

File I

Implementation

1 l3backend-basics Implementation

```
1 <*package>
```

Whilst there is a reasonable amount of code overlap between backends, it is much clearer to have the blocks more-or-less separated than run in together and DocStripped out in parts. As such, most of the following is set up on a per-backend basis, though there is some common code (again given in blocks not interspersed with other material).

All the file identifiers are up-front so that they come out in the right place in the files.

```
2 \ProvidesExplFile
3 <*dvipdfmx>
4   {l3backend-dvipdfmx.def}{2022-04-14}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2022-04-14}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2022-04-14}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2022-04-14}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2022-04-14}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2022-04-14}{ }
25   {L3 backend support: XeTeX}
26 </xetex>
```

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

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

```

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

```

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

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

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

```

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

```

(End definition for `__kernel_backend_literal:e`.)

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

```

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

```

(End definition for `__kernel_backend_first_shipout:n`.)

1.1 dvips backend

```

60 <*dvips>

```

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

```

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

```

(End definition for `_kernel_backend_literal_postscript:n`.)

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

`_kernel_backend_postscript:x`

```

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

```

(End definition for `_kernel_backend_postscript:n`.)

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

```

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

```

`_kernel_backend_align_begin:`

`_kernel_backend_align_end:`

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

```

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

```

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

`_kernel_backend_scope_begin:`

`_kernel_backend_scope_end:`

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

```

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

```

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

```

87 </dvips>

```

1.2 LuaTeX and pdfTeX backends

88 `<*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 </dvipdfmx | xetex>

```

1.4 dvisvgm backend

151 $\langle *dvisvgm \rangle$

_kernel_backend_literal_svg:n
_kernel_backend_literal_svg:x

Unlike the other backends, the requirements for making SVG files mean that we can't conveniently transform all operations to the current point. That makes life a bit more tricky later as that needs to be accounted for. A new line is added after each call to help to keep the output readable for debugging.

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

(End definition for _kernel_backend_literal_svg:n.)

\g__kernel_backend_scope_int
\l__kernel_backend_scope_int

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

155 \int_new:N \g__kernel_backend_scope_int
156 \int_new:N \l__kernel_backend_scope_int

(End definition for \g__kernel_backend_scope_int and \l__kernel_backend_scope_int.)

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

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

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

```

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

(End definition for \__kernel_backend_scope_begin: and others.)

192 </dvisvgm>
193 </package>

```

2 l3backend-box Implementation

```

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

```

2.1 dvips backend

```

196 <*dvips>

```

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

```

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

```

(End definition for `__box_backend_clip:N`.)

`__box_backend_rotate:Nn` Rotating using `dvips` does not require that the box dimensions are altered and has a very convenient built-in operation. Zero rotation must be written as 0 not -0 so there is a quick test.

`__box_backend_rotate_aux:Nn`

```

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

```

(End definition for __box_backend_rotate:Nn and __box_backend_rotate_aux:Nn.)

__box_backend_scale:Nnn The dvips backend once again has a dedicated operation we can use here.

```

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

```

(End definition for __box_backend_scale:Nnn.)

250 </dvips>

2.2 LuaTeX and pdfTeX backends

251 <*luatex | pdftex>

__box_backend_clip:N The general method is to save the current location, define a clipping path equivalent to the bounding box, then insert the content at the current position and in a zero width box. The “real” width is then made up using a horizontal skip before tidying up. There are other approaches that can be taken (for example using XForm objects), but the logic here shares as much code as possible and uses the same conversions (and so same rounding errors) in all cases.

```

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

```



```

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

```

(End definition for __box_backend_clip:N.)

__box_backend_rotate:Nn Rotations are set using an affine transformation matrix which therefore requires sine/cosine values not the angle itself. We store the rounded values to avoid rounding twice. There are also a couple of comparisons to ensure that -0 is not written to the output, as this avoids any issues with problematic display programs. Note that numbers are compared to 0 after rounding.

```

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

```

(End definition for __box_backend_rotate:Nn and others.)

__box_backend_scale:Nnn The same idea as for rotation but without the complexity of signs and cosines.

```

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

```

```

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

```

(End definition for __box_backend_scale:Nnn.)

```

307 </luatex | pdftex>

```

2.3 dvipdfmx/X_YTeX backend

```

308 <*dvipdfmx | xetex>

```

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

```

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

```

(End definition for __box_backend_clip:N.)

__box_backend_rotate:Nn Rotating in dvipdfmx/X_YTeX can be implemented using either PDF or backend-specific code. The former approach however is not “aware” of the content of boxes: this means that any embedded links would not be adjusted by the rotation. As such, the backend-native approach is preferred: the code therefore is similar (though not identical) to the dvips version (notice the rotation angle here is positive). As for dvips, zero rotation is written as 0 not -0.

```

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

```

```

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

```

(End definition for `__box_backend_rotate:Nn` and `__box_backend_rotate_aux:Nn`.)

`__box_backend_scale:Nnn` Much the same idea for scaling: use the higher-level backend operation to allow for box content.

```

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

```

(End definition for `__box_backend_scale:Nnn`.)

```

351 </dviPDFmx | xetex>

```

2.4 dvisvgm backend

```

352 <*dvisvgm>

```

`__box_backend_clip:N`
`\g__kernel_clip_path_int`

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

```

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

```

```

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

```

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

```

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

```

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

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

```

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

```

```

414 \__kernel_backend_scope_end:
415 }

```

(End definition for __box_backend_rotate:Nn.)

__box_backend_scale:Nnn

In contrast to rotation, we have to account for the current position in this case. That is done using a couple of translations in addition to the scaling (which is therefore done backward with a flip).

```

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

```

(End definition for __box_backend_scale:Nnn.)

435 </dvisvgm>

436 </package>

3 l3backend-color Implementation

437 <*package>

438 <@@=color>

Color support is split into parts: collecting data from L^AT_EX 2_ε, the color stack, general color, separations, and color for drawings. We have different approaches in each backend, and have some choices to make about dvipdfmx/X_YL_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

439 <*dvisvgm | dvipdfmx | dvips | xetex>

__color_backend_pickup:N

__color_backend_pickup:w

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.

```

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

```

```

442 {
443   \cs_set_protected:Npn \__color_backend_pickup:N #1
444   {
445     \exp_args:NV \tl_if_head_is_space:nTF \current@color
446     {
447       \tl_set:Nx #1
448       {
449         { named }
450         { \exp_after:wN \use:n \current@color }
451       }
452     }
453     {
454       \exp_last_unbraced:Nx \__color_backend_pickup:w
455       { \current@color } \s__color_stop #1
456     }
457   }
458   \cs_new_protected:Npn \__color_backend_pickup:w #1 ~ #2 \s__color_stop #3
459   { \tl_set:Nn #3 { {#1} {#2} } }
460 }

```

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

```

461 </dvipsvm | dvipdfmx | dvips | xetex>

```

3.1.2 LuaTeX and pdfTeX

```

462 <*luatex | pdftex>

```

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

```

463 \cs_new_protected:Npn \__color_backend_pickup:N #1 { }
464 \cs_if_exist:cT { ver@color.sty }
465 {
466   \cs_set_protected:Npn \__color_backend_pickup:N #1
467   {
468     \exp_last_unbraced:Nx \__color_backend_pickup:w
469     { \current@color } ~ 0 ~ 0 ~ 0 \s__color_stop #1
470   }
471   \cs_new_protected:Npn \__color_backend_pickup:w
472   #1 ~ #2 ~ #3 ~ #4 ~ #5 ~ #6 \s__color_stop #7
473   {
474     \str_if_eq:nnTF {#2} { g }
475     { \tl_set:Nn #7 { { gray } {#1} } }
476     {
477       \str_if_eq:nnTF {#4} { rg }
478       { \tl_set:Nn #7 { { rgb } { #1 ~ #2 ~ #3 } } }
479       {
480         \str_if_eq:nnTF {#5} { k }
481         { \tl_set:Nn #7 { { cmyk } { #1 ~ #2 ~ #3 ~ #4 } } }
482         {
483           \str_if_eq:nnTF {#2} { cs }
484           {

```

```

485         \tl_set:Nx #7 { { \use:n #1 } { #5 } }
486     }
487     {
488         \tl_set:Nn #7 { { gray } { 0 } }
489     }
490 }
491 }
492 }
493 }
494 }

```

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

```

495 </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/XYTeX` the backend version.

3.2.1 Common code

```

496 <*dvipdfmx | luatex | pdftex | xetex>

```

`__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.

```

497 \int_new:N \__color_backend_stack_int

```

(End definition for `__color_backend_stack_int`.)

```

498 </dvipdfmx | luatex | pdftex | xetex>

```

3.2.2 dvipdfmx/X_YTeX

```

499 <*dvipdfmx | xetex>

```

`__kernel_color_backend_stack_init:Nnn` 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.

```

\__color_backend_stack_int
\c_color_backend_main_stack_int
500 \int_new:N \g__color_backend_stack_int
501 \cs_new_protected:Npx \__kernel_color_backend_stack_init:Nnn #1#2#3
502 {
503     \int_gincr:N \exp_not:N \g__color_backend_stack_int
504     \int_const:Nn #1 { \exp_not:N \g__color_backend_stack_int }
505     \use:x
506     {
507         \__kernel_backend_first_shipout:n
508         {
509             \__kernel_backend_literal:n
510             {
511                 pdfcolorstackinit ~
512                 \exp_not:N \int_use:N \exp_not:N \g__color_backend_stack_int
513                 \c_space_tl
514                 \exp_not:N \tl_if_blank:nF {#2} { #2 ~ }
515                 (#3)
516             }

```

```

517     }
518   }
519 }
520 \cs_if_exist:cTF { main@pdfcolorstack }
521 {
522   \int_set:Nn \l__color_backend_stack_int
523     { \int_use:c { main@pdfcolorstack } }
524 }
525 {
526   \__kernel_color_backend_stack_init:Nnn \c__color_backend_main_stack_int
527     { page ~ direct } { 0 ~ g ~ 0 ~ G }
528   \int_set_eq:NN \l__color_backend_stack_int
529     \c__color_backend_main_stack_int
530   \int_const:cn { main@pdfcolorstack } { \c__color_backend_main_stack_int }
531 }

```

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.

```

532 \cs_gset_protected:Npn \__kernel_backend_scope_end:
533 {
534   \__kernel_backend_literal:n { x:grestore }
535   \__kernel_backend_literal:x
536   {
537     pdfcolorstack ~
538     \int_use:N \g__color_backend_stack_int \c_space_tl current
539   }
540 }

```

(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.

```

541 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
542 {
543   \__kernel_backend_literal:x
544   {
545     pdfcolorstack ~
546     \int_eval:n {#1} ~
547     push ~ (#2)
548   }
549 }
550 \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
551 \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
552 {
553   \__kernel_backend_literal:x
554   {
555     pdfcolorstack ~
556     \int_eval:n {#1} ~
557     pop
558   }
559 }

```

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

```

560 </dvipdfmx | xetex>

```


3.2.3 LuaTeX and pdfTeX

561 $\langle *luatex \mid pdftex \rangle$

$\backslash_kernel_color_backend_stack_init:Nnn$

```

562 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
563 {
564   \int_const:Nn #1
565   {
566      $\langle *luatex \rangle$ 
567     \tex_pdffeedback:D colorstackinit ~
568    $\langle /luatex \rangle$ 
569      $\langle *pdftex \rangle$ 
570     \tex_pdfcolorstackinit:D
571    $\langle /pdftex \rangle$ 
572     \tl_if_blank:nF {#2} { #2 ~ }
573     {#3}
574   }
575 }
```

(End definition for $\backslash_kernel_color_backend_stack_init:Nnn$.)

$\backslash_kernel_color_backend_stack_push:nn$

$\backslash_kernel_color_backend_stack_push:nx$

$\backslash_kernel_color_backend_stack_pop:n$

```

576 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
577 {
578    $\langle *luatex \rangle$ 
579   \tex_pdfextension:D colorstack ~
580  $\langle /luatex \rangle$ 
581    $\langle *pdftex \rangle$ 
582   \tex_pdfcolorstack:D
583  $\langle /pdftex \rangle$ 
584   \int_eval:n {#1} ~ push ~ {#2}
585 }
586 \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
587 \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
588 {
589    $\langle *luatex \rangle$ 
590   \tex_pdfextension:D colorstack ~
591  $\langle /luatex \rangle$ 
592    $\langle *pdftex \rangle$ 
593   \tex_pdfcolorstack:D
594  $\langle /pdftex \rangle$ 
595   \int_eval:n {#1} ~ pop \scan_stop:
596 }
```

(End definition for $\backslash_kernel_color_backend_stack_push:nn$ and $\backslash_kernel_color_backend_stack_pop:n$.)

