187 /pdf.linkdp.pad { 0 } def
               3188 /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 3189 /pdf.rect
pdf.save.linkur 3190 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.llx 3191 /pdf.save.ll
pdf.lly 3192 {
pdf.urx 3193     currentpoint
pdf.ury 3194     /pdf.lly exch def
        3195     /pdf.llx exch def
        3196 }
        3197 def
        3198 /pdf.save.ur
        3199 {
        3200     currentpoint
        3201     /pdf.ury exch def
        3202     /pdf.urx exch def
        3203 }
        3204 def
        3205 /pdf.save.linkll
        3206 {
        3207     currentpoint
        3208     pdf.linkmargin add
        3209     pdf.linkdp.pad add
        3210     /pdf.lly exch def
        3211     pdf.linkmargin sub
        3212     /pdf.llx exch def
        3213 }
        3214 def
        3215 /pdf.save.linkur
        3216 {
        3217     currentpoint
        3218     pdf.linkmargin sub
        3219     pdf.linkht.pad sub
        3220     /pdf.ury exch def
        3221     pdf.linkmargin add
        3222     /pdf.urx exch def
        3223 }
        3224 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.)

```

3225 /pdf.dest.anchor
3226 {
3227     currentpoint exch
3228     pdf.dvi.pt 72 add
3229     /pdf.dest.x exch def
3230     pdf.dvi.pt
3231     vsize 72 sub exch sub
3232     /pdf.dest.y exch def
3233 }
3234 def
3235 /pdf.dest.point
3236 { pdf.dest.x pdf.dest.y } def
3237 /pdf.dest2device
3238 {
3239     /pdf.dest.y exch def
3240     /pdf.dest.x exch def
3241     matrix currentmatrix
3242     matrix defaultmatrix
3243     matrix invertmatrix
3244     matrix concatmatrix
3245     cvx exec
3246     /pdf.dev.y exch def
3247     /pdf.dev.x exch def
3248     /pdf.tmpd exch def
3249     /pdf.tmpc exch def
3250     /pdf.tmpb exch def
3251     /pdf.tmpa exch def
3252     pdf.dest.x pdf.tmpa mul
3253     pdf.dest.y pdf.tmpc mul add
3254     pdf.dev.x add
3255     pdf.dest.x pdf.tmpb mul
3256     pdf.dest.y pdf.tmpd mul add
3257     pdf.dev.y add
3258 }
3259 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	3260 /pdf.bordertracking false def
pdf.brokenlink.skip	3261 /pdf.bordertracking.begin
pdf.brokenlink.dict	3262 {
pdf.bordertracking.endpage	3263 SDict /pdf.bordertracking true put
pdf.bordertracking.continue	3264 SDict /pdf.leftboundary undef
pdf.originx	3265 SDict /pdf.rightboundary undef
pdf.originy	3266 /a where
	3267 {
	3268 /a
	3269 {

```

3270         currentpoint pop
3271         SDict /pdf.rightboundary known dup
3272         {
3273             SDict /pdf.rightboundary get 2 index lt
3274             { not }
3275             if
3276         }
3277         if
3278         { pop }
3279         { SDict exch /pdf.rightboundary exch put }
3280         ifelse
3281         moveto
3282         currentpoint pop
3283         SDict /pdf.leftboundary known dup
3284         {
3285             SDict /pdf.leftboundary get 2 index gt
3286             { not }
3287             if
3288         }
3289         if
3290         { pop }
3291         { SDict exch /pdf.leftboundary exch put }
3292         ifelse
3293     }
3294     put
3295 }
3296 if
3297 }
3298 def
3299 /pdf.bordertracking.end
3300 {
3301     /a where { /a { moveto } put } if
3302     /x where { /x { 0 exch rmoveto } put } if
3303     SDict /pdf.leftboundary known
3304     { pdf.outerbox 0 pdf.leftboundary put }
3305     if
3306     SDict /pdf.rightboundary known
3307     { pdf.outerbox 2 pdf.rightboundary put }
3308     if
3309     SDict /pdf.bordertracking false put
3310 }
3311 def
3312 /pdf.bordertracking.endpage
3313 {
3314     pdf.bordertracking
3315     {
3316         pdf.bordertracking.end
3317         true setglobal
3318         pdf.globaldict
3319         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3320         pdf.globaldict
3321         /pdf.brokenlink.skip pdf.baselineskip put
3322         pdf.globaldict
3323         /pdf.brokenlink.dict

```

```

3324         pdf.link.dict pdf.cvs put
3325     false setglobal
3326     mark pdf.link.dict cvx exec /Rect
3327     [
3328         pdf.llx
3329         pdf.lly
3330         pdf.outerbox 2 get pdf.linkmargin add
3331         currentpoint exch pop
3332         pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3333     ]
3334     /ANN pdf.pdfmark
3335 }
3336 if
3337 }
3338 def
3339 /pdf.bordertracking.continue
3340 {
3341     /pdf.link.dict pdf.globaldict
3342     /pdf.brokenlink.dict get def
3343     /pdf.outerbox pdf.globaldict
3344     /pdf.brokenlink.rect get def
3345     /pdf.baselineskip pdf.globaldict
3346     /pdf.brokenlink.skip get def
3347     pdf.globaldict dup dup
3348     /pdf.brokenlink.dict undef
3349     /pdf.brokenlink.skip undef
3350     /pdf.brokenlink.rect undef
3351     currentpoint
3352     /pdf.originy exch def
3353     /pdf.originx exch def
3354     /a where
3355     {
3356         /a
3357         {
3358             moveto
3359             SDict
3360             begin
3361                 currentpoint pdf.originy ne exch
3362                 pdf.originx ne or
3363                 {
3364                     pdf.save.linkll
3365                     /pdf.lly
3366                     pdf.lly pdf.outerbox 1 get sub def
3367                     pdf.bordertracking.begin
3368                 }
3369                 if
3370                 end
3371             }
3372             put
3373         }
3374         if
3375         /x where
3376         {
3377             /x

```