597 $\langle /luatex \mid pdftex \rangle$

3.3 General color

3.3.1 dvips-style

598 $\langle *dvips \mid dvisvgm \rangle$

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_named:n
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc

```

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

```

599 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
600 { \__color_backend_select:n { cmyk ~ #1 } }
601 \cs_new_protected:Npn \__color_backend_select_gray:n #1
602 { \__color_backend_select:n { gray ~ #1 } }
603 \cs_new_protected:Npn \__color_backend_select_named:n #1
604 { \__color_backend_select:n { ~ #1 } }
605 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
606 { \__color_backend_select:n { rgb ~ #1 } }
607 \cs_new_protected:Npn \__color_backend_select:n #1
608 {
609   \__kernel_backend_literal:n { color~push~ #1 }
610 }
611 \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
612 </dvips>
613 }
614 \cs_new_protected:Npn \__color_backend_reset:
615 { \__kernel_backend_literal:n { color~pop } }

```

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

```

616 </dvips | dvisvgm>

```

3.3.2 LuaTeX and pdfTeX

```

617 <*dvipdfmx | luatex | pdftex | xetex>

```

```

\l__color_backend_fill_tl
\l__color_backend_stroke_tl

```

```

618 \tl_new:N \l__color_backend_fill_tl
619 \tl_new:N \l__color_backend_stroke_tl

```

(End definition for \l__color_backend_fill_tl and \l__color_backend_stroke_tl.)

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_named:n
\__color_backend_select_rgb:n
\__color_backend_select:nn
\__color_backend_reset:

```

Store the values then pass to the stack.

```

620 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
621 { \__color_backend_select:nn { #1 ~ k } { #1 ~ K } }
622 \cs_new_protected:Npn \__color_backend_select_gray:n #1
623 { \__color_backend_select:nn { #1 ~ g } { #1 ~ G } }
624 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
625 { \__color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
626 \cs_new_protected:Npn \__color_backend_select:nn #1#2
627 {
628   \tl_set:Nn \l__color_backend_fill_tl {#1}
629   \tl_set:Nn \l__color_backend_stroke_tl {#2}
630   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int { #1 ~ #2 }
631 }
632 \cs_new_protected:Npn \__color_backend_reset:
633 { \__kernel_color_backend_stack_pop:n \l__color_backend_stack_int }

```

(End definition for __color_backend_select_cmyk:n and others.)

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

```

634 \cs_new_protected:Npn \_color_backend_select_named:n #1
635 {
636   \str_if_eq:nnTF {#1} { Black }
637     { \_color_backend_select_gray:n { 0 } }
638     { \msg_error:nnn { color } { unknown-named-color } {#1} }
639 }
640 \msg_new:nnn { color } { unknown-named-color }
641 { Named-color~'#1'~is-not-known. }

```

(End definition for `_color_backend_select_named:n`.)

```

642 </dvipdfmx | luatex | pdftex | xetex>

```

3.3.3 dvipdfmx/X_YTeX

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).

3.4 Separations

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

```

643 <*dvipdfmx | luatex | pdftex | xetex | dvips>

```

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

`\g_color_backend_colorant_prop`

```

644 \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`

```

645 \cs_new:Npx \_color_backend_devicen_colorants:n #1
646 {
647   \exp_not:N \tl_if_blank:nF {#1}
648   {
649     \c_space_tl
650     << ~
651     /Colorants ~
652     << ~
653     \exp_not:N \_color_backend_devicen_colorants:w #1 ~
654     \exp_not:N \q_recursion_tail \c_space_tl
655     \exp_not:N \q_recursion_stop
656     >> ~
657     >>
658   }
659 }
660 \cs_new:Npn \_color_backend_devicen_colorants:w #1 ~
661 {

```

```

662 \quark_if_recursion_tail_stop:n {#1}
663 \prop_if_in:NnT \g__color_backend_colorant_prop {#1}
664 {
665     #1 ~
666     \prop_item:Nn \g__color_backend_colorant_prop {#1} ~
667 }
668 \__color_backend_devicen_colorants:w
669 }

```

(End definition for __color_backend_devicen_colorants:n and __color_backend_devicen_colorants:w.)

```

670 </dvipdfmx | luatex | pdftex | xetex | dvips>
671 <*dvips>

```

```

\__color_backend_select_separation:nn
\__color_backend_select_devicen:nn

```

```

672 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
673 { \__color_backend_select:n { separation ~ #1 ~ #2 } }
674 \cs_new_eq:NN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn

```

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

```

\__color_backend_select_iccbased:nn

```

No support.

```

675 \cs_new_protected:Npn \__color_backend_select_iccbased:nn #1#2 { }

```

(End definition for __color_backend_select_iccbased:nn.)

```

\__color_backend_separation_init:nnnnn
\__color_backend_separation_init:nxxnn
\__color_backend_separation_init_aux:nnnnnn
\__color_backend_separation_init_DeviceCMYK:nnnn
\__color_backend_separation_init_DeviceGray:nnnn
\__color_backend_separation_init_DeviceRGB:nnnn
\__color_backend_separation_init_Device:NN
\__color_backend_separation_init:nnnn
\__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:nnnn

```

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.

```

676 \cs_new_protected:Npx \__color_backend_separation_init:nnnnn #1#2#3#4#5
677 {
678     \bool_if:NT \g__kernel_backend_header_bool
679     {
680         \exp_args:Nx \__kernel_backend_first_shipout:n
681         {
682             \exp_not:N \__color_backend_separation_init_aux:nnnnnn
683             { \exp_not:N \int_use:N \g__color_model_int }
684             {#1} {#2} {#3} {#4} {#5}
685         }
686         \prop_gput:Nxx \exp_not:N \g__color_backend_colorant_prop
687         { / \exp_not:N \str_convert_pdfname:n {#1} }
688         {
689             << ~
690             /setcolorspace ~ {} ~
691             >> ~ begin ~
692             color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
693             end
694         }
695     }
696 }
697 \cs_generate_variant:Nn \__color_backend_separation_init:nnnnn { nxx }
698 \cs_new_protected:Npn \__color_backend_separation_init_aux:nnnnnn #1#2#3#4#5#6
699 {
700     \__kernel_backend_literal:e

```

```

701 {
702 !
703 TeXDict ~ begin ~
704 /color #1
705 {
706 [ ~
707 /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
708 [ ~ #3 ~ ] ~
709 {
710 \cs_if_exist_use:cF { __color_backend_separation_init_ #3 :nnn }
711 { \__color_backend_separation_init:nnn }
712 {#4} {#5} {#6}
713 }
714 ] ~ setcolorspace
715 } ~ def ~
716 end
717 }
718 }
719 \cs_new:cpn { __color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
720 { \__color_backend_separation_init_Device:Nn 4 {#3} }
721 \cs_new:cpn { __color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
722 { \__color_backend_separation_init_Device:Nn 1 {#3} }
723 \cs_new:cpn { __color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
724 { \__color_backend_separation_init_Device:Nn 2 {#3} }
725 \cs_new:Npn \__color_backend_separation_init_Device:Nn #1#2
726 {
727 #2 ~
728 \prg_replicate:nn {#1}
729 { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
730 \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
731 }

```

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.

```

732 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3
733 {
734 \exp_args:Ne \__color_backend_separation_init:nnnn
735 { \__color_backend_separation_init_count:n {#2} }
736 {#1} {#2} {#3}
737 }
738 \cs_new:Npn \__color_backend_separation_init_count:n #1
739 { \int_eval:n { 0 \__color_backend_separation_init_count:w #1 ~ \s_color_stop } }
740 \cs_new:Npn \__color_backend_separation_init_count:w #1 ~ #2 \s_color_stop
741 {
742 +1
743 \tl_if_blank:nF {#2}
744 { \__color_backend_separation_init_count:w #2 \s_color_stop }
745 }

```

Now we implement the algorithm. In the terms in the PostScript manual, we have **N** = 1 and **Domain** = [0 1], with **Range** as #2, **C0** as #3 and **C1** as #4, with the number of output components in #1. So all we have to do is implement $y_i = C0_i + x(C1_i - C0_i)$ 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 **C0** and **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.

```

746 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
747 {
748   \__color_backend_separation_init:w #3 ~ \s__color_stop #4 ~ \s__color_stop
749   \prg_replicate:nn {#1}
750   {
751     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
752     \int_eval:n { 3 * #1 } ~ index ~ mul ~
753     2 ~ index ~ add ~
754     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
755   }
756   \int_step_function:nnnN {#1} { -1 } { 1 }
757   \__color_backend_separation_init:n
758   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
759   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
760   \tl_if_blank:nF {#2}
761   { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
762 }
763 \cs_new:Npn \__color_backend_separation_init:w
764 #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
765 {
766   #1 ~ #3 ~ 0 ~
767   \tl_if_blank:nF {#2}
768   { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }
769 }
770 \cs_new:Npn \__color_backend_separation_init:n #1
771 { \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.

```

772 \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s__color_stop
773 {
774   #2 ~ #3 ~
775   2 ~ index ~ 2 ~ index ~ lt ~
776   { ~ pop ~ exch ~ pop ~ } ~
777   { ~
778     2 ~ index ~ 1 ~ index ~ gt ~
779     { ~ exch ~ pop ~ exch ~ pop ~ } ~
780     { ~ pop ~ pop ~ } ~
781     ifelse ~
782   }
783   ifelse ~
784   #1 ~ 1 ~ roll ~
785   \tl_if_blank:nF {#4}
786   { \__color_backend_separation_init:nw {#1} #4 \s__color_stop }
787 }

```

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.

```

788 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
789 {
790   \__color_backend_separation_init:nxxnn
791   {#2}
792   {
793     /CIEBasedABC ~
794     << ~
795     /RangeABC ~ [ ~ \c__color_model_range_CIELAB_tl \c_space_tl ] ~
796     /DecodeABC ~
797     [ ~
798       { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
799       { ~ 500 ~ div ~ } ~ bind ~
800       { ~ 200 ~ div ~ } ~ bind ~
801     ] ~
802     /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
803     /DecodeLMN ~
804     [ ~
805     { ~
806       dup ~ 6 ~ 29 ~ div ~ ge ~
807       { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
808       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
809       ifelse ~
810       0.9505 ~ mul ~
811     } ~ bind ~
812     { ~
813       dup ~ 6 ~ 29 ~ div ~ ge ~
814       { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
815       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
816       ifelse ~
817     } ~ bind ~
818     { ~
819       dup ~ 6 ~ 29 ~ div ~ ge ~
820       { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
821       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
822       ifelse ~
823       1.0890 ~ mul ~
824     } ~ bind
825     ] ~
826     /WhitePoint ~
827     [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
828     >>
829   }
830   { \c__color_model_range_CIELAB_tl }
831   { 100 ~ 0 ~ 0 }
832   {#3}
833 }

```

(End definition for __color_backend_separation_init:nnnnn and others.)

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

```

834 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
835 {

```

```

836 \__kernel_backend_literal:e
837 {
838   !
839   TeXDict ~ begin ~
840   /color \int_use:N \g__color_model_int
841   {
842     [ ~
843     /DeviceN ~
844     [ ~ #1 ~ ] ~
845     #2 ~
846     { ~ #3 ~ } ~
847     \__color_backend_devicen_colorants:n {#1}
848   ] ~ setcolorspace
849   } ~ def ~
850   end
851 }
852 }

```

(End definition for __color_backend_devicen_init:nnn.)

__color_backend_iccbased_init:nnn No support at present.

```

853 \cs_new_protected:Npn \__color_backend_iccbased_init:nnn #1#2#3 { }

```

(End definition for __color_backend_iccbased_init:nnn.)

```

854 </dvips>

```

```

855 <*dvisvgm>

```

__color_backend_select_separation:nn No support at present.

__color_backend_select_devicen:nn

```

856 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2 { }

```

```

857 \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 No support at present.

__color_backend_separation_init_CIELAB:nnn

```

858 \cs_new_protected:Npn \__color_backend_separation_init:nnnnn #1#2#3#4#5 { }

```

```

859 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnnnn #1#2#3 { }

```

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

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

```

860 \cs_new_protected:Npn \__color_backend_select_iccbased:nn #1#2
861 {
862   \__kernel_backend_literal_svg:x
863   {
864     <style>
865     @color-profile ~
866     \str_if_eq:nnTF {#2} { cmyk }
867     { device-cmyk }
868     { --color \int_use:N \g__color_model_int }
869     \c_space_tl
870     {
871       src:("#1")

```



```

872         }
873     </style>
874 }
875 }

```

(End definition for `_color_backend_select_iccbased:nn`.)

```

876 </dvisvgm>
877 <*dvipdfmx | luatex | pdftex | xetex>

```

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

```

878 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
879 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
880 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
881 \cs_new_eq:NN \_color_backend_select_iccbased:nn \_color_backend_select_separation:nn

```

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

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.

```

882 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5
883 {
884     \pdf_object_unnamed_write:nx { dict }
885     {
886         /FunctionType ~ 2
887         /Domain ~ [0 ~ 1]
888         \tl_if_blank:nF {#3} { /Range ~ [#3] }
889         /C0 ~ [#4] ~
890         /C1 ~ [#5] /N ~ 1
891     }
892     \exp_args:Nx \_color_backend_separation_init:nn
893     { \str_convert_pdfname:n {#1} } {#2}
894     \bool_lazy_and:nnT
895     { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
896     { \pdfmanagement_if_active_p: }
897     {
898         \use:x
899         {
900             \pdfmanagement_add:nnn
901             { Page / Resources / ColorSpace }
902             { color \int_use:N \g__color_model_int }
903             { \pdf_object_ref_last: }
904         }
905     }
906 }
907 \cs_new_protected:Npn \_color_backend_separation_init:nn #1#2
908 {
909     \pdf_object_unnamed_write:nx { array }
910     { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
911     \prop_gput:Nnx \g__color_backend_colorant_prop { /#1 }

```

```

912     { \pdf_object_ref_last: }
913 }

```

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

```

914 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
915 {
916   \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
917   {
918     \pdf_object_new:nn { __color_illuminant_CIELAB_ #1 } { array }
919     \pdf_object_write:nx { __color_illuminant_CIELAB_ #1 }
920     {
921       /Lab ~
922       <<
923       /WhitePoint ~
924       [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
925       /Range ~ [ \c__color_model_range_CIELAB_tl ]
926       >>
927     }
928   }
929   \__color_backend_separation_init:nnnnn
930   {#2}
931   { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
932   { \c__color_model_range_CIELAB_tl }
933   { 100 ~ 0 ~ 0 }
934   {#3}
935 }

```

(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.