```

3378     {
3379         0 exch rmoveto
3380         SDict
3381         begin
3382         currentpoint
3383         pdf.originy ne exch pdf.originx ne or
3384         {
3385             pdf.save.linkll
3386             /pdf.lly
3387             pdf.lly pdf.outerbox 1 get sub def
3388             pdf.bordertracking.begin
3389         }
3390         if
3391         end
3392     }
3393     put
3394 }
3395 if
3396 }
3397 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.

```

3398 /pdf.breaklink
3399 {
3400     pop
3401     counttomark 2 mod 0 eq
3402     {
3403         counttomark /pdf.count exch def
3404         {
3405             pdf.count 0 eq { exit } if
3406             counttomark 2 roll
3407             1 index /Rect eq
3408             {
3409                 dup 4 array copy
3410                 dup dup
3411                 1 get
3412                 pdf.outerbox pdf.rect.ht
3413                 pdf.linkmargin 2 mul add sub
3414                 3 exch put
3415                 dup
3416                 pdf.outerbox 2 get
3417                 pdf.linkmargin add
3418                 2 exch put
3419                 dup dup
3420                 3 get
3421                 pdf.outerbox pdf.rect.ht
3422                 pdf.linkmargin 2 mul add add
3423                 1 exch put

```



```

3424 /pdf.currentrect exch def
3425 pdf.breaklink.write
3426 {
3427     pdf.currentrect
3428     dup
3429     pdf.outerbox 0 get
3430     pdf.linkmargin sub
3431     0 exch put
3432     dup
3433     pdf.outerbox 2 get
3434     pdf.linkmargin add
3435     2 exch put
3436     dup dup
3437     1 get
3438     pdf.baselineskip add
3439     1 exch put
3440     dup dup
3441     3 get
3442     pdf.baselineskip add
3443     3 exch put
3444     /pdf.currentrect exch def
3445     pdf.breaklink.write
3446 }
3447 1 index 3 get
3448 pdf.linkmargin 2 mul add
3449 pdf.outerbox pdf.rect.ht add
3450 2 index 1 get sub
3451 pdf.baselineskip div round cvi 1 sub
3452 exch
3453 repeat
3454 pdf.currentrect
3455 dup
3456 pdf.outerbox 0 get
3457 pdf.linkmargin sub
3458 0 exch put
3459 dup dup
3460 1 get
3461 pdf.baselineskip add
3462 1 exch put
3463 dup dup
3464 3 get
3465 pdf.baselineskip add
3466 3 exch put
3467 dup 2 index 2 get 2 exch put
3468 /pdf.currentrect exch def
3469 pdf.breaklink.write
3470 SDict /pdf.pdfmark.good false put
3471 exit
3472 }
3473 { pdf.count 2 sub /pdf.count exch def }
3474 ifelse
3475 }
3476 loop
3477 }

```

```

3478   if
3479   /ANN
3480 }
3481 def
3482 /pdf.breaklink.write
3483 {
3484   counttomark 1 sub
3485   index /_objdef eq
3486   {
3487     counttomark -2 roll
3488     dup wcheck
3489     {
3490       readonly
3491       counttomark 2 roll
3492     }
3493     { pop pop }
3494   } ifelse
3495 }
3496 if
3497 counttomark 1 add copy
3498 pop pdf.currentrect
3499 /ANN pdfmark
3500 }
3501 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.

```

3502 /pdf.pdfmark
3503 {
3504   SDict /pdf.pdfmark.good true put
3505   dup /ANN eq
3506   {
3507     pdf.pdfmark.store
3508     pdf.pdfmark.dict
3509     begin
3510       Subtype /Link eq
3511       currentdict /Rect known and
3512       SDict /pdf.outerbox known and
3513       SDict /pdf.baselineskip known and
3514       {
3515         Rect 3 get
3516         pdf.linkmargin 2 mul add
3517         pdf.outerbox pdf.rect.ht add
3518         Rect 1 get sub
3519         pdf.baselineskip div round cvi 0 gt
3520         { pdf.breaklink }
3521       } if
3522     }
3523   } if

```

```

3524         end
3525         SDict /pdf.outerbox undef
3526         SDict /pdf.baselineskip undef
3527         currentdict /pdf.pdfmark.dict undef
3528     }
3529     if
3530     pdf.pdfmark.good
3531     { pdfmark }
3532     { cleartomark }
3533     ifelse
3534 }
3535 def
3536 /pdf.pdfmark.store
3537 {
3538     /pdf.pdfmark.dict 65534 dict def
3539     counttomark 1 add copy
3540     pop
3541     {
3542         dup mark eq
3543         {
3544             pop
3545             exit
3546         }
3547         {
3548             pdf.pdfmark.dict
3549             begin def end
3550         }
3551     } ifelse
3552 }
3553 loop
3554 }
3555 def

```

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

```

3556 </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>	1211, 1230, 1253, 1275, 1291, 1392, 1629, 1665, 2233, 2279	<code>_box_backend_scale:Nnn</code>	252, <u>311</u> , 355, <u>432</u>
<code>\bool_gset_true:N</code>	1209, 1278, 1390, 1644, 2226, 2232	<code>\l_box_backend_sin_fp</code>	<u>283</u>
<code>\bool_if:NTF</code>	67, 703, 1221, 1225, 1241, 1244, 1248, 1259, 1266, 1270, 1282, 1286, 1403, 1408, 1413, 1603, 1648, 1761, 1796, 1906, 1948, 2221, 2236, 2241, 2246	<code>\g_box_clip_path_int</code>	<u>369</u>
<code>\bool_if:nTF</code>	2455, 2721, 2975	C	
<code>\bool_lazy_and:nnTF</code>	901, 975, 3065, 3092	char commands:	
<code>\bool_lazy_or:nnTF</code>	1788, 1941	<code>\char_set_catcode_space:n</code>	154
<code>\bool_new:N</code>	1212, 1279, 1393, 1645, 2206, 2207	clist commands:	
<code>\bool_set_false:N</code>	1771, 1873, 1966, 2030	<code>\clist_map_function:nN</code> ...	1299, 1423
box commands:		<code>\clist_map_function:nn</code>	1672
<code>\box_dp:N</code>	224, 226, 274, 276, 331, 333, 380, 382, 384, 386, 2258, 2291, 2292, 2317	color internal commands:	
<code>\box_ht:N</code>	226, 276, 333, 384, 386, 1808, 2003, 2263, 2302, 2303, 2319	<code>_color_backend:nnn</code>	<u>1097</u>
<code>\box_if_empty:N</code>	2352	<code>_color_backend_cmyk:w</code>	<u>1098</u>
<code>\box_move_down:nn</code>	2180, 2258	<code>\g_color_backend_colorant_prop</code>	670, 689, 692, 711, 918
<code>\box_move_up:nn</code>	2182, 2263	<code>_color_backend_devicen_colorants:n</code>	671, 872, 973
<code>\box_new:N</code>	2065, 2170, 2171	<code>_color_backend_devicen_colorants:w</code>	671
<code>\box_set_dp:Nn</code>	1728	<code>_color_backend_devicen_init:nnn</code>	859, <u>943</u>
<code>\box_set_ht:Nn</code>	1727	<code>_color_backend_devicen_init:w</code>	<u>943</u>
<code>\box_set_wd:Nn</code>	288, 1726	<code>_color_backend_fill:n</code>	1004, 1031, 1061, 1079, <u>1086</u>
<code>\box_use:N</code>	231, 249, 263, 279, 306, 320, 336, 352, 364, 415, 429, 448, 1343, 1538, 1729, 2211	<code>_color_backend_fill_cmyk:n</code>	1004, <u>1038</u> , 1061, <u>1086</u>
<code>\box_wd:N</code>	225, 233, 275, 281, 332, 338, 381, 383, 1807, 2002	<code>_color_backend_fill_devicen:nn</code>	1030, 1052, <u>1078</u> , <u>1148</u>
box internal commands:		<code>_color_backend_fill_gray:n</code>	1004, <u>1038</u> , 1061, <u>1086</u>
<code>_box_backend_clip:N</code>	213, 268, 325, <u>369</u>	<code>_color_backend_fill_rgb:n</code>	1004, <u>1038</u> , 1061, <u>1086</u>
<code>\l_box_backend_cos_fp</code>	<u>283</u>	<code>_color_backend_fill_separation:nn</code>	1030, <u>1038</u> , <u>1078</u> , <u>1148</u>
		<code>\l_color_backend_fill_tl</code>	637, 647, 1012, 1027
		<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, 1015, 1028, <u>1038</u> , 1070, 1095
		<code>_color_backend_rgb:w</code>	1121
		<code>_color_backend_select:n</code> ..	619, 699