```

936 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
937 {
938   \pdf_object_unnamed_write:nx { stream }
939   {
940     {
941       /FunctionType ~ 4 ~
942       /Domain ~
943       [ ~
944         \prg_replicate:nn
945         { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
946         { 0 ~ 1 ~ }
947       ] ~
948       /Range ~
949       [ ~
950         \str_case:nn {#2}
951         {
952           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
953           { /DeviceGray } { 0 ~ 1 }
954           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
955         } ~
956       ]
957     }
958   }

```

```

958     { {#3} }
959   }
960   \pdf_object_unnamed_write:nx { array }
961   {
962     /DeviceN ~
963     [ ~ #1 ~ ] ~
964     #2 ~
965     \pdf_object_ref_last:
966     \__color_backend_devicen_colorants:n {#1}
967   }
968   \bool_lazy_and:nnT
969   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
970   { \pdfmanagement_if_active_p: }
971   {
972     \use:x
973     {
974       \pdfmanagement_add:nnn
975       { Page / Resources / ColorSpace }
976       { color \int_use:N \g__color_model_int }
977       { \pdf_object_ref_last: }
978     }
979   }
980 }
981 \cs_new:Npn \__color_backend_devicen_init:w #1 ~ #2 \s__color_stop
982 {
983   + 1
984   \tl_if_blank:nF {#2}
985   { \__color_backend_devicen_init:w #2 \s__color_stop }
986 }

```

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

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

```

987 \cs_new_protected:Npn \__color_backend_iccbased_init:nnn #1#2#3
988 {
989   \pdf_object_if_exist:nF { __color_icc_ #1 }
990   {
991     \pdf_object_new:nn { __color_icc_ #1 } { fstream }
992     \pdf_object_write:nx { __color_icc_ #1 }
993     {
994       {
995         /N ~ \exp_not:n { #2 } ~
996         \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
997       }
998       {#1}
999     }
1000   }
1001   \pdf_object_unnamed_write:nx { array }
1002   { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
1003   \cs_if_exist:NT \pdfmanagement_add:nnn
1004   {
1005     \use:x
1006     {
1007       \pdfmanagement_add:nnn { Page / Resources / ColorSpace }

```

```

1008         { color \int_use:N \g__color_model_int }
1009         { ~ \pdf_object_ref_last: }
1010     }
1011 }
1012 }

```

(End definition for _color_backend_iccbased_init:nnn.)

_color_backend_iccbased_device:nnn

This is very similar to setting up a color space: the only part we skip is adding it to the page resources.

```

1013 \cs_new_protected:Npn \_color_backend_iccbased_device:nnn #1#2#3
1014 {
1015     \pdf_object_if_exist:nF { __color_icc_ #1 }
1016     {
1017         \pdf_object_new:nn { __color_icc_ #1 } { fstream }
1018         \pdf_object_write:nn { __color_icc_ #1 }
1019         {
1020             { /N ~ #3 }
1021             {#1}
1022         }
1023     }
1024     \pdf_object_unnamed_write:nx { array }
1025     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
1026     \cs_if_exist:NT \pdfmanagement_add:nnn
1027     {
1028         \use:x
1029         {
1030             \pdfmanagement_add:nnn
1031             { Page / Resources / ColorSpace }
1032             { Default #2 }
1033             { \pdf_object_ref_last: }
1034         }
1035     }
1036 }

```

(End definition for _color_backend_iccbased_device:nnn.)

```

1037 </dvipdfmx | luatex | pdftex | 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.

```

1038 < *dvipdfmx | luatex | pdftex | xetex>

```

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

```

1039 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1040 { \_color_backend_fill:n { #1 ~ k } }
1041 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1042 { \_color_backend_fill:n { #1 ~ g } }

```

```

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

```

```

1043 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1044 { \__color_backend_fill:n { #1 ~ rg } }
1045 \cs_new_protected:Npn \__color_backend_fill:n #1
1046 {
1047   \tl_set:Nn \l__color_backend_fill_tl {#1}
1048   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1049     { #1 ~ \l__color_backend_stroke_tl }
1050   \group_insert_after:N \__color_backend_reset:
1051 }
1052 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1053 { \__color_backend_stroke:n { #1 ~ K } }
1054 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1055 { \__color_backend_stroke:n { #1 ~ G } }
1056 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1057 { \__color_backend_stroke:n { #1 ~ RG } }
1058 \cs_new_protected:Npn \__color_backend_stroke:n #1
1059 {
1060   \tl_set:Nn \l__color_backend_stroke_tl {#1}
1061   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1062     { \l__color_backend_fill_tl \c_space_tl #1 }
1063   \group_insert_after:N \__color_backend_reset:
1064 }

```

(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
1065 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
1066 { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
1067 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1068 { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
1069 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1070 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

1071 </dviptfm | luatex | pdftex | xetex>
1072 <*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
1073 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1074 { \__color_backend_fill:n { cmyk ~ #1 } }
1075 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1076 { \__color_backend_fill:n { gray ~ #1 } }
1077 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1078 { \__color_backend_fill:n { rgb ~ #1 } }
1079 \cs_new_protected:Npn \__color_backend_fill:n #1
1080 {
1081   \__kernel_backend_literal:n { color-push~ #1 }
1082   \group_insert_after:N \__color_backend_reset:
1083 }
1084 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1085 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1086 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1087 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1088 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1089 { \__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
1090 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1091   { \_color_backend_fill:n { separation ~ #1 ~ #2 } }
1092 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1093   { \_kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1094 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1095 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

```

1096 </dvips>
1097 <*dvisvgm>

```

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

```

\_color_backend_fill_gray:n
\_color_backend_fill_rgb:n
  \_color_backend_fill:n
1098 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1099   { \_color_backend_fill:n { cmyk ~ #1 } }
1100 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1101   { \_color_backend_fill:n { gray ~ #1 } }
1102 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1103   { \_color_backend_fill:n { rgb ~ #1 } }
1104 \cs_new_protected:Npn \_color_backend_fill:n #1
1105   {
1106     \_kernel_backend_literal:n { color~push~ #1 }
1107     \group_insert_after:N \_color_backend_reset:
1108   }

```

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

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

```

\_color_backend_stroke_cmyk:w
\_color_backend_stroke_gray:n
\_color_backend_stroke_gray_aux:n
\_color_backend_stroke_rgb:n
\_color_backend_stroke_rgb:w
  \_color_backend:nnn
1109 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1110   { \_color_backend_cmyk:w #1 \s_color_stop }
1111 \cs_new_protected:Npn \_color_backend_stroke_cmyk:w
1112   #1 ~ #2 ~ #3 ~ #4 \s_color_stop
1113   {
1114     \use:x
1115     {
1116       \_color_backend:nnn
1117       { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1118       { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1119       { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }
1120     }
1121   }
1122 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1123   {
1124     \use:x
1125     {
1126       \_color_backend_stroke_gray_aux:n
1127       { \fp_eval:n { 100 * (#1) } }
1128     }
1129   }
1130 \cs_new_protected:Npn \_color_backend_stroke_gray_aux:n #1

```

```

1131 { \_color_backend:nnn {#1} {#1} {#1} }
1132 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1133 { \_color_backend_rgb:w #1 \s_color_stop }
1134 \cs_new_protected:Npn \_color_backend_stroke_rgb:w
1135 #1 ~ #2 ~ #3 \s_color_stop
1136 {
1137   \use:x
1138   {
1139     \_color_backend:nnn
1140     { \fp_eval:n { 100 * (#1) } }
1141     { \fp_eval:n { 100 * (#2) } }
1142     { \fp_eval:n { 100 * (#3) } }
1143   }
1144 }
1145 \cs_new_protected:Npx \_color_backend:nnn #1#2#3
1146 {
1147   \_kernel_backend_scope:n
1148   {
1149     stroke =
1150     "
1151     rgb
1152     (
1153       #1 \c_percent_str ,
1154       #2 \c_percent_str ,
1155       #3 \c_percent_str
1156     )
1157     "
1158   }
1159 }

```

(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

```

At present, these are no-ops.

```

1160 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1161 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2 { }
1162 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1163 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

(End definition for _color_backend_fill_separation:nn and others.)

```

\_color_backend_devicen_init:nnn
\_color_backend_iccbased_init:nnn

```

No support at present.

```

1164 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3 { }
1165 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3 { }

```

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

```

1166 </dvisvgm>
1167 </package>

```

4 l3backend-draw Implementation

```

1168 <*package>
1169 <@@=draw>

```

4.1 dvips backend

1170 `<*dvips>`

`_draw_backend_literal:n` The same as literal PostScript: same arguments about positioning apply her.
`_draw_backend_literal:x` 1171 `\cs_new_eq:NN _draw_backend_literal:n _kernel_backend_literal_postscript:n`
 1172 `\cs_generate_variant:Nn _draw_backend_literal:n { x }`
(End definition for _draw_backend_literal:n.)

`_draw_backend_begin:` The `ps::[begin]` special here deals with positioning but allows us to continue on to a
`_draw_backend_end:` matching `ps::[end]`: contrast with `ps:`, which positions but where we can't split material between separate calls. The `@beginspecial/@endspecial` pair are from `special.pro` and correct the scale and *y*-axis direction. In contrast to `pgf`, we don't save the current point: discussion with Tom Rokici suggested a better way to handle the necessary translations (see `_draw_backend_box_use:Nnnnn`). (Note that `@beginspecial/@endspecial` forms a backend scope.) The `[begin]/[end]` lines are handled differently from the rest as they are conceptually different: not really drawing literals but instructions to `dvips` itself.

1173 `\cs_new_protected:Npn _draw_backend_begin:`
 1174 `{`
 1175 `_kernel_backend_literal:n { ps::[begin] }`
 1176 `_draw_backend_literal:n { @beginspecial }`
 1177 `}`
 1178 `\cs_new_protected:Npn _draw_backend_end:`
 1179 `{`
 1180 `_draw_backend_literal:n { @endspecial }`
 1181 `_kernel_backend_literal:n { ps::[end] }`
 1182 `}`

(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.

1183 `\cs_new_protected:Npn _draw_backend_scope_begin:`
 1184 `{ _draw_backend_literal:n { save } }`
 1185 `\cs_new_protected:Npn _draw_backend_scope_end:`
 1186 `{ _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.

1187 `\cs_new_protected:Npn _draw_backend_moveto:nn #1#2`
 1188 `{`
 1189 `_draw_backend_literal:x`
 1190 `{`
 1191 `\dim_to_decimal_in_bp:n {#1} ~`
 1192 `\dim_to_decimal_in_bp:n {#2} ~ moveto`
 1193 `}`
 1194 `}`


```

1195 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1196 {
1197   \__draw_backend_literal:x
1198   {
1199     \dim_to_decimal_in_bp:n {#1} ~
1200     \dim_to_decimal_in_bp:n {#2} ~ lineto
1201   }
1202 }
1203 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1204 {
1205   \__draw_backend_literal:x
1206   {
1207     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1208     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1209     moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1210   }
1211 }
1212 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1213 {
1214   \__draw_backend_literal:x
1215   {
1216     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1217     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1218     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1219     curveto
1220   }
1221 }

```

(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
1222 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1223 { \bool_gset_true:N \g__draw_draw_eor_bool }
1224 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1225 { \bool_gset_false:N \g__draw_draw_eor_bool }
1226 \bool_new:N \g__draw_draw_eor_bool

```

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

__draw_backend_closepath: Unlike PDF, PostScript doesn't track separate colors for strokes and other elements. It is also desirable to have the clip keyword after a stroke or fill. To achieve those outcomes, there is some work to do. For color, the stroke color is simple but the fill one has to be inserted by hand. For clipping, the required ordering is achieved using a T_EX switch. All of the operations end with a new path instruction as they do not terminate (again in contrast to PDF).

```

\__draw_backend_closepath:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
1227 \cs_new_protected:Npn \__draw_backend_closepath:
1228 { \__draw_backend_literal:n { closepath } }
1229 \cs_new_protected:Npn \__draw_backend_stroke:
1230 {
1231   \__draw_backend_literal:n { gsave }
1232   \__draw_backend_literal:n { color.sc }
1233   \__draw_backend_literal:n { stroke }
1234   \__draw_backend_literal:n { grestore }

```

```

1235 \bool_if:NT \g__draw_draw_clip_bool
1236 {
1237   \__draw_backend_literal:x
1238   {
1239     \bool_if:NT \g__draw_draw_eor_bool { eo }
1240     clip
1241   }
1242 }
1243 \__draw_backend_literal:n { newpath }
1244 \bool_gset_false:N \g__draw_draw_clip_bool
1245 }
1246 \cs_new_protected:Npn \__draw_backend_closestroke:
1247 {
1248   \__draw_backend_closepath:
1249   \__draw_backend_stroke:
1250 }
1251 \cs_new_protected:Npn \__draw_backend_fill:
1252 {
1253   \__draw_backend_literal:x
1254   {
1255     \bool_if:NT \g__draw_draw_eor_bool { eo }
1256     fill
1257   }
1258   \bool_if:NT \g__draw_draw_clip_bool
1259   {
1260     \__draw_backend_literal:x
1261     {
1262       \bool_if:NT \g__draw_draw_eor_bool { eo }
1263       clip
1264     }
1265   }
1266   \__draw_backend_literal:n { newpath }
1267   \bool_gset_false:N \g__draw_draw_clip_bool
1268 }
1269 \cs_new_protected:Npn \__draw_backend_fillstroke:
1270 {
1271   \__draw_backend_literal:x
1272   {
1273     \bool_if:NT \g__draw_draw_eor_bool { eo }
1274     fill
1275   }
1276   \__draw_backend_literal:n { gsave }
1277   \__draw_backend_literal:n { color.sc }
1278   \__draw_backend_literal:n { stroke }
1279   \__draw_backend_literal:n { grestore }
1280   \bool_if:NT \g__draw_draw_clip_bool
1281   {
1282     \__draw_backend_literal:x
1283     {
1284       \bool_if:NT \g__draw_draw_eor_bool { eo }
1285       clip
1286     }
1287   }
1288   \__draw_backend_literal:n { newpath }

```

```

1289     \bool_gset_false:N \g__draw_draw_clip_bool
1290   }
1291   \cs_new_protected:Npn \__draw_backend_clip:
1292     { \bool_gset_true:N \g__draw_draw_clip_bool }
1293   \bool_new:N \g__draw_draw_clip_bool
1294   \cs_new_protected:Npn \__draw_backend_discardpath:
1295     {
1296       \bool_if:NT \g__draw_draw_clip_bool
1297       {
1298         \__draw_backend_literal:x
1299         {
1300           \bool_if:NT \g__draw_draw_eor_bool { eo }
1301           clip
1302         }
1303       }
1304       \__draw_backend_literal:n { newpath }
1305       \bool_gset_false:N \g__draw_draw_clip_bool
1306     }

```

(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:
1307 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1308 {
1309   \__draw_backend_literal:x
1310   {
1311     [
1312       \exp_args:Nf \use:n
1313       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1314     ] ~
1315     \dim_to_decimal_in_bp:n {#2} ~ setdash
1316   }
1317 }
1318 \cs_new:Npn \__draw_backend_dash:n #1
1319 { ~ \dim_to_decimal_in_bp:n {#1} }
1320 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1321 {
1322   \__draw_backend_literal:x
1323   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1324 }
1325 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1326 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1327 \cs_new_protected:Npn \__draw_backend_cap_butt:
1328 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1329 \cs_new_protected:Npn \__draw_backend_cap_round:
1330 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1331 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1332 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1333 \cs_new_protected:Npn \__draw_backend_join_miter:
1334 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1335 \cs_new_protected:Npn \__draw_backend_join_round:
1336 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1337 \cs_new_protected:Npn \__draw_backend_join_bevel:
1338 { \__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/XYTeX`). Thus we take the shortest path available and simply dump the matrix as given.

```

1339 \cs_new_protected:Npn \_draw_backend_cm:nnnn #1#2#3#4
1340 {
1341   \_draw_backend_literal:n
1342     { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1343 }

```

(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).

```

1344 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
1345 {
1346   \_draw_backend_literal:n { @endspecial }
1347   \_draw_backend_literal:n { [end] }
1348   \_draw_backend_literal:n { [begin] }
1349   \_draw_backend_literal:n { save }
1350   \_draw_backend_literal:n { currentpoint }
1351   \_draw_backend_literal:n { currentpoint~translate }
1352   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1353   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1354   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1355   \_draw_backend_literal:n { neg~exch~neg~exch~translate }
1356   \_draw_backend_literal:n { [end] }
1357   \hbox_overlap_right:n { \box_use:N #1 }
1358   \_draw_backend_literal:n { [begin] }
1359   \_draw_backend_literal:n { restore }
1360   \_draw_backend_literal:n { [end] }
1361   \_draw_backend_literal:n { [begin] }
1362   \_draw_backend_literal:n { @beginspecial }
1363 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```

1364 </dvips>

```

4.2 LuaTeX, pdfTeX, dvipdfmx and X_YTeX

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

```

1365 < *dvipdfmx | luatex | pdftex | xetex >

```

4.2.1 Drawing

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

`__draw_backend_literal:x` 1366 `\cs_new_eq:NN __draw_backend_literal:n __kernel_backend_literal_pdf:n`
1367 `\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:` 1368 `\cs_new_protected:Npn __draw_backend_begin:`
1369 `{ __draw_backend_scope_begin: }`
1370 `\cs_new_protected:Npn __draw_backend_end:`
1371 `{ __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:` 1372 `\cs_new_eq:NN __draw_backend_scope_begin: __kernel_backend_scope_begin:`
1373 `\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` 1374 `\cs_new_protected:Npn __draw_backend_moveto:nn #1#2`
1375 `{`
1376 `__draw_backend_literal:x`
1377 `{ \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }`
1378 `}`
1379 `\cs_new_protected:Npn __draw_backend_lineto:nn #1#2`
1380 `{`
1381 `__draw_backend_literal:x`
1382 `{ \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }`
1383 `}`
1384 `\cs_new_protected:Npn __draw_backend_curveto:nnnnnn #1#2#3#4#5#6`
1385 `{`
1386 `__draw_backend_literal:x`
1387 `{`
1388 `\dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~`
1389 `\dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~`
1390 `\dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~`
1391 `c`
1392 `}`
1393 `}`
1394 `\cs_new_protected:Npn __draw_backend_rectangle:nnnn #1#2#3#4`
1395 `{`
1396 `__draw_backend_literal:x`
1397 `{`
1398 `\dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~`
1399 `\dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~`
1400 `re`
1401 `}`
1402 `}`

(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
1403 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1404 { \bool_gset_true:N \g__draw_draw_eor_bool }
1405 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1406 { \bool_gset_false:N \g__draw_draw_eor_bool }
1407 \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:
1408 \cs_new_protected:Npn \__draw_backend_closepath:
1409 { \__draw_backend_literal:n { h } }
1410 \cs_new_protected:Npn \__draw_backend_stroke:
1411 { \__draw_backend_literal:n { S } }
1412 \cs_new_protected:Npn \__draw_backend_closestroke:
1413 { \__draw_backend_literal:n { s } }
1414 \cs_new_protected:Npn \__draw_backend_fill:
1415 {
1416   \__draw_backend_literal:x
1417   { f \bool_if:NT \g__draw_draw_eor_bool * }
1418 }
1419 \cs_new_protected:Npn \__draw_backend_fillstroke:
1420 {
1421   \__draw_backend_literal:x
1422   { B \bool_if:NT \g__draw_draw_eor_bool * }
1423 }
1424 \cs_new_protected:Npn \__draw_backend_clip:
1425 {
1426   \__draw_backend_literal:x
1427   { W \bool_if:NT \g__draw_draw_eor_bool * }
1428 }
1429 \cs_new_protected:Npn \__draw_backend_discardpath:
1430 { \__draw_backend_literal:n { n } }

(End definition for \__draw_backend_closepath: and others.)

```

```

\__draw_backend_dash_pattern:nn Converting paths to output is again a case of mapping directly to PDF operations.
\__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:
1431 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1432 {
1433   \__draw_backend_literal:x
1434   {
1435     [
1436       \exp_args:Nf \use:n
1437       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1438     ] ~
1439     \dim_to_decimal_in_bp:n {#2} ~ d
1440   }
1441 }
1442 \cs_new:Npn \__draw_backend_dash:n #1
1443 { ~ \dim_to_decimal_in_bp:n {#1} }
1444 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1445 {
1446   \__draw_backend_literal:x

```

```

1447     { \dim_to_decimal_in_bp:n {#1} ~ w }
1448   }
1449   \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1450     { \__draw_backend_literal:x { #1 ~ M } }
1451   \cs_new_protected:Npn \__draw_backend_cap_but:
1452     { \__draw_backend_literal:n { 0 ~ J } }
1453   \cs_new_protected:Npn \__draw_backend_cap_round:
1454     { \__draw_backend_literal:n { 1 ~ J } }
1455   \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1456     { \__draw_backend_literal:n { 2 ~ J } }
1457   \cs_new_protected:Npn \__draw_backend_join_miter:
1458     { \__draw_backend_literal:n { 0 ~ j } }
1459   \cs_new_protected:Npn \__draw_backend_join_round:
1460     { \__draw_backend_literal:n { 1 ~ j } }
1461   \cs_new_protected:Npn \__draw_backend_join_bevel:
1462     { \__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/XqTeX. In the former, we have a direct method to maintain alignment: the backend can use a matrix itself. For dvipdfmx/XqTeX, 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/XqTeX, 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!

```

1463   \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1464     {
1465       <*luatex | pdftex>
1466         \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1467       </luatex | pdftex>
1468       <*dvipdfmx | xetex>
1469         \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1470         \__draw_backend_cm_aux:nnnn
1471       </dvipdfmx | xetex>
1472     }
1473   <*dvipdfmx | xetex>
1474   \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1475     {
1476       \__kernel_backend_literal:x
1477       {
1478         x:rotate~
1479         \fp_compare:nNnTF {#1} = \c_zero_fp
1480           { 0 }
1481           { \fp_eval:n { round ( -#1 , 5 ) } }
1482       }
1483       \__kernel_backend_literal:x
1484       {
1485         x:scale~
1486         \fp_eval:n { round ( #2 , 5 ) } ~
1487         \fp_eval:n { round ( #3 , 5 ) }
1488       }
1489       \__kernel_backend_literal:x
1490       {

```

```

1491      x:rotate~
1492      \fp_compare:nNnTF {#4} = \c_zero_fp
1493      { 0 }
1494      { \fp_eval:n { round ( -#4 , 5 ) } }
1495    }
1496  }
1497  \</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.

```

1498  \<dvipdfmx | xetex>
1499  \cs_new_protected:Npn \_draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1500  {
1501    \use:x
1502    {
1503      \_draw_backend_cm_decompose_auxi:nnnnN
1504      { \fp_eval:n { (#1 + #4) / 2 } }
1505      { \fp_eval:n { (#1 - #4) / 2 } }
1506      { \fp_eval:n { (#3 + #2) / 2 } }
1507      { \fp_eval:n { (#3 - #2) / 2 } }
1508    }
1509    #5
1510  }
1511  \cs_new_protected:Npn \_draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1512  {
1513    \use:x

```



```

1514 {
1515   \_draw_backend_cm_decompose_auxii:nnnnN
1516   { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1517   { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1518   { \fp_eval:n { atand ( #3 , #2 ) } }
1519   { \fp_eval:n { atand ( #4 , #1 ) } }
1520 }
1521 #5
1522 }
1523 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1524 {
1525   \use:x
1526   {
1527     \_draw_backend_cm_decompose_auxiii:nnnnN
1528     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1529     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1530     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1531     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1532   }
1533   #5
1534 }
1535 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1536 {
1537   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1538   { #5 {#1} {#2} {#3} {#4} }
1539   { #5 {#1} {#3} {#2} {#4} }
1540 }
1541 \</dvipdfmx | 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.

```

1542 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
1543 {
1544   \_kernel_backend_scope_begin:
1545   < *luatex | pdftex >
1546   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1547   < /luatex | pdftex >
1548   < *dvipdfmx | xetex >
1549   \_kernel_backend_literal:n
1550   { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1551   < /dvipdfmx | xetex >
1552   \hbox_overlap_right:n { \box_use:N #1 }
1553   < *dvipdfmx | xetex >
1554   \_kernel_backend_literal:n { pdf:etrans }
1555   < /dvipdfmx | xetex >
1556   \_kernel_backend_scope_end:
1557 }

```

(End definition for _draw_backend_box_use:Nnnnn.)

```

1558 \</dvipdfmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

1559 `<*dvisvgm>`

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

`__draw_backend_literal:x` 1560 `\cs_new_eq:NN __draw_backend_literal:n __kernel_backend_literal_svg:n`
 1561 `\cs_generate_variant:Nn __draw_backend_literal:n { x }`

(End definition for `__draw_backend_literal:n`.)

`__draw_backend_scope_begin:` Use the backend-level scope mechanisms.

`__draw_backend_scope_end:` 1562 `\cs_new_eq:NN __draw_backend_scope_begin: __kernel_backend_scope_begin:`
 1563 `\cs_new_eq:NN __draw_backend_scope_end: __kernel_backend_scope_end:`

(End definition for `__draw_backend_scope_begin:` and `__draw_backend_scope_end:`.)

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

1564 `\cs_new_protected:Npn __draw_backend_begin:`
 1565 `{`
 1566 `__kernel_backend_scope_begin:`
 1567 `__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }`
 1568 `}`
 1569 `\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_backend_path_tl` 1570 `\cs_new_protected:Npn __draw_backend_moveto:nn #1#2`
 1571 `{`
 1572 `__draw_backend_add_to_path:n`
 1573 `{ M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }`
 1574 `}`
 1575 `\cs_new_protected:Npn __draw_backend_lineto:nn #1#2`
 1576 `{`
 1577 `__draw_backend_add_to_path:n`
 1578 `{ L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }`
 1579 `}`
 1580 `\cs_new_protected:Npn __draw_backend_rectangle:nnnn #1#2#3#4`
 1581 `{`
 1582 `__draw_backend_add_to_path:n`
 1583 `{`
 1584 `M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}`
 1585 `h ~ \dim_to_decimal:n {#3} ~`
 1586 `v ~ \dim_to_decimal:n {#4} ~`
 1587 `h ~ \dim_to_decimal:n { -#3 } ~`
 1588 `Z`
 1589 `}`
 1590 `}`
 1591 `\cs_new_protected:Npn __draw_backend_curveto:nnnnnn #1#2#3#4#5#6`
 1592 `{`

```

1593 \__draw_backend_add_to_path:n
1594 {
1595     C ~
1596     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1597     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1598     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1599 }
1600 }
1601 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1602 {
1603     \tl_gset:Nx \g__draw_backend_path_tl
1604     {
1605         \g__draw_backend_path_tl
1606         \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1607         #1
1608     }
1609 }
1610 \tl_new:N \g__draw_backend_path_tl

```

(End definition for __draw_backend_moveto:nn and others.)

```

\__draw_backend_evenodd_rule: The fill rules here have to be handled as scopes.
\__draw_backend_nonzero_rule:
1611 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1612 { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1613 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1614 { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

(End definition for __draw_backend_evenodd_rule: and __draw_backend_nonzero_rule:.)