_color_backend_select:nn .	639 , 887	_color_backend_stroke_gray:n ..	1004 , 1061 , 1097
_color_backend_select_cmyk:n ..	619 , 639 , 656	_color_backend_stroke_gray-	
_color_backend_select_devicen:nn	698 , 880 , 886 , 996	aux:n	1097
_color_backend_select_gray:n ..	619 , 639 , 656	_color_backend_stroke_rgb:n ..	1004 , 1061 , 1097
_color_backend_select_rgb:n ...	619 , 639 , 656	_color_backend_stroke_rgb:w .	1097
_color_backend_select_separation:nn	698 , 880 , 886 , 996	_color_backend_stroke_separation:nn	1030 , 1038 , 1078 , 1148
_color_backend_separation-		\l_color_backend_stroke_tl	637 , 648 , 1014 , 1025
init:n	701	\g_color_model_int	708 , 717 , 865 , 909 , 983
_color_backend_separation-		\c_color_model_range_CIELAB_tl .	820 , 855 , 932 , 939
init:nn	889	color.sc	619 , 3175
_color_backend_separation-		cs commands:	
init:nnn	701	\cs_generate_variant:Nn ..	49 , 63 , 66 , 99 , 138 , 143 , 170 , 201 , 207 , 569 , 606 , 722 , 1158 , 1353 , 1547 , 1920 , 1977 , 1993 , 2069 , 2106 , 2165 , 2657 , 2685 , 2795 , 2817 , 2852 , 3062 , 3130
_color_backend_separation-		\cs_gset:Npx ..	2467 , 2471 , 2980 , 2985
init:nnnn	701	\cs_gset_eq:NN	663 , 664 , 999 , 1045 , 1046 , 1052 , 1054 , 1056
_color_backend_separation-		\cs_gset_protected:Npn	551 , 658 , 665 , 998 , 1040 , 1047 , 1049 , 1051 , 3096 , 3135 , 3144
init:nwn	701	\cs_if_exist:NTF	27 , 50 , 457 , 480 , 539 , 2348 , 2746 , 2772
_color_backend_separation-		\cs_if_exist_p:N .	902 , 976 , 3066 , 3093
init:w	701	\cs_if_exist_use:NTF	38 , 735
_color_backend_separation-		\cs_new:Npn	686 , 744 , 746 , 748 , 750 , 757 , 763 , 765 , 771 , 788 , 795 , 797 , 988 , 1304 , 1428 , 1676 , 2006 , 2015 , 2059 , 2084 , 2166 , 2168 , 2201 , 2373 , 2473 , 2474 , 2627 , 2658 , 2659 , 2777 , 2810 , 2853 , 2855 , 2871 , 2988 , 2989 , 2999 , 3004 , 3005 , 3010 , 3011
_color_backend_separation-		\cs_new:Npx	671 , 2494 , 2529 , 2686 , 2697 , 2764 , 2901
init_/DeviceCMYK:nnn	701	\cs_new_eq:NN ..	46 , 57 , 59 , 700 , 888 , 1034 , 1035 , 1082 , 1083 , 1150 , 1151 , 1157 , 1352 , 1358 , 1359 , 1546 , 1553 , 1738 , 1767 , 1818 , 1819 , 1861 , 1869 , 1891 , 1962 , 2019 , 2026 , 2058 , 2211
_color_backend_separation-		\cs_new_protected:Npn	47 , 54 , 61 , 64 , 72 , 78 , 83 , 85 , 89 , 100 , 110 , 119 , 128 , 141 , 144 , 146 , 148 , 168 , 173 , 182 , 192 , 202 , 213 , 235 , 237 , 252 , 268 , 283 , 285 , 311 , 325 , 340 , 342 , 355 , 369 , 419 , 432 , 456 , 474 , 479 , 487 , 517 , 560 , 570 , 582 , 596 , 607 , 619 ,
_color_backend_separation-			
init_/DeviceGray:nnn	701		
_color_backend_separation-			
init_/DeviceRGB:nnn	701		
_color_backend_separation-			
init_aux:nnnnnn	701		
_color_backend_separation-			
init_CIELAB:nnn	701 , 882 , 889		
_color_backend_separation-			
init_CIELAB:nnnnnn	883		
_color_backend_separation-			
init_count:n	701		
_color_backend_separation-			
init_count:w	701		
_color_backend_separation-			
init_Device:Nn	701		
\g_color_backend_stack_int ...	516		
\l_color_backend_stack_int	513 , 541 , 547 , 649 , 653 , 1013 , 1026		
_color_backend_stroke:n	1004 , 1033 , 1038		
_color_backend_stroke_cmyk:n ..	1004 , 1061 , 1097		
_color_backend_stroke_cmyk:w	1097		
_color_backend_stroke_devicen:nn	1030 , 1056 , 1078 , 1148		