```

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

```

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

```

1615 \cs_new_protected:Npn \__draw_backend_closepath:
1616 { \__draw_backend_add_to_path:n { Z } }
1617 \cs_new_protected:Npn \__draw_backend_path:n #1
1618 {
1619     \bool_if:NTF \g__draw_draw_clip_bool
1620     {
1621         \int_gincr:N \g__kernel_clip_path_int
1622         \__draw_backend_literal:x
1623         {
1624             < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1625             { ?nl }
1626             <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1627             < /clipPath > { ? nl }
1628             <
1629             use~xlink:href =
1630             "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1631             #1
1632             />
1633         }
1634         \__kernel_backend_scope:x

```

```

1635         {
1636             clip-path =
1637                 "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1638         }
1639     }
1640     {
1641         \__draw_backend_literal:x
1642         { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1643     }
1644     \tl_gclear:N \g__draw_backend_path_tl
1645     \bool_gset_false:N \g__draw_draw_clip_bool
1646 }
1647 \int_new:N \g__draw_backend_path_int
1648 \cs_new_protected:Npn \__draw_backend_stroke:
1649 { \__draw_backend_path:n { style="fill:none" } }
1650 \cs_new_protected:Npn \__draw_backend_closestroke:
1651 {
1652     \__draw_backend_closepath:
1653     \__draw_backend_stroke:
1654 }
1655 \cs_new_protected:Npn \__draw_backend_fill:
1656 { \__draw_backend_path:n { style="stroke:none" } }
1657 \cs_new_protected:Npn \__draw_backend_fillstroke:
1658 { \__draw_backend_path:n { } }
1659 \cs_new_protected:Npn \__draw_backend_clip:
1660 { \bool_gset_true:N \g__draw_draw_clip_bool }
1661 \bool_new:N \g__draw_draw_clip_bool
1662 \cs_new_protected:Npn \__draw_backend_discardpath:
1663 {
1664     \bool_if:NT \g__draw_draw_clip_bool
1665     {
1666         \int_gincr:N \g__kernel_clip_path_int
1667         \__draw_backend_literal:x
1668         {
1669             < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1670             { ?nl }
1671             <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1672             < /clipPath >
1673         }
1674         \__kernel_backend_scope:x
1675         {
1676             clip-path =
1677                 "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1678         }
1679     }
1680     \tl_gclear:N \g__draw_path_tl
1681     \bool_gset_false:N \g__draw_draw_clip_bool
1682 }

```

(End definition for __draw_backend_path:n and others.)

```

\__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_but:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:

```

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

```

1683 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2

```

```

1684 {
1685   \use:x
1686   {
1687     \__draw_backend_dash_aux:nn
1688     { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1689     { \dim_to_decimal:n {#2} }
1690   }
1691 }
1692 \cs_new:Npn \__draw_backend_dash:n #1
1693 { , \dim_to_decimal_in_bp:n {#1} }
1694 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1695 {
1696   \__kernel_backend_scope:x
1697   {
1698     stroke-dasharray =
1699     "
1700     \tl_if_empty:nTF {#1}
1701     { none }
1702     { \use_none:n #1 }
1703     " ~
1704     stroke-offset=" #2 "
1705   }
1706 }
1707 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1708 { \__kernel_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1709 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1710 { \__kernel_backend_scope:x { stroke-miterlimit=" #1 " } }
1711 \cs_new_protected:Npn \__draw_backend_cap_but:
1712 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1713 \cs_new_protected:Npn \__draw_backend_cap_round:
1714 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1715 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1716 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1717 \cs_new_protected:Npn \__draw_backend_join_miter:
1718 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1719 \cs_new_protected:Npn \__draw_backend_join_round:
1720 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1721 \cs_new_protected:Npn \__draw_backend_join_bevel:
1722 { \__kernel_backend_scope:n { stroke-linejoin="bevel" } }

```

(End definition for __draw_backend_dash_pattern:nn and others.)

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

```

1723 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1724 {
1725   \__kernel_backend_scope:n
1726   {
1727     transform =
1728     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1729   }
1730 }

```

(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.

```
1731 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
1732 {
1733   \_kernel_backend_scope_begin:
1734   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1735   \_kernel_backend_literal_svg:n
1736   {
1737     < g~
1738     stroke="none"~
1739     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1740     >
1741   }
1742   \box_set_wd:Nn #1 { Opt }
1743   \box_set_ht:Nn #1 { Opt }
1744   \box_set_dp:Nn #1 { Opt }
1745   \box_use:N #1
1746   \_kernel_backend_literal_svg:n { </g> }
1747   \_kernel_backend_scope_end:
1748 }

(End definition for \_draw_backend_box_use:Nnnnn.)

1749 </dvisvgm>
1750 </package>
```

5 l3backend-graphics Implementation

```
1751 <*package>
1752 <@@=graphics>
```

`_graphics_backend_loaded:n`

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

```
1753 \cs_new_protected:Npn \_graphics_backend_loaded:n #1
1754 {
1755   \cs_if_exist:NTF \hook_gput_code:nnn
1756   {
1757     \hook_gput_code:nnn
1758     { file / l3graphics.sty / after }
1759     { backend }
1760     {#1}
1761   }
1762   {#1}
1763 }
```

(End definition for `_graphics_backend_loaded:n`.)

5.1 dvips backend

```
1764 <*dvips>
```

`\l_graphics_search_ext_seq`

```
1765 \_graphics_backend_loaded:n
1766 { \seq_set_from_clist:Nn \l_graphics_search_ext_seq { .eps , .ps } }
```

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

`_graphics_backend_getbb_eps:n`
`_graphics_backend_getbb_ps:n`

Simply use the generic function.

```
1767 \_graphics_backend_loaded:n
1768 {
1769   \cs_new_eq:NN \_graphics_backend_getbb_eps:n \_graphics_read_bb:n
1770   \cs_new_eq:NN \_graphics_backend_getbb_ps:n \_graphics_read_bb:n
1771 }
```

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

`_graphics_backend_include_eps:n`
`_graphics_backend_include_ps:n`

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

```
1772 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1773 {
1774   \_kernel_backend_literal:x
1775   {
1776     PSfile = #1 \c_space_tl
1777     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1778     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1779     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1780     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1781   }
1782 }
1783 \cs_new_eq:NN \_graphics_backend_include_ps:n \_graphics_backend_include_eps:n
```

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

`_graphics_backend_get_pagecount:n`

```
1784 \_graphics_backend_loaded:n
1785 { \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n }
```

(End definition for `_graphics_backend_get_pagecount:n`.)

```
1786 </dvips>
```

5.2 LuaTeX and pdfTeX backends

```
1787 <*luatex | pdftex>
```

`\l_graphics_search_ext_seq`

```
1788 \_graphics_backend_loaded:n
1789 {
1790   \seq_set_from_clist:Nn
1791   \l_graphics_search_ext_seq
1792   { .pdf , .eps , .ps , .png , .jpg , .jpeg }
1793 }
```

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

`\l__graphics_graphics_attr_tl`

In PDF mode, additional attributes of an graphic (such as page number) are needed both to obtain the bounding box and when inserting the graphic: this occurs as the graphic dictionary approach means they are read as part of the bounding box operation. As such, it is easier to track additional attributes using a dedicated `tl` rather than build up the same data twice.

```
1794 \tl_new:N \l__graphics_graphics_attr_tl
```

(End definition for \l__graphics_graphics_attr_tl.)

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_jpeg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
\__graphics_backend_getbb_auxi:n
\__graphics_backend_getbb_auxii:n
\__graphics_backend_getbb_auxiii:n
\__graphics_backend_dequote:w
1795 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1796 {
1797   \int_zero:N \l__graphics_page_int
1798   \tl_clear:N \l__graphics_pagebox_tl
1799   \tl_set:Nx \l__graphics_graphics_attr_tl
1800   {
1801     \tl_if_empty:NF \l__graphics_decodearray_str
1802     { :D \l__graphics_decodearray_str }
1803     \bool_if:NT \l__graphics_interpolate_bool
1804     { :I }
1805   }
1806   \tl_clear:N \l__graphics_graphics_attr_tl
1807   \__graphics_backend_getbb_auxi:n {#1}
1808 }
1809 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1810 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1811 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1812 {
1813   \tl_clear:N \l__graphics_decodearray_str
1814   \bool_set_false:N \l__graphics_interpolate_bool
1815   \tl_set:Nx \l__graphics_graphics_attr_tl
1816   {
1817     : \l__graphics_pagebox_tl
1818     \int_compare:nNnT \l__graphics_page_int > 1
1819     { :P \int_use:N \l__graphics_page_int }
1820   }
1821   \__graphics_backend_getbb_auxi:n {#1}
1822 }
1823 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1824 {
1825   \__graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1826   { \__graphics_backend_getbb_auxii:n {#1} }
1827 }

```

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. Quotes need to be *removed* as LuaTeX does not like them here.

```

1828 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1829 {
1830   \exp_args:Ne \__graphics_backend_getbb_auxiii:n
1831   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1832   \int_const:cn { c__graphics_ #1 \l__graphics_graphics_attr_tl _int }
1833   { \tex_the:D \tex_pdflastximage:D }
1834   \__graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1835 }
1836 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1

```



```

1837 {
1838   \tex_immediate:D \tex_pdfximage:D
1839   \bool_lazy_or:nnT
1840     { \l__graphics_interpolate_bool }
1841     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1842     {
1843       attr ~
1844       {
1845         \tl_if_empty:NF \l__graphics_decodearray_str
1846         { /Decode~[ \l__graphics_decodearray_str ] }
1847         \bool_if:NT \l__graphics_interpolate_bool
1848         { /Interpolate~true }
1849       }
1850     }
1851     \int_compare:nNnT \l__graphics_page_int > 0
1852     { page ~ \int_use:N \l__graphics_page_int }
1853     \tl_if_empty:NF \l__graphics_pagebox_tl
1854     { \l__graphics_pagebox_tl }
1855     {#1}
1856     \hbox_set:Nn \l__graphics_internal_box
1857     { \tex_pdfrefximage:D \tex_pdflastximage:D }
1858     \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1859     \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1860   }
1861   \cs_new:Npn \__graphics_backend_dequote:w #1 " #2 " #3 \s__graphics_stop {#2}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

__graphics_backend_include_jpg:n
 __graphics_backend_include_jpeg:n
 __graphics_backend_include_pdf:n
 __graphics_backend_include_png:n

Images are already loaded for the measurement part of the code, so inclusion is straightforward, with only any attributes to worry about. The latter carry through from determination of the bounding box.

```

1862 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1863 {
1864   \tex_pdfrefximage:D
1865   \int_use:c { c__graphics_ #1 \l__graphics_graphics_attr_tl _int }
1866 }
1867 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1868 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1869 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

(End definition for __graphics_backend_include_jpg:n and others.)

__graphics_backend_getbb_eps:n
 __graphics_backend_getbb_ps:n
 __graphics_backend_getbb_eps:nm
 __graphics_backend_include_eps:n
 __graphics_backend_include_ps:n

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.

```

1870 \sys_if_shell:T
1871 {
1872   \str_new:N \l__graphics_backend_dir_str
1873   \str_new:N \l__graphics_backend_name_str
1874   \str_new:N \l__graphics_backend_ext_str
1875   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1876   {
1877     \file_parse_full_name:nNNN {#1}
1878     \l__graphics_backend_dir_str

```

```

1879         \l__graphics_backend_name_str
1880         \l__graphics_backend_ext_str
1881     \exp_args:Nx \__graphics_backend_getbb_eps:nn
1882     {
1883         \exp_args:Ne \__kernel_file_name_quote:n
1884         {
1885             \l__graphics_backend_name_str
1886             - \str_tail:N \l__graphics_backend_ext_str
1887             -converted-to.pdf
1888         }
1889     }
1890     {#1}
1891 }
1892 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_backend_getbb_eps:n
1893 \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1894 {
1895     \file_compare_timestamp:nNnT {#2} > {#1}
1896     {
1897         \sys_shell_now:n
1898         { repstopdf ~ #2 ~ #1 }
1899     }
1900     \tl_set:Nn \l__graphics_final_name_str {#1}
1901     \__graphics_backend_getbb_pdf:n {#1}
1902 }
1903 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1904 {
1905     \file_parse_full_name:nNNN {#1}
1906     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1907     \exp_args:Nx \__graphics_backend_include_pdf:n
1908     {
1909         \exp_args:Ne \__kernel_file_name_quote:n
1910         {
1911             \l__graphics_backend_name_str
1912             - \str_tail:N \l__graphics_backend_ext_str
1913             -converted-to.pdf
1914         }
1915     }
1916 }
1917 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1918 }

```

(End definition for __graphics_backend_getbb_eps:n and others.)

__graphics_backend_get_pagecount:n Simply load and store.

```

1919 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1920 {
1921     \tex_immediate:D \tex_pdfximage:D {#1}
1922     \int_const:cn { c__graphics_ #1 _pages_int }
1923     { \int_use:N \tex_pdflastximagepages:D }
1924 }

```

(End definition for __graphics_backend_get_pagecount:n.)

1925 </luatex | pdftex>

5.3 dvipdfmx backend

1926 $\langle *dvipdfmx | xetex \rangle$

$\backslash l_graphics_search_ext_seq$

```
1927 \__graphics_backend_loaded:n
1928 {
1929   \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1930   { .pdf , .eps , .ps , .png , .jpg ., jpeg , .bmp }
1931 }
```

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

$\backslash_graphics_backend_getbb_eps:n$

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

$\backslash_graphics_backend_getbb_ps:n$

```
1932 \__graphics_backend_loaded:n
```

$\backslash_graphics_backend_getbb_jpg:n$

```
1933 {
```

$\backslash_graphics_backend_getbb_jpeg:n$

```
1934   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
```

$\backslash_graphics_backend_getbb_pdf:n$

```
1935   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
```

$\backslash_graphics_backend_getbb_png:n$

```
1936 }
```

$\backslash_graphics_backend_getbb_bmp:n$

```
1937  $\langle *dvipdfmx \rangle$ 
```

```
1938 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
```

```
1939 {
```

```
1940   \int_zero:N \l__graphics_page_int
```

```
1941   \tl_clear:N \l__graphics_pagebox_tl
```

```
1942   \__graphics_extract_bb:n {#1}
```

```
1943 }
```

```
1944 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
```

```
1945 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
```

```
1946 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
```

```
1947 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
```

```
1948 {
```

```
1949   \tl_clear:N \l__graphics_decodearray_str
```

```
1950   \bool_set_false:N \l__graphics_interpolate_bool
```

```
1951   \__graphics_extract_bb:n {#1}
```

```
1952 }
```

```
1953  $\langle /dvipdfmx \rangle$ 
```

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

$\backslash g_graphics_track_int$

Used to track the object number associated with each graphic.

```
1954 \int_new:N \g__graphics_track_int
```

(End definition for $\backslash g_graphics_track_int$.)

$\backslash_graphics_backend_include_eps:n$

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.

$\backslash_graphics_backend_include_ps:n$

$\backslash_graphics_backend_include_jpg:n$

$\backslash_graphics_backend_include_jpeg:n$

$\backslash_graphics_backend_include_pdf:n$

```
1955 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
```

$\backslash_graphics_backend_include_png:n$

```
1956 {
```

$\backslash_graphics_backend_include_bmp:n$

```
1957   \__kernel_backend_literal:x
```

```
1958   {
```

$\backslash_graphics_backend_include_auxi:n$

```
1959     PSfile = #1 \c_space_tl
```

$\backslash_graphics_backend_include_auxii:nn$

```
1960     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
```

$\backslash_graphics_backend_include_auxiii:nn$

```
1961     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
```

```
1962     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
```

```

1963         ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1964     }
1965 }
1966 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1967 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1968 { \__graphics_backend_include_auxi:nn {#1} { image } }
1969 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1970 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
1971 \cs_new_eq:NN \__graphics_backend_include_bmp:n \__graphics_backend_include_jpg:n
1972 \<dvipdfmx>
1973 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1974 { \__graphics_backend_include_auxi:nn {#1} { epdf } }
1975 \</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.

```

1976 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1977 {
1978     \__graphics_backend_include_auxii:xnn
1979     {
1980         \tl_if_empty:NF \l__graphics_pagebox_tl
1981         { : \l__graphics_pagebox_tl }
1982         \int_compare:nNnT \l__graphics_page_int > 1
1983         { :P \int_use:N \l__graphics_page_int }
1984         \tl_if_empty:NF \l__graphics_decodearray_str
1985         { :D \l__graphics_decodearray_str }
1986         \bool_if:NT \l__graphics_interpolate_bool
1987         { :I }
1988     }
1989     {#1} {#2}
1990 }
1991 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1992 {
1993     \int_if_exist:cTF { c__graphics_ #2#1 _int }
1994     {
1995         \__kernel_backend_literal:x
1996         { pdf:usexobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
1997     }
1998     { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1999 }
2000 \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!

```

2001 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
2002 {
2003     \int_gincr:N \g__graphics_track_int
2004     \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
2005     \__kernel_backend_literal:x
2006     {
2007         pdf:#3~

```

```

2008 @graphic \int_use:c { c__graphics_ #1#2 _int } ~
2009 \int_compare:nNnT \l__graphics_page_int > 1
2010 { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2011 \tl_if_empty:NF \l__graphics_pagebox_tl
2012 {
2013   pagebox ~ \l__graphics_pagebox_tl \c_space_tl
2014   bbox ~
2015     \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2016     \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2017     \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2018     \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
2019 }
2020 (#1)
2021 \bool_lazy_or:nnT
2022 { \l__graphics_interpolate_bool }
2023 { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
2024 {
2025   <<
2026     \tl_if_empty:NF \l__graphics_decodearray_str
2027     { /Decode~[ \l__graphics_decodearray_str ] }
2028     \bool_if:NT \l__graphics_interpolate_bool
2029     { /Interpolate~true> }
2030   >>
2031 }
2032 }
2033 }

```

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

`__graphics_backend_get_pagecount:n`

```

2034 <*\dviptfm>
2035 \__graphics_backend_loaded:n
2036 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }
2037 </dviptfm>

```

(End definition for `__graphics_backend_get_pagecount:n`.)

```

2038 </dviptfm | xetex>

```

5.4 X_YTeX backend

```

2039 <*\xetex>

```

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.

```

2040 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2041 {
2042   \int_zero:N \l__graphics_page_int
2043   \tl_clear:N \l__graphics_pagebox_tl
2044   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2045 }
2046 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2047 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n

```

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_jpeg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
\__graphics_backend_getbb_bmp:n
\__graphics_backend_getbb_auxi:nN
\__graphics_backend_getbb_auxii:nnN
\__graphics_backend_getbb_auxiii:VnN
\__graphics_backend_getbb_auxiiii:nNnN
\__graphics_backend_getbb_auxiv:nnNnN
\__graphics_backend_getbb_auxiv:VnNnN
\__graphics_backend_getbb_auxv:nNnN
\__graphics_backend_getbb_auxv:nNnN
\__graphics_backend_getbb_pagebox:n

```

```

2048 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2049 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2050 {
2051   \tl_clear:N \l__graphics_decodearray_str
2052   \bool_set_false:N \l__graphics_interpolate_bool
2053   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdf:file:D
2054 }
2055 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2056 {
2057   \int_compare:nNnTF \l__graphics_page_int > 1
2058   { \__graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2059   { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2060 }
2061 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2062 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2063 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2064 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2065 {
2066   \tl_if_empty:NTF \l__graphics_pagebox_tl
2067   { \__graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2068   { \__graphics_backend_getbb_auxv:nNnn }
2069   {#1} #2 {#3} {#4}
2070 }
2071 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2072 {
2073   \use:x
2074   {
2075     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2076     {
2077       #5
2078       \tl_if_blank:nF {#1}
2079       { \c_space_tl \__graphics_backend_getbb_pagebox:w #1 }
2080     }
2081   }
2082 }
2083 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2084 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2085 {
2086   \__graphics_bb_restore:nF {#1#3}
2087   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2088 }
2089 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2090 {
2091   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
2092   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2093   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2094   \__graphics_bb_save:n {#1#3}
2095 }
2096 \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

__graphics_backend_include_pdf:n For PDF graphics, properly supporting the pagebox concept in X_YTeX is best done using the \tex_XeTeXpdf:file: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`.

```

2097 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2098 {
2099   \tex_XeTeXpdffile:D #1 ~
2100   \int_compare:nNnT \l__graphics_page_int > 0
2101     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2102   \exp_after:wN \__graphics_backend_getbb_pagebox:w \l__graphics_pagebox_tl
2103 }

```

(End definition for `__graphics_backend_include_pdf:n`.)

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

```

2104 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
2105 {
2106   \int_const:cn { c__graphics_#1_pages_int }
2107   {
2108     \int_max:nn
2109     { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2110     { 1 }
2111   }
2112 }

```

(End definition for `__graphics_backend_get_pagecount:n`.)

2113 `\</xetex>`

5.5 dvisvgm backend

2114 `*dvisvgm`

`\l_graphics_search_ext_seq`

```

2115 \__graphics_backend_loaded:n
2116 {
2117   \seq_set_from_clist:Nn
2118   \l_graphics_search_ext_seq
2119   { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }
2120 }

```

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

`__graphics_backend_getbb_svg:n`
`__graphics_backend_getbb_svg_auxi:nNn`
`__graphics_backend_getbb_svg_auxii:w`
`__graphics_backend_getbb_svg_auxiii:Nw`
`__graphics_backend_getbb_svg_auxiv:Nw`
`__graphics_backend_getbb_svg_auxv:Nw`
`__graphics_backend_getbb_svg_auxvi:Nn`
`__graphics_backend_getbb_svg_auxvii:w`

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

```

2121 \cs_new_protected:Npn \__graphics_backend_getbb_svg:n #1
2122 {
2123   \__graphics_bb_restore:nF {#1}
2124   {
2125     \ior_open:Nn \l__graphics_internal_ior {#1}
2126     \ior_if_eof:NTF \l__graphics_internal_ior
2127     { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2128     {
2129       \dim_zero:N \l__graphics_llx_dim
2130       \dim_zero:N \l__graphics_lly_dim

```

```

2131 \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2132 \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2133 \ior_str_map_inline:Nn \l__graphics_internal_ior
2134 {
2135   \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2136   {
2137     \__graphics_backend_getbb_svg_auxi:nNn
2138     { width } \l__graphics_urx_dim {##1}
2139   }
2140   \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2141   {
2142     \__graphics_backend_getbb_svg_auxi:nNn
2143     { height } \l__graphics_ury_dim {##1}
2144   }
2145   \bool_lazy_and:nnF
2146   { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2147   { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2148   { \ior_map_break: }
2149 }
2150 \__graphics_bb_save:n {#1}
2151 }
2152 \ior_close:N \l__graphics_internal_ior
2153 }
2154 }
2155 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2156 {
2157   \use:x
2158   {
2159     \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxii:w
2160     #####1 \tl_to_str:n {#1} = #####2 \tl_to_str:n {#1} = #####3
2161     \s__graphics_stop
2162   }
2163   {
2164     \tl_if_blank:nF {##2}
2165     {
2166       \peek_remove_spaces:n
2167       {
2168         \peek_meaning:NTF ' % '
2169         { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2170         {
2171           \peek_meaning:NTF " % "
2172           { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2173           { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2174         }
2175       }
2176       ##2 \s__graphics_stop
2177     }
2178   }
2179   \use:x
2180   {
2181     \__graphics_backend_getbb_svg_auxii:w #3
2182     \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2183     \s__graphics_stop
2184   }

```



```

2185 }
2186 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2187 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop
2188 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2189 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2190 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2191 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2192 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2193 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2194 {
2195   \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2196   \l__graphics_internal_dim #2 bp \scan_stop:
2197   \dim_set_eq:NN #1 \l__graphics_internal_dim
2198 }
2199 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvii:w #1 \scan_stop: { }

```

(End definition for __graphics_backend_getbb_svg:n and others.)

__graphics_backend_getbb_eps:n
 __graphics_backend_getbb_ps:n

Simply use the generic function.

```

2200 \__graphics_backend_loaded:n
2201 {
2202   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2203   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
2204 }

```

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

__graphics_backend_getbb_png:n
 __graphics_backend_getbb_jpg:n
 __graphics_backend_getbb_jpeg:n

These can be included by extracting the bounding box data.

```

2205 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2206 {
2207   \int_zero:N \l__graphics_page_int
2208   \tl_clear:N \l__graphics_pagebox_tl
2209   \__graphics_extract_bb:n {#1}
2210 }
2211 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2212 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n

```

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

__graphics_backend_getbb_pdf:n

Same as for dvipdfmx: use the generic function

```

2213 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2214 {
2215   \tl_clear:N \l__graphics_decodearray_str
2216   \bool_set_false:N \l__graphics_interpolate_bool
2217   \__graphics_extract_bb:n {#1}
2218 }

```

(End definition for __graphics_backend_getbb_pdf:n.)

__graphics_backend_include_eps:n
 __graphics_backend_include_ps:n
 __graphics_backend_include_pdf:n
 __graphics_backend_include:nn

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

```

2219 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2220 { \__graphics_backend_include:nn { PSfile } {#1} }
2221 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n

```

```

2222 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2223 { \__graphics_backend_include:nn { pdf file } {#1} }
2224 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
2225 {
2226   \__kernel_backend_literal:x
2227   {
2228     #1 = #2 \c_space_tl
2229     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2230     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2231     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2232     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2233   }
2234 }

```

(End definition for __graphics_backend_include_eps:n and others.)

```

\__graphics_backend_include_svg:n
\__graphics_backend_include_png:n
\__graphics_backend_include_jpg:n
\__graphics_backend_include_jpeg:n
\__graphics_backend_include_dequote:w

```

The backend here has built-in support for basic graphic inclusion (see `dvisvgm.def` for a more complex approach, needed if clipping, *etc.*, is covered at the graphic backend level). We have to deal with the fact that the image reference point is at the *top*, so there is a need for a veritcal shift to put it in the right place. The other issue is that `#1` must be quote-corrected. The `dvisvgm:img` operation quotes the file name, but if it is already quoted (contains spaces) then we have an issue: we simply strip off any quotes as a result.

```

2235 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2236 {
2237   \box_move_up:nn { \l__graphics_ury_dim }
2238   {
2239     \hbox:n
2240     {
2241       \__kernel_backend_literal:x
2242       {
2243         dvisvgm:img~
2244         \dim_to_decimal:n { \l__graphics_urx_dim } ~
2245         \dim_to_decimal:n { \l__graphics_ury_dim } ~
2246         \__graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2247       }
2248     }
2249   }
2250 }
2251 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_svg:n
2252 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_svg:n
2253 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_svg:n
2254 \cs_new:Npn \__graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2255 {#2}

```

(End definition for __graphics_backend_include_svg:n and others.)

```

\__graphics_backend_get_pagecount:n

```

```

2256 \__graphics_backend_loaded:n
2257 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }

```

(End definition for __graphics_backend_get_pagecount:n.)

```

2258 </dvisvgm>
2259 </package>

```

6 l3backend-pdf Implementation

```
2260 ⟨*package⟩
2261 ⟨@@=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
2262 \box_new:N \l__pdf_internal_box
(End definition for \l__pdf_internal_box.)
```