621, 623, 625, 633, 639, 641, 643,
 645, 652, 698, 723, 813, 859, 880,
 881, 882, 883, 886, 889, 914, 921,
 943, 1004, 1006, 1008, 1010, 1017,
 1019, 1021, 1023, 1030, 1032, 1061,
 1063, 1065, 1067, 1072, 1074, 1076,
 1078, 1080, 1086, 1088, 1090, 1092,
 1097, 1099, 1110, 1118, 1120, 1122,
 1148, 1149, 1159, 1164, 1169, 1171,
 1173, 1181, 1189, 1198, 1208, 1210,
 1213, 1215, 1232, 1237, 1255, 1277,
 1280, 1293, 1306, 1311, 1313, 1315,
 1317, 1319, 1321, 1323, 1325, 1330,
 1354, 1356, 1360, 1365, 1370, 1380,
 1389, 1391, 1394, 1396, 1398, 1400,
 1405, 1410, 1415, 1417, 1430, 1435,
 1437, 1439, 1441, 1443, 1445, 1447,
 1449, 1460, 1485, 1497, 1509, 1521,
 1528, 1548, 1554, 1559, 1564, 1575,
 1585, 1595, 1597, 1599, 1601, 1632,
 1634, 1639, 1641, 1643, 1646, 1667,
 1678, 1691, 1693, 1695, 1697, 1699,
 1701, 1703, 1705, 1707, 1715, 1739,
 1753, 1768, 1780, 1785, 1813, 1825,
 1838, 1848, 1863, 1870, 1878, 1889,
 1893, 1896, 1911, 1921, 1956, 1963,
 1969, 1975, 1978, 1985, 1994, 1999,
 2007, 2020, 2027, 2033, 2035, 2037,
 2048, 2067, 2070, 2072, 2076, 2086,
 2107, 2112, 2117, 2122, 2132, 2137,
 2145, 2173, 2178, 2210, 2212, 2217,
 2219, 2224, 2239, 2244, 2281, 2310,
 2329, 2338, 2375, 2382, 2408, 2413,
 2441, 2453, 2465, 2469, 2475, 2477,
 2481, 2505, 2507, 2509, 2520, 2540,
 2550, 2573, 2587, 2597, 2608, 2629,
 2660, 2708, 2719, 2725, 2753, 2787,
 2789, 2796, 2798, 2802, 2812, 2818,
 2823, 2828, 2833, 2835, 2837, 2845,
 2858, 2874, 2876, 2899, 2909, 2911,
 2933, 2938, 2971, 2973, 2978, 2983,
 2990, 2992, 2996, 2997, 2998, 3000,
 3001, 3002, 3003, 3006, 3007, 3008,
 3009, 3012, 3013, 3019, 3024, 3029,
 3036, 3043, 3076, 3081, 3098, 3100,
 3106, 3112, 3164, 3166, 3168, 3170

 \cs_new_protected:Npx
 520, 701, 1133, 2736, 2793, 2878

 \cs_set:Npn 152

 \cs_set_eq:NN 2369, 2370

 \cs_set_protected:Npn 459, 482

D

dim commands:

\dim_eval:n 2176, 2411,
 2489, 2490, 2491, 2548, 2583, 2584,
 2585, 2865, 2866, 2867, 2910, 2936
 \dim_max:nn 2289, 2300
 \dim_set:Nn ... 1807, 1808, 2002, 2003
 \dim_to_decimal:n .. 380, 381, 382,
 383, 384, 386, 1557, 1562, 1568,
 1569, 1570, 1571, 1580, 1581, 1582,
 1673, 1692, 2053, 2054, 2287, 2298,
 2316, 2317, 2318, 2319, 2323, 2379
 \dim_to_decimal_in_bp:n
 224, 225, 226, 274, 275, 276,
 331, 332, 333, 1177, 1178, 1185,
 1186, 1193, 1194, 1202, 1203, 1204,
 1301, 1305, 1309, 1363, 1368, 1374,
 1375, 1376, 1384, 1385, 1425, 1429,
 1433, 1677, 1744, 1745, 1746, 1747,
 1883, 1884, 1885, 1886, 1935, 1936,
 1937, 1938, 2042, 2043, 2044, 2045

draw internal commands:

__draw_align_currentpoint... .. 36
 __draw_backend_add_to_path:n ...
 1554, 1600
 __draw_backend_begin:
 1159, 1354, 1548
 __draw_backend_box_use:Nnnnn ...
 31, 1330, 1528, 1715
 __draw_backend_cap_but:
 1293, 1417, 1667
 __draw_backend_cap_rectangle: ..
 1293, 1417, 1667
 __draw_backend_cap_round:
 1293, 1417, 1667
 __draw_backend_clip: 1213, 1394, 1599
 __draw_backend_closepath:
 1213, 1394, 1599
 __draw_backend_closestroke: ...
 1213, 1394, 1599
 __draw_backend_cm:nnnn 1325, 1338,
 1339, 1340, 1449, 1532, 1707, 1718
 __draw_backend_cm_aux:nnnn .. 1449
 __draw_backend_cm_decompose:nnnnN
 1455, 1484
 __draw_backend_cm_decompose_-
 auxi:nnnnN 1484
 __draw_backend_cm_decompose_-
 auxii:nnnnN 1484
 __draw_backend_cm_decompose_-
 auxiii:nnnnN 1484
 __draw_backend_curveto:nnnnnn ..
 1173, 1360, 1554