6.2 dvips backend

```
2263 ⟨*dvips⟩
\__pdf_backend_pdfmark:n Used often enough it should be a separate function.
\__pdf_backend_pdfmark:x 2264 \cs_new_protected:Npn \__pdf_backend_pdfmark:n #1
2265 { \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2266 \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 2267 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2268 { \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2269 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2270 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }
(End definition for \__pdf_backend_catalog_gput:nn and \__pdf_backend_info_gput:nn.)
```

6.2.2 Objects

```
\g__pdf_backend_object_int For tracking objects to allow finalisation.
\g__pdf_backend_object_prop 2271 \int_new:N \g__pdf_backend_object_int
2272 \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 Tracking objects is similar to dvipdfmx.
\__pdf_backend_object_ref:n 2273 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2274 {
2275 \int_gincr:N \g__pdf_backend_object_int
2276 \int_const:cn
2277 { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2278 { \g__pdf_backend_object_int }
```

```

2279     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2280   }
2281   \cs_new:Npn \__pdf_backend_object_ref:n #1
2282     { { 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.)

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

```

\__pdf_backend_object_write:nn
\__pdf_backend_object_write:nx
\__pdf_backend_object_write_array:nn
\__pdf_backend_object_write_dict:nn
\__pdf_backend_object_write_fstream:nn
\__pdf_backend_object_write_stream:nn
\__pdf_backend_object_write_stream:nnn

2283 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2284 {
2285   \__pdf_backend_pdfmark:x
2286   {
2287     /objdef ~ \__pdf_backend_object_ref:n {#1}
2288     /type
2289     \str_case_e:nn
2290       { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2291       {
2292         { array } { /array }
2293         { dict } { /dict }
2294         { fstream } { /stream }
2295         { stream } { /stream }
2296       }
2297     /OBJ
2298   }
2299   \use:c
2300     { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2301     { \__pdf_backend_object_ref:n {#1} } {#2}
2302   }
2303   \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2304   \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2305   {
2306     \__pdf_backend_pdfmark:x
2307     { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2308   }
2309   \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2310   {
2311     \__pdf_backend_pdfmark:x
2312     { #1 << \exp_not:n {#2} >> /PUT }
2313   }
2314   \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2315   {
2316     \exp_args:Nx
2317     \__pdf_backend_object_write_fstream:nnn {#1} #2
2318   }
2319   \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2320   {
2321     \__kernel_backend_postscript:n
2322     {
2323       SDict ~ begin ~
2324       mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2325       mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2326       end
2327     }
2328   }

```

```

2329 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2330 {
2331   \exp_args:Nx
2332     \__pdf_backend_object_write_stream:nnn {#1} #2
2333 }
2334 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2335 {
2336   \__kernel_backend_postscript:n
2337   {
2338     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2339     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2340   }
2341 }

```

(End definition for __pdf_backend_object_write:nn and others.)

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

```

\__pdf_backend_object_now:nx
2342 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2343 {
2344   \int_gincr:N \g__pdf_backend_object_int
2345   \__pdf_backend_pdfmark:x
2346   {
2347     /_objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2348     /type
2349     \str_case:nn
2350       {#1}
2351       {
2352         { array } { /array }
2353         { dict } { /dict }
2354         { fstream } { /stream }
2355         { stream } { /stream }
2356       }
2357     /OBJ
2358   }
2359   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2360   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2361 }
2362 \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.

```

2363 \cs_new:Npn \__pdf_backend_object_last:
2364 { { 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.

```

2365 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2366 { { 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.
2367 \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.
2368 \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.
2369 \int_new:N \g__pdf_backend_annotation_int
(End definition for \g__pdf_backend_annotation_int.)

\__pdf_backend_annotation:nnnn Annotations are objects, but we track them separately. Notably, they are not in the
object data lists. Here, to get the co-ordinates of the annotation, we need to have the
data collected at the PostScript level. That requires a bit of box trickery (effectively a
LATEX 2ε picture of zero size). Once the data is collected, use it to set up the annotation
border.
2370 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2371 {
2372   \exp_args:Nf \__pdf_backend_annotation_aux:nnnn
2373   { \dim_eval:n {#1} } {#2} {#3} {#4}
2374 }
2375 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2376 {
2377   \box_move_down:nn {#3}
2378   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2379   \box_move_up:nn {#2}
2380   {
2381     \hbox:n
2382     {
2383       \__kernel_kern:n {#1}
2384       \__kernel_backend_postscript:n { pdf.save.ur }
2385       \__kernel_kern:n { -#1 }
2386     }
2387   }
2388   \int_gincr:N \g__pdf_backend_object_int
2389   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2390   \__pdf_backend_pdfmark:x
2391   {
2392     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2393     pdf.rect
2394     #4 ~
2395     /ANN
2396   }
2397 }
(End definition for \__pdf_backend_annotation:nnnn.)

```

<code>_pdf_backend_annotation_last:</code>	Provide the last annotation we created: could get tricky of course if other packages are loaded. <pre> 2398 \cs_new:Npn _pdf_backend_annotation_last: 2399 { { pdf.obj \int_use:N \g__pdf_backend_annotation_int } } </pre> (End definition for <code>_pdf_backend_annotation_last:</code>)
<code>\g__pdf_backend_link_int</code>	To track annotations which are links. <pre> 2400 \int_new:N \g__pdf_backend_link_int </pre> (End definition for <code>\g__pdf_backend_link_int.</code>)
<code>\g__pdf_backend_link_dict_tl</code>	To pass information to the end-of-link function. <pre> 2401 \tl_new:N \g__pdf_backend_link_dict_tl </pre> (End definition for <code>\g__pdf_backend_link_dict_tl.</code>)
<code>\g__pdf_backend_link_sf_int</code>	Needed to save/restore space factor, which is needed to deal with the face we need a box. <pre> 2402 \int_new:N \g__pdf_backend_link_sf_int </pre> (End definition for <code>\g__pdf_backend_link_sf_int.</code>)
<code>\g__pdf_backend_link_math_bool</code>	Needed to save/restore math mode. <pre> 2403 \bool_new:N \g__pdf_backend_link_math_bool </pre> (End definition for <code>\g__pdf_backend_link_math_bool.</code>)
<code>\g__pdf_backend_link_bool</code>	Track link formation: we cannot nest at all. <pre> 2404 \bool_new:N \g__pdf_backend_link_bool </pre> (End definition for <code>\g__pdf_backend_link_bool.</code>)
<code>\l__pdf_breaklink_pdfmark_tl</code>	Swappable content for link breaking. <pre> 2405 \tl_new:N \l__pdf_breaklink_pdfmark_tl 2406 \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdfmark } </pre> (End definition for <code>\l__pdf_breaklink_pdfmark_tl.</code>)
<code>_pdf_breaklink_postscript:n</code>	To allow dropping material unless link breaking is active. <pre> 2407 \cs_new_protected:Npn _pdf_breaklink_postscript:n #1 { } </pre> (End definition for <code>_pdf_breaklink_postscript:n.</code>)
<code>__pdf_breaklink_usebox:N</code>	Swappable box unpacking or use. <pre> 2408 \cs_new_eq:NN __pdf_breaklink_usebox:N \box_use:N </pre> (End definition for <code>__pdf_breaklink_usebox:N.</code>)

```

\__pdf_backend_link_begin_goto:nnw
\__pdf_backend_link_begin_user:nnw
\__pdf_backend_link:nw
\__pdf_backend_link_aux:nw
\__pdf_backend_link_end:
\__pdf_backend_link_end_aux:
\__pdf_backend_link_minima:
\__pdf_backend_link_outerbox:n
\__pdf_backend_link_sf_save:
\__pdf_backend_link_sf_restore:
pdf.linkdp.pad
pdf.linkht.pad
pdf.llx
pdf.lly
pdf.ury
pdf.link.dict
pdf.outerbox
pdf.baselineskip

```

Links are created like annotations but with dedicated code to allow for adjusting the size of the rectangle. In contrast to `hyperref`, we grab the link content as a box which can then unbox: this allows the same interface as for `pdfTeX`.

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

Taking the idea of `evenboxes` from `hypdvips`, we implement a minimum box height and depth for link placement. This means that “underlining” with a hyperlink will generally give an even appearance. However, to ensure that the full content is always above the link border, we do not allow this to be negative (contrast `hypdvips` approach). The result should be similar to `pdfTeX` in the vast majority of foreseeable cases.

The object number for a link is saved separately from the rest of the dictionary as this allows us to insert it just once, at either an unbroken link or only in the first line of a broken one. That makes the code clearer but also avoids a low-level PostScript error with the code as taken from `hypdvips`.

Getting the outer dimensions of the text area may be better using a two-pass approach and `\tex_savepos:D`. That plus generic mode are still to re-examine.

```

2409 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2410 {
2411   \__pdf_backend_link_begin:nw
2412   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2413 }
2414 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
2415 { \__pdf_backend_link_begin:nw {#1#2} }
2416 \cs_new_protected:Npn \__pdf_backend_link_begin:nw #1
2417 {
2418   \bool_if:NF \g__pdf_backend_link_bool
2419   { \__pdf_backend_link_begin_aux:nw {#1} }
2420 }

```

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.

```

2421 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2422 {
2423   \bool_gset_true:N \g__pdf_backend_link_bool
2424   \__kernel_backend_postscript:n
2425   { /pdf.link.dict ( #1 ) def }
2426   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2427   \__pdf_backend_link_sf_save:
2428   \mode_if_math:TF
2429   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2430   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2431   \hbox_set:Nw \l__pdf_backend_content_box
2432   \__pdf_backend_link_sf_restore:
2433   \bool_if:NT \g__pdf_backend_link_math_bool
2434   { \c_math_toggle_token }
2435 }
2436 \cs_new_protected:Npn \__pdf_backend_link_end:
2437 {
2438   \bool_if:NT \g__pdf_backend_link_bool
2439   { \__pdf_backend_link_end_aux: }
2440 }
2441 \cs_new_protected:Npn \__pdf_backend_link_end_aux:

```



```

2442 {
2443     \bool_if:NT \g__pdf_backend_link_math_bool
2444     { \c_math_toggle_token }
2445     \__pdf_backend_link_sf_save:
2446     \hbox_set_end:
2447     \__pdf_backend_link_minima:
2448     \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2449     \exp_args:Nx \__pdf_backend_link_outerbox:n
2450     {
2451         \int_if_odd:nTF { \value { page } }
2452         { \oddsidemargin }
2453         { \evensidemargin }
2454     }
2455     \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2456     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2457     \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2458     \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2459     \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2460     \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2461     {
2462         \hbox:n
2463         { \__kernel_backend_postscript:n { pdf.save.linkur } }
2464     }
2465     \int_gincr:N \g__pdf_backend_object_int
2466     \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2467     \__kernel_backend_postscript:x
2468     {
2469         mark
2470         /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2471         \g__pdf_backend_link_dict_tl \c_space_tl
2472         pdf.rect
2473         /ANN ~ \l__pdf_breaklink_pdfmark_tl
2474     }
2475     \__pdf_backend_link_sf_restore:
2476     \bool_gset_false:N \g__pdf_backend_link_bool
2477 }
2478 \cs_new_protected:Npn \__pdf_backend_link_minima:
2479 {
2480     \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2481     \__kernel_backend_postscript:x
2482     {
2483         /pdf.linkdp.pad ~
2484         \dim_to_decimal:n
2485         {
2486             \dim_max:nn
2487             {
2488                 \box_dp:N \l__pdf_backend_model_box
2489                 - \box_dp:N \l__pdf_backend_content_box
2490             }
2491             { Opt }
2492         } ~
2493         pdf.pt.dvi ~ def
2494         /pdf.linkht.pad ~
2495         \dim_to_decimal:n

```

```

2496         {
2497             \dim_max:nn
2498             {
2499                 \box_ht:N \l__pdf_backend_model_box
2500                 - \box_ht:N \l__pdf_backend_content_box
2501             }
2502             { Opt }
2503         } ~
2504         pdf.pt.dvi ~ def
2505     }
2506 }
2507 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2508 {
2509     \__kernel_backend_postscript:x
2510     {
2511         /pdf.outerbox
2512         [
2513             \dim_to_decimal:n {#1} ~
2514             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2515             \dim_to_decimal:n { #1 + \textwidth } ~
2516             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2517         ]
2518         [ exch { pdf.pt.dvi } forall ] def
2519         /pdf.baselineskip ~
2520         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2521         { pdf.pt.dvi ~ def }
2522         { pop ~ pop }
2523         ifelse
2524     }
2525 }
2526 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2527 {
2528     \int_gset:Nn \g__pdf_backend_link_sf_int
2529     {
2530         \mode_if_horizontal:TF
2531         { \tex_spacefactor:D }
2532         { 0 }
2533     }
2534 }
2535 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2536 {
2537     \mode_if_horizontal:T
2538     {
2539         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2540         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2541     }
2542 }

```

(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.