<code>__draw_backend_dash:n</code>	1293 , 1417 , 1667
<code>__draw_backend_dash_aux:nn</code> ..	1667
<code>__draw_backend_dash_pattern:nn</code> .	1293 , 1417 , 1667
<code>__draw_backend_discardpath:</code> ...	1213 , 1394 , 1599
<code>__draw_backend_end:</code> 1159 , 1354 , 1548	
<code>__draw_backend_evenodd_rule:</code> ...	1208 , 1389 , 1595
<code>__draw_backend_fill:</code> 1213 , 1394 , 1599	
<code>__draw_backend_fillstroke:</code>	1213 , 1394 , 1599
<code>__draw_backend_join_bevel:</code>	1293 , 1417 , 1667
<code>__draw_backend_join_miter:</code>	1293 , 1417 , 1667
<code>__draw_backend_join_round:</code>	1293 , 1417 , 1667
<code>__draw_backend_lineto:nn</code>	1173 , 1360 , 1554
<code>__draw_backend_linewidth:n</code>	1293 , 1417 , 1667
<code>__draw_backend_literal:n</code>	1157 , 1162 , 1166 , 1170 , 1172 , 1175 , 1183 , 1191 , 1200 , 1214 , 1217 , 1218 , 1219 , 1220 , 1223 , 1229 , 1239 , 1246 , 1252 , 1257 , 1262 , 1263 , 1264 , 1265 , 1268 , 1274 , 1284 , 1290 , 1295 , 1308 , 1312 , 1314 , 1316 , 1318 , 1320 , 1322 , 1324 , 1327 , 1332 , 1333 , 1334 , 1335 , 1336 , 1337 , 1341 , 1342 , 1344 , 1345 , 1346 , 1347 , 1348 , 1352 , 1362 , 1367 , 1372 , 1382 , 1395 , 1397 , 1399 , 1402 , 1407 , 1412 , 1416 , 1419 , 1432 , 1436 , 1438 , 1440 , 1442 , 1444 , 1446 , 1448 , 1546 , 1606 , 1625 , 1651
<code>__draw_backend_miterlimit:n</code> ...	1293 , 1417 , 1667
<code>__draw_backend_moveto:nn</code>	1173 , 1360 , 1554
<code>__draw_backend_nonzero_rule:</code> ...	1208 , 1389 , 1595
<code>__draw_backend_path:n</code>	1599
<code>__draw_backend_rectangle:nmmn</code> ..	1173 , 1360 , 1554
<code>__draw_backend_scope:n</code> 1596 , 1598 , 1618 , 1658 , 1680 , 1692 , 1694 , 1696 , 1698 , 1700 , 1702 , 1704 , 1706 , 1709	
<code>__draw_backend_scope_begin:</code> ...	1169 , 1355 , 1358
<code>__draw_backend_scope_end:</code>	1169 , 1357 , 1358
<code>__draw_backend_stroke:</code>	1213 , 1394 , 1599
<code>\g__draw_clip_path_int</code>	1605 , 1608 , 1621 , 1650 , 1653 , 1661
<code>\g__draw_draw_clip_bool</code> ..	1213 , 1599
<code>\g__draw_draw_eor_bool</code>	1208 , 1225 , 1241 , 1248 , 1259 , 1270 , 1286 , 1389 , 1403 , 1408 , 1413
<code>\g__draw_draw_path_int</code>	1599
<code>\g__draw_draw_path_tl</code>	1554 , 1610 , 1626 , 1628 , 1655 , 1664
<code>\g__draw_path_int</code>	1614 , 1631
E	
<code>\errmessage</code>	38
<code>\evensidemargin</code>	2256
exp commands:	
<code>\exp_after:wN</code>	159 , 465 , 2013
<code>\exp_args:Ne</code>	759 , 2410 , 2935
<code>\exp_args:Nf</code>	1298 , 1422 , 2175
<code>\exp_args:NNf</code>	236 , 284 , 341
<code>\exp_args:Nnx</code>	2162 , 2848
<code>\exp_args:NV</code>	461
<code>\exp_args:Nx</code> .	705 , 899 , 1831 , 1852 , 2119 , 2134 , 2252 , 2814 , 3021 , 3078
<code>\exp_last_unbraced:Nx</code>	470 , 484
<code>\exp_not:N</code>	522 , 523 , 531 , 533 , 673 , 679 , 680 , 681 , 707 , 708 , 711 , 712 , 717 , 2496 , 2498 , 2501 , 2531 , 2533 , 2536 , 2688 , 2690 , 2693 , 2699 , 2701 , 2704 , 2741 , 2742 , 2748 , 2749 , 2768 , 2773 , 2882 , 2890 , 2906
<code>\exp_not:n</code> ..	48 , 97 , 108 , 136 , 2110 , 2115 , 2404 , 2643 , 2644 , 2658 , 2659 , 2671 , 2672 , 2826 , 2831 , 2842 , 2915
<code>\ExplBackendFileDate</code>	1
F	
file commands:	
<code>\file_compare_timestamp:nNnTF</code> .	1840
<code>\file_parse_full_name:nNNN</code> .	1827 , 1850
<code>\fmtversion</code>	52
fp commands:	
<code>\fp_compare:nNnTF</code>	243 , 290 , 296 , 348 , 1465 , 1478 , 1523
<code>\fp_eval:n</code> .	236 , 245 , 258 , 259 , 284 , 301 , 316 , 318 , 341 , 350 , 361 , 362 , 426 , 441 , 442 , 1105 , 1106 , 1107 , 1115 , 1128 , 1129 , 1130 , 1467 , 1472 , 1473 , 1480 , 1490 , 1491 , 1492 , 1493 , 1502 , 1503 , 1504 , 1505 , 1514 , 1515 , 1516 , 1517 , 2401 , 2570 , 2929 , 3022 , 3032 , 3039 , 3079 , 3103 , 3110 , 3171
<code>\fp_new:N</code>	309 , 310

`\fp_set:Nn` 289, 292
`\fp_use:N` 295, 299, 304
`\fp_zero:N` 291
`\c_zero_fp` 243, 290, 296, 348, 1465, 1478

G

graphics commands:

`\graphics_bb_restore:nTF` . 1782, 1996
`\graphics_bb_save:n` 1811, 2004
`\l_graphics_decodearray_tl`
..... 1759, 1760,
1770, 1790, 1794, 1795, 1872, 1904,
1905, 1943, 1946, 1947, 1965, 2029
`\graphics_extract_bb:n`
..... 1867, 1874, 2024, 2031
`\l_graphics_interpolate_bool` ...
..... 1761, 1771, 1789, 1796,
1873, 1906, 1942, 1948, 1966, 2030
`\l_graphics_llx_dim`
..... 1744, 1883, 1935, 2042
`\l_graphics_lly_dim`
..... 1745, 1884, 1936, 2043
`\l_graphics_name_tl` 1845
`\l_graphics_page_int`
..... 1755, 1775, 1776, 1800,
1801, 1865, 1902, 1903, 1929, 1930,
1958, 1971, 1972, 2011, 2012, 2022
`\l_graphics_pagebox_tl`
..... 52, 1756, 1774,
1802, 1803, 1866, 1900, 1901, 1931,
1933, 1959, 1980, 1981, 2013, 2023
`\graphics_read_bb:n` . 1738, 1861, 2019
`\l_graphics_urx_dim`
.. 1746, 1807, 1885, 1937, 2002, 2044
`\l_graphics_ury_dim` .. 1747, 1808,
1886, 1938, 2003, 2045, 2053, 2054

graphics internal commands:

`\l_graphics_backend_dir_str` . 1820
`\l_graphics_backend_ext_str` . 1820
`__graphics_backend_getbb_auxi:n`
..... 1753
`__graphics_backend_getbb_-
auxi:nN` 1956
`__graphics_backend_getbb_-
auxii:n` 1753
`__graphics_backend_getbb_-
auxii:nnN` 1956
`__graphics_backend_getbb_-
auxiii:nNnn` 1956
`__graphics_backend_getbb_-
auxiv:nnNnn` 1956
`__graphics_backend_getbb_-
auxv:nNnn` 1956

`__graphics_backend_getbb_-
auxvi:nNnn` 1997, 1999
`__graphics_backend_getbb_eps:n` .
..... 1738, 1820, 1861, 2019
`__graphics_backend_getbb_eps:nm`
..... 1820
`__graphics_backend_getbb_eps:nn`
..... 1831, 1838
`__graphics_backend_getbb_jpg:n` .
..... 1753, 1861, 1956, 2020
`__graphics_backend_getbb_-
pagebox:w` 1956, 2013
`__graphics_backend_getbb_pdf:n` .
..... 1753, 1846, 1861, 1956, 2027
`__graphics_backend_getbb_png:n` .
..... 1753, 1861, 1956, 2020
`__graphics_backend_include:nn` 2033
`__graphics_backend_include_-
auxi:nn` 1878
`__graphics_backend_include_-
auxii:nnn` 1878
`__graphics_backend_include_-
auxiii:nnn` 1878
`__graphics_backend_include_-
bitmap_quote:w` 2007, 2048
`__graphics_backend_include_-
eps:n` 1739, 1820, 1878, 2033
`__graphics_backend_include_-
jpg:n` 1813, 1878, 2048
`__graphics_backend_include_-
pdf:n` .. 1813, 1852, 1878, 2007, 2033
`__graphics_backend_include_pdf_-
quote:w` 2010, 2015
`__graphics_backend_include_-
png:n` 1813, 1878, 2048
`\l_graphics_backend_name_str` . 1820
`\l_graphics_graphics_attr_tl` ...
..... 1752, 1757,
1764, 1772, 1782, 1809, 1811, 1816
`\l_graphics_internal_box`
.. 1805, 1807, 1808, 2001, 2002, 2003
`\g_graphics_track_int`
..... 1877, 1923, 1924

group commands:

`\group_begin:` 151, 179, 198
`\group_end:` 164, 187
`\group_insert_after:N`
..... 631, 650, 661, 1015,
1028, 1043, 1070, 1095, 3090, 3127

H

hbox commands:

`\hbox:n` 2181, 2184,
2259, 2265, 2418, 2425, 2943, 2954

`\hbox_overlap_right:n` 231,
 263, 279, 320, 336, 364, 448, 1343, 1538
`\hbox_set:Nn` .. 1805, 2001, 2251, 2283
`\hbox_set:Nw` 2234
`\hbox_set_end:` 2249
`\hbox_unpack:N` 2370
 hook commands:
`\hook_gput_code:nnn` 55

I

int commands:
`\int_compare:nNnTF` 516,
 558, 656, 996, 1038, 1775, 1800,
 1902, 1929, 1971, 2011, 2342, 2443,
 2739, 2767, 2880, 2887, 2903, 3133
`\int_const:Nn` 157, 163, 523,
 549, 584, 1809, 1924, 2079, 2617, 2805
`\int_eval:n`
 . 565, 575, 604, 615, 755, 764, 777,
 779, 783, 796, 2467, 2471, 2717,
 2742, 2749, 2762, 2972, 2980, 2985
`\int_gincr:N` 205, 371,
 522, 1605, 1650, 1923, 2078, 2147,
 2191, 2268, 2804, 2847, 2860, 2882
`\int_gset:Nn` 180, 199, 2331
`\int_gset_eq:NN` 188, 2192, 2269, 2861
`\int_if_exist:NTF` 1913
`\int_if_odd:nTF` 2254
`\int_new:N` 171, 172,
 418, 513, 519, 1631, 1877, 2074,
 2172, 2203, 2205, 2800, 2857, 2873
`\int_set:Nn` 541
`\int_set_eq:NN` ... 176, 195, 547, 2343
`\int_step_function:nnnN` 781
`\int_use:N` 373, 404,
 531, 542, 708, 717, 865, 909, 983,
 1608, 1614, 1621, 1653, 1661, 1776,
 1801, 1816, 1903, 1916, 1928, 1930,
 2012, 2085, 2150, 2163, 2167, 2195,
 2202, 2273, 2374, 2628, 2638, 2811,
 2849, 2854, 2864, 2872, 2890, 2906
`\int_value:w`
 2496, 2531, 2688, 2699, 2717
`\int_zero:N` ... 1755, 1865, 1958, 2022

K

kernel internal commands:
`__kernel_backend_align_begin:` ..
 72, 216, 240, 255
`__kernel_backend_align_end:` ...
 72, 230, 248, 262
`__kernel_backend_first_shipout:n`
 50, 69, 526, 705

`\g__kernel_backend_header_bool` ..
 67, 703
`__kernel_backend_literal:n`
 46, 62, 65, 70,
 74, 81, 84, 86, 142, 145, 147, 149,
 169, 345, 358, 528, 553, 554, 562,
 572, 627, 634, 660, 666, 725, 861,
 1042, 1048, 1050, 1069, 1094, 1161,
 1167, 1462, 1469, 1475, 1535, 1540,
 1741, 1880, 1915, 1925, 2039, 2050,
 2794, 2910, 2972, 2976, 2981, 2986
`__kernel_backend_literal_page:n`
 100, 144, 2788, 2790, 2991, 2993
`__kernel_backend_literal_pdf:n` ..
 .. 89, 141, 271, 328, 1352, 3142, 3157
`__kernel_backend_literal_-`
`postscript:n`
 61, 75, 76, 80, 217, 218, 220,
 221, 229, 241, 256, 1157, 2445, 2457
`__kernel_backend_literal_svg:n` ..
 168, 175, 186, 194,
 204, 372, 374, 391, 1546, 1719, 1730
`__kernel_backend_matrix:n`
 128, 293, 314, 1452
`__kernel_backend_postscript:n` ..
 64,
 629, 1073, 1075, 1077, 1081, 2068,
 2124, 2139, 2181, 2187, 2227, 2259,
 2266, 2270, 2284, 2312, 2356, 2363,
 2369, 2377, 2384, 2418, 2425, 3045
`__kernel_backend_scope:n`
 173, 401, 406, 1135, 1551, 3171
`__kernel_backend_scope_begin:` ..
 83, 110, 146,
 173, 215, 239, 254, 270, 287, 313,
 327, 344, 357, 1358, 1530, 1550, 1717
`__kernel_backend_scope_begin:n` ..
 173, 393, 421, 434
`__kernel_backend_scope_end:` ...
 . 83, 110, 146, 173, 232, 250, 264,
 280, 307, 321, 337, 353, 365, 416,
 430, 449, 551, 1359, 1542, 1553, 1731
`\g__kernel_backend_scope_int` ...
 171, 178, 180, 185, 189, 197, 199, 205
`\l__kernel_backend_scope_int` ...
 171, 177, 190, 196
`__kernel_color_backend_stack_-`
`init:Nnn` 516, 582, 3069
`__kernel_color_backend_stack_-`
`pop:n` 558, 596, 653, 3099
`__kernel_color_backend_stack_-`
`push:nn` 558,
 596, 649, 1013, 1026, 3088, 3125

__kernel_dependency_version_- check:Nn	1		
__kernel_dependency_version_- check:nn	27, 29		
__kernel_kern:n			
.	2186, 2188, 2417, 2421, 2424, 2428, 2942, 2950, 2953, 2969		
\c_kernel_sys_dvipdfmx_version_- int	151, 516, 558, 656, 996, 1038, 2880, 2887, 2903, 3133		
M			
\MessageBreak	40		
mode commands:			
\mode_if_horizontal:TF . . .	2333, 2340		
\mode_if_math:TF	2231		
O			
\oddsidemargin	2255		
opacity internal commands:			
__opacity_backend:nn	3164		
__opacity_backend:nnn	3019		
__opacity_backend_fill:n	3019, 3100, 3164		
__opacity_backend_fill_stroke:nn	3102, 3108, 3112, 3130, 3144		
\l__opacity_backend_fill_tl	3074, 3083, 3109, 3117, 3137, 3149		
__opacity_backend_fillstroke:nn	3100		
__opacity_backend_reset:	3076, 3127		
__opacity_backend_select:n	3019, 3076, 3133, 3164		
__opacity_backend_select_aux:n	3019, 3076, 3115, 3135, 3147		
\c__opacity_backend_stack_int	3065, 3088, 3099, 3125		
__opacity_backend_stroke:n	3019, 3100, 3164		
\l__opacity_backend_stroke_tl	3074, 3084, 3104, 3118, 3138, 3150		
P			
pdf commands:			
\pdf_object_if_exist:nTF	923		
\pdf_object_new:nn	925		
\pdf_object_ref:n	938		
\pdf_object_ref_last:	910, 917, 919, 972, 984		
\pdf_object_unnamed_write:nn	891, 916, 945, 967		
\pdf_object_write:nn	926		
		pdf internal commands:	
		__pdf_backend:n	2793, 2797, 2799, 2825, 2830, 2839, 2862, 2884, 2900, 2913, 2945, 2946, 2956
		__pdf_backend_annotation:nnnn	2173, 2481, 2858
		__pdf_backend_annotation_- aux:nnnn	2175, 2178
		\g__pdf_backend_annotation_int	2172, 2192, 2202, 2857, 2861, 2872
		__pdf_backend_annotation_last:	2201, 2494, 2871
		__pdf_backend_bdc:nn	2475, 2787, 2990, 3012
		__pdf_backend_catalog_gput:nn	2070, 2587, 2796, 2996
		__pdf_backend_compress_objects:n	2441, 2708, 2971, 3006
		__pdf_backend_compresslevel:n	2441, 2708, 2971, 3006
		\l__pdf_backend_content_box	2170, 2234, 2258, 2261, 2263, 2292, 2303
		__pdf_backend_destination:nn	2382, 2550, 2911
		__pdf_backend_destination:nnnn	2382, 2550, 2911
		__pdf_backend_destination_- aux:nnnn	2382, 2911
		__pdf_backend_emc:	2475, 2787, 2990, 3012
		__pdf_backend_info_gput:nn	2070, 2587, 2796, 2996
		__pdf_backend_link:nw	2212
		__pdf_backend_link_aux:nw	2212
		__pdf_backend_link_begin:n	2874
		__pdf_backend_link_begin:nnnw	2505
		__pdf_backend_link_begin:nw	2214, 2218, 2219
		__pdf_backend_link_begin_aux:nw	2222, 2224
		__pdf_backend_link_begin_- goto:nnw	2212, 2505, 2874
		__pdf_backend_link_begin_- user:nnw	2212, 2505, 2874
		\g__pdf_backend_link_bool	2207, 2221, 2226, 2241, 2279
		\g__pdf_backend_link_dict_tl	2204, 2229, 2274
		__pdf_backend_link_end:	2212, 2505, 2874
		__pdf_backend_link_end_aux:	2212
		\g__pdf_backend_link_int	2203, 2269, 2273, 2374, 2873, 2882, 2890, 2906

__pdf_backend_link_last:	__pdf_backend_version_major: ...
..... 2373, 2529, 2901 2467,
__pdf_backend_link_margin:n ...	2473, 2764, 2980, 2981, 2988, 3010
..... 2375, 2540, 2909	__pdf_backend_version_major_-
\g__pdf_backend_link_math_bool ..	gset:n 2465, 2736, 2978, 3008
..... 2206, 2232, 2233, 2236, 2246	__pdf_backend_version_minor: ...
__pdf_backend_link_minima: .. 2212 2471,
__pdf_backend_link_outerbox:n 2212	2473, 2764, 2985, 2986, 2988, 3010
\g__pdf_backend_link_sf_int	__pdf_backend_version_minor_-
..... 2205, 2331, 2342, 2343	gset:n 2465, 2736, 2978, 3008
__pdf_backend_link_sf_restore: 2212	\l__pdf_breaklink_pdfmark_tl ...
__pdf_backend_link_sf_save: . 2212 2208, 2276, 2368
\l__pdf_backend_model_box . 2171,	__pdf_breaklink_postscript:n ...
2251, 2283, 2291, 2302, 2317, 2319 2210, 2260, 2262, 2369
__pdf_backend_objcompresslevel:n	__pdf_breaklink_usebox:N
..... 2708 2211, 2261, 2370
\g__pdf_backend_object_int	__pdf_exp_not_i:nn . 2629, 2675, 2680
..... 2074, 2078, 2081,	__pdf_exp_not_ii:nn 2629, 2676, 2681
2147, 2150, 2163, 2167, 2191, 2192,	\l__pdf_internal_box
2195, 2268, 2269, 2800, 2804, 2807,	pdf.baselineskip
2847, 2849, 2854, 2860, 2861, 2864	pdf.bordertracking
__pdf_backend_object_last:	pdf.bordertracking.begin
..... 2166, 2686, 2853, 2998	pdf.bordertracking.continue
__pdf_backend_object_new:nn ...	pdf.bordertracking.end
..... 2076, 2608, 2802, 2998	pdf.bordertracking.endpage
__pdf_backend_object_now:nn ...	pdf.breaklink
..... 2145, 2660, 2845, 2998	pdf.breaklink.write
\g__pdf_backend_object_prop	pdf.brokenlink.dict
..... 2074, 2082, 2093, 2103,	pdf.brokenlink.rect
2607, 2625, 2641, 2800, 2808, 2815	pdf.brokenlink.skip
__pdf_backend_object_ref:n 2076,	pdf.count
2090, 2104, 2608, 2802, 2821, 2998	pdf.currentrect
__pdf_backend_object_write:nn ..	pdf.cvs
..... 2086, 2629, 2812, 2998	pdf.dest.anchor
__pdf_backend_object_write:nnn 2812	pdf.dest.point
__pdf_backend_object_write_-	pdf.dest.x
array:nn	pdf.dest.y
2086, 2812	pdf.dest2device
__pdf_backend_object_write_-	pdf.dev.x
dict:nn	pdf.dev.y
2086, 2812	pdf.dvi.pt
__pdf_backend_object_write_-	pdf.globaldict
fstream:nn	pdf.leftboundary
2086, 2812	pdf.link.dict
__pdf_backend_object_write_-	pdf.linkdp.pad
fstream:nnn	2212, 3186
2120, 2122	pdf.linkht.pad
__pdf_backend_object_write_-	2212, 3186
stream:nn	pdf.linkmargin
2086, 2812	3186
__pdf_backend_object_write_-	pdf.llx
stream:nnn	2212, 3189
2086	pdf.lly
__pdf_backend_object_write_-	2212, 3189
stream:nnnn	pdf.originx
2812	3260
__pdf_backend_pageobject_ref:n .	pdf.originy
..... 2168, 2697, 2855, 2998	3260
__pdf_backend_pdfmark:n	pdf.outerbox
2067, 2071, 2073, 2088, 2109, 2114,	2212, 3502
2148, 2193, 2385, 2429, 2476, 2478	pdf.pdfmark
	3502
	pdf.pdfmark.dict
	3502

pdf.pdfmark.good	3502	skip commands:	
pdf.pt.dvi	3182	\skip_horizontal:n	233, 281, 338
pdf.rect	3189	str commands:	
pdf.rect.ht	3182	\c_hash_str	404, 1614, 1621, 1661
pdf.rightboundary	3260	\c_percent_str	1141, 1142, 1143
pdf.save.linkll	3189	\str_case:nn	957, 2152, 2668
pdf.save.linkur	3189	\str_case:nnTF	2389, 2559, 2918
pdf.save.ll	3189	\str_case:e:nn	2092, 2640
pdf.save.ur	3189	\str_convert_pdfname:n	712, 732, 900
pdf.tmpa	3225	\str_if_eq:nnTF	
pdf.tmpb	3225		490, 493, 496, 499, 3114, 3146
pdf.tmpc	3225	\str_new:N	1822, 1823, 1824
pdf.tmpd	3225	\str_tail:N	1833, 1854
pdf.urx	3189	sys commands:	
pdf.ury	2212, 3189	\sys_get_shell:nnNTF	153
pdfmanagement commands:		\sys_if_shell:TF	1820
\pdfmanagement_add:nnn		\sys_shell_now:n	1842
	907, 981, 3071,	sys internal commands:	
	3085, 3119, 3122, 3139, 3151, 3154	\l_sys_internal_tl	155, 159
\pdfmanagement_if_active_p:	902,	__sys_tmp:w	152, 159
	903, 976, 977, 3066, 3067, 3093, 3094		
prg commands:		T	
\prg_replicate:nn		TeX and L ^A T _E X 2 _ε commands:	
	184, 753, 774, 784, 951	\@cclv	2352, 2354, 2362
prop commands:		\@ifl@t@r	50, 52
\prop_gput:Nnn		\@makecol@hook	2346
	711, 918, 2082, 2625, 2808	\current@color	14, 461, 465, 471, 485
\prop_if_in:NnTF	689	\special	2
\prop_item:Nn		tex commands:	
	692, 2093, 2103, 2641, 2815	\tex_baselineskip:D	2323
\prop_new:N	670, 2075, 2607, 2801	\tex_endinput:D	44
\ProvidesExplFile	2	\tex_global:D	
			2710, 2727, 2741, 2748, 2755
Q		\tex_immediate:D	
quark commands:			1787, 2632, 2635, 2663, 2666
\quark_if_recursion_tail_stop:n	688	\tex_luatexversion:D	2739, 2767
\q_recursion_stop	681	\tex_pdfannot:D	2487
\q_recursion_tail	680	\tex_pdftocatalog:D	2593
\q_stop	152, 160	\tex_pdfcolorstack:D	602, 613
		\tex_pdfcolorstackinit:D	590
S		\tex_pdfcompresslevel:D	2715
scan commands:		\tex_pdfdest:D	2556, 2579
\scan_stop:		\tex_pdfendlink:D	2526
	113, 122, 615, 2523, 2548, 2571,	\tex_pdfextension:D	
	2585, 2717, 2734, 2742, 2749, 2762		92, 103, 113, 122, 131,
scan internal commands:			599, 610, 2484, 2512, 2523, 2553,
\s_color_stop			2576, 2590, 2600, 2611, 2632, 2663
	471, 474, 485, 488, 764, 765,	\tex_pdffeedback:D	
	769, 773, 786, 789, 793, 797, 811,		587, 2498, 2533, 2620, 2690, 2701
	952, 988, 992, 1098, 1100, 1121, 1123	\tex_pdfinfo:D	2603
\s_graphics_stop		\tex_pdflastannot:D	2501
	2010, 2015, 2055, 2059	\tex_pdflastlink:D	2536
separation	3176	\tex_pdflastobj:D	2623, 2693
		\tex_pdflastximage:D	1806, 1810