```

2543 \use_none:n
2544 {
2545   \cs_if_exist:NT \@makecol@hook
2546   {
2547     \tl_put_right:Nn \@makecol@hook
2548     {
2549       \box_if_empty:NF \@cclv
2550       {
2551         \vbox_set:Nn \@cclv
2552         {
2553           \__kernel_backend_postscript:n
2554           {
2555             pdf.globaldict /pdf.brokenlink.rect ~ known
2556             { pdf.bordertracking.continue }
2557             if
2558             }
2559             \vbox_unpack_drop:N \@cclv
2560             \__kernel_backend_postscript:n
2561             { pdf.bordertracking.endpage }
2562           }
2563         }
2564       }
2565       \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2566       \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2567       \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2568     }
2569   }

```

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

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

```

2570 \cs_new:Npn \__pdf_backend_link_last:
2571 { { 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.

```

2572 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2573 {
2574   \__kernel_backend_postscript:x
2575   {
2576     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2577   }
2578 }

```

(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.

```

2579 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2580 {
2581   \__kernel_backend_postscript:n { pdf.dest.anchor }

```

```

2582 \__pdf_backend_pdfmark:x
2583 {
2584   /View
2585   [
2586     \str_case:nnF {#2}
2587     {
2588       { xyz } { /XYZ ~ pdf.dest.point ~ null }
2589       { fit } { /Fit }
2590       { fitb } { /FitB }
2591       { fitbh } { /FitBH ~ pdf.dest.y }
2592       { fitbv } { /FitBV ~ pdf.dest.x }
2593       { fith } { /FitH ~ pdf.dest.y }
2594       { fitv } { /FitV ~ pdf.dest.x }
2595       { fitr } { /Fit }
2596     }
2597     {
2598       /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2599     }
2600   ]
2601   /Dest ( \exp_not:n {#1} ) cvn
2602   /DEST
2603 }
2604 }
2605 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2606 {
2607   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2608   { \dim_eval:n {#2} } {#1} {#3} {#4}
2609 }
2610 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2611 {
2612   \vbox_to_zero:n
2613   {
2614     \__kernel_kern:n {#4}
2615     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2616     \tex_vss:D
2617   }
2618   \__kernel_kern:n {#1}
2619   \vbox_to_zero:n
2620   {
2621     \__kernel_kern:n { -#3 }
2622     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2623     \tex_vss:D
2624   }
2625   \__kernel_kern:n { -#1 }
2626   \__pdf_backend_pdfmark:n
2627   {
2628     /View
2629     [
2630       /FitR ~
2631       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2632       pdf.urx ~ pdf.ury ~ pdf.dest2device
2633     ]
2634     /Dest ( #2 ) cvn
2635     /DEST

```

```

2636     }
2637 }

```

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

6.2.4 Structure

```

\_pdf_backend_compresslevel:n
\_pdf_backend_compress_objects:n

```

Doable for the usual ps2pdf method.

```

2638 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2639 {
2640   \int_compare:nNnT {#1} = 0
2641   {
2642     \__kernel_backend_literal_postscript:n
2643     {
2644       /setdistillerparams ~ where
2645       { pop << /CompressPages ~ false >> setdistillerparams }
2646       if
2647     }
2648   }
2649 }
2650 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2651 {
2652   \bool_if:nF {#1}
2653   {
2654     \__kernel_backend_literal_postscript:n
2655     {
2656       /setdistillerparams ~ where
2657       { pop << /CompressStreams ~ false >> setdistillerparams }
2658       if
2659     }
2660   }
2661 }

```

(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

```

```

2662 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2663 {
2664   \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2665 }
2666 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2667 {
2668   \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2669 }

```

(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!

```

2670 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2671 \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

Simple wrappers.

```

\__pdf_backend_bdc:nn
\__pdf_backend_emc:
2672 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2673 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2674 \cs_new_protected:Npn \__pdf_backend_emc:
2675 { \__pdf_backend_pdfmark:n { /EMC } }

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

2676 </dvips>

```

6.3 LuaTeX and pdfTeX backend

2677 <*luatex | pdftex>

6.3.1 Annotations

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

```

2678 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2679 {
2680 <*luatex>
2681 \tex_pdfextension:D annot ~
2682 </luatex>
2683 <*pdftex>
2684 \tex_pdfannot:D
2685 </pdftex>
2686 width ~ \dim_eval:n {#1} ~
2687 height ~ \dim_eval:n {#2} ~
2688 depth ~ \dim_eval:n {#3} ~
2689 {#4}
2690 }

```

(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.

```

2691 \cs_new:Npx \__pdf_backend_annotation_last:
2692 {
2693 \exp_not:N \int_value:w
2694 <*luatex>
2695 \exp_not:N \tex_pdffeedback:D lastannot ~
2696 </luatex>
2697 <*pdftex>
2698 \exp_not:N \tex_pdflastannot:D
2699 </pdftex>
2700 \c_space_tl 0 ~ R
2701 }

```

(End definition for __pdf_backend_annotation_last:.)

__pdf_backend_link_begin_goto:nnw Links are all created using the same internals.

```

\__pdf_backend_link_begin_user:nnw
\__pdf_backend_link_begin:nnnw
2702 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2703 { \__pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
\__pdf_backend_link_end:
2704 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2

```

```

2705 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2706 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2707 {
2708   \*luatex
2709   \tex_pdfextension:D startlink ~
2710   \*pdftex
2711   \tex_pdfstartlink:D
2712   \*pdftex
2713   attr {#1}
2714   #2 {#3}
2715 }
2716 \cs_new_protected:Npn \_pdf_backend_link_end:
2717 {
2718   \*luatex
2719   \tex_pdfextension:D endlink \scan_stop:
2720   \*pdftex
2721   \tex_pdfendlink:D
2722   \*pdftex
2723 }
2724
2725

```

(End definition for _pdf_backend_link_begin_goto:nnw and others.)

_pdf_backend_link_last: Formatted for direct use.

```

2726 \cs_new:Npx \_pdf_backend_link_last:
2727 {
2728   \exp_not:N \int_value:w
2729   \*luatex
2730   \exp_not:N \tex_pdffeedback:D lastlink ~
2731   \*pdftex
2732   \exp_not:N \tex_pdflastlink:D
2733   \*pdftex
2734   \c_space_tl 0 ~ R
2735 }
2736

```

(End definition for _pdf_backend_link_last:.)

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

```

2737 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2738 {
2739   \*luatex
2740   \tex_pdfvariable:D linkmargin
2741   \*pdftex
2742   \tex_pdflinkmargin:D
2743   \*pdftex
2744   \dim_eval:n {#1} \scan_stop:
2745 }
2746

```

(End definition for _pdf_backend_link_margin:n.)

`_pdf_backend_destination:nn`
`_pdf_backend_destination:nnnn`

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

```

2747 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2748 {
2749   \*luatex
2750     \tex_pdfextension:D dest ~
2751   \*pdfextension
2752     \tex_pdfdest:D
2753   \*pdfextension
2754     name {#1}
2755     \str_case:nnF {#2}
2756     {
2757       { xyz } { xyz }
2758       { fit } { fit }
2759       { fitb } { fitb }
2760       { fitbh } { fitbh }
2761       { fitbv } { fitbv }
2762       { fith } { fith }
2763       { fitv } { fitv }
2764       { fitr } { fitr }
2765     }
2766     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2767     \scan_stop:
2768   }
2769 }
2770 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2771 {
2772   \*luatex
2773     \tex_pdfextension:D dest ~
2774   \*pdfextension
2775     \tex_pdfdest:D
2776     name {#1}
2777     fitr ~
2778     width \dim_eval:n {#2} ~
2779     height \dim_eval:n {#3} ~
2780     depth \dim_eval:n {#4} \scan_stop:
2781   }
2782 }

```

(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`

```

2784 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2785 {
2786   \*luatex
2787     \tex_pdfextension:D catalog
2788   \*pdfextension
2789     \tex_pdfcatalog:D
2790   \*pdfextension
2791   }

```



```

2792     { / #1 ~ #2 }
2793   }
2794   \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2795   {
2796     \*luatex
2797       \tex_pdfextension:D info
2798     \*pdftex
2799     \tex_pdfinfo:D
2800     \*pdftex
2801     { / #1 ~ #2 }
2802   }
2803 }

```

(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.

```

2804 \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
2805 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2806 {
2807   \*luatex
2808     \tex_pdfextension:D obj ~
2809   \*pdftex
2810     \tex_pdfobj:D
2811     \reserveobjnum ~
2812     \int_const:cn
2813     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2814   \*luatex
2815     { \tex_pdffeedback:D lastobj }
2816   \*pdftex
2817     { \tex_pdflastobj:D }
2818   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2819 }
2820 \cs_new:Npn \__pdf_backend_object_ref:n #1
2821 { \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }
2822 }
2823 }
2824 }
2825 }

```

(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
2826 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2827 {
2828   \*luatex
2829     \tex_immediate:D \tex_pdfextension:D obj ~
2830   \*pdftex
2831     \tex_immediate:D \tex_pdfobj:D
2832 }

```

```

2833 </pdfTex>
2834     useobjnum ~
2835     \int_use:c
2836     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2837     \str_case_e:nn
2838     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2839     {
2840       { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2841       { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2842       { fstream }
2843       {
2844         stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2845         file ~ { \__pdf_exp_not_ii:nn #2 }
2846       }
2847       { stream }
2848       {
2849         stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2850         { \__pdf_exp_not_ii:nn #2 }
2851       }
2852     }
2853   }
2854   \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2855   \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2856   \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
 __pdf_backend_object_now:nx

Much like writing, but direct creation.

```

2857   \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2858   {
2859     <*luatex>
2860     \tex_immediate:D \tex_pdfextension:D obj ~
2861     </luatex>
2862     <*pdfTex>
2863     \tex_immediate:D \tex_pdfobj:D
2864     </pdfTex>
2865     \str_case:nn
2866     {#1}
2867     {
2868       { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2869       { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2870       { fstream }
2871       {
2872         stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2873         file ~ { \__pdf_exp_not_ii:nn #2 }
2874       }
2875       { stream }
2876       {
2877         stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2878         { \__pdf_exp_not_ii:nn #2 }
2879       }
2880     }
2881   }
2882   \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.

```

2883 \cs_new:Npx \_pdf_backend_object_last:
2884 {
2885   \exp_not:N \int_value:w
2886   <*luatex>
2887   \exp_not:N \tex_pdffeedback:D lastobj ~
2888   </luatex>
2889   <*pdftex>
2890   \exp_not:N \tex_pdflastobj:D
2891   </pdftex>
2892   \c_space_tl 0 ~ R
2893 }

```

(End definition for `_pdf_backend_object_last:.`)

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

```

2894 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2895 {
2896   \exp_not:N \int_value:w
2897   <*luatex>
2898   \exp_not:N \tex_pdffeedback:D pageref
2899   </luatex>
2900   <*pdftex>
2901   \exp_not:N \tex_pdfpageref:D
2902   </pdftex>
2903   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2904 }

```

(End definition for `_pdf_backend_pageobject_ref:n`.)

6.3.4 Structure

`_pdf_backend_compresslevel:n` Simply pass data to the engine.

```

\_pdf_backend_compress_objects:n
\_pdf_backend_objcompresslevel:n
2905 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2906 {
2907   \tex_global:D
2908   <*luatex>
2909   \tex_pdfvariable:D compresslevel
2910   </luatex>
2911   <*pdftex>
2912   \tex_pdfcompresslevel:D
2913   </pdftex>
2914   \int_value:w \int_eval:n {#1} \scan_stop:
2915 }
2916 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2917 {
2918   \bool_if:nTF {#1}
2919   { \_pdf_backend_objcompresslevel:n { 2 } }
2920   { \_pdf_backend_objcompresslevel:n { 0 } }
2921 }
2922 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2923 {

```

```

2924 \tex_global:D
2925 <*luatex>
2926 \tex_pdfvariable:D objcompresslevel
2927 </luatex>
2928 <*pdftex>
2929 \tex_pdfobjcompresslevel:D
2930 </pdftex>
2931 #1 \scan_stop:
2932 }

```

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

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

_pdf_backend_version_minor_gset:n

```

2933 \cs_new_protected:Npx \_pdf_backend_version_major_gset:n #1
2934 {
2935 <*luatex>
2936 \int_compare:nNnT \tex luatexversion:D > { 106 }
2937 {
2938 \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2939 \exp_not:N \int_eval:n {#1} \scan_stop:
2940 }
2941 </luatex>
2942 <*pdftex>
2943 \cs_if_exist:NT \tex_pdfmajorversion:D
2944 {
2945 \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2946 \exp_not:N \int_eval:n {#1} \scan_stop:
2947 }
2948 </pdftex>
2949 }
2950 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2951 {
2952 \tex_global:D
2953 <*luatex>
2954 \tex_pdfvariable:D minorversion
2955 </luatex>
2956 <*pdftex>
2957 \tex_pdfminorversion:D
2958 </pdftex>
2959 \int_eval:n {#1} \scan_stop:
2960 }

```

(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:

```

2961 \cs_new:Npx \_pdf_backend_version_major:
2962 {
2963 <*luatex>
2964 \int_compare:nNnTF \tex luatexversion:D > { 106 }
2965 { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2966 { 1 }
2967 </luatex>
2968 <*pdftex>
2969 \cs_if_exist:NTF \tex_pdfmajorversion:D

```

```

2970     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2971     { 1 }
2972   \</pdftex>
2973 }
2974 \cs_new:Npn \__pdf_backend_version_minor:
2975 {
2976   \tex_the:D
2977   \*luatex
2978   \tex_pdfvariable:D minorversion
2979   \</luatex>
2980   \*pdftex
2981   \tex_pdfminorversion:D
2982   \</pdftex>
2983 }

```

(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>.

```

2984 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2985 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2986 \cs_new_protected:Npn \__pdf_backend_emc:
2987 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2988 \</luatex | pdftex>

```

6.4 dvipdfmx backend

```

2989 \*dvipdfmx | xetex>

```

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

```

2990 \cs_new_protected:Npx \__pdf_backend:n #1
2991 { \__kernel_backend_literal:n { pdf: #1 } }
2992 \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

```

```

2993 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2994 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2995 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2996 { \__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
\g__pdf_backend_object_prop

For tracking objects to allow finalisation.

```
2997 \int_new:N \g__pdf_backend_object_int
2998 \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

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

```
2999 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
3000 {
3001   \int_gincr:N \g__pdf_backend_object_int
3002   \int_const:cn
3003   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
3004   { \g__pdf_backend_object_int }
3005   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
3006 }
3007 \cs_new:Npn \__pdf_backend_object_ref:n #1
3008 { @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: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

This is where we choose the actual type.

```
3009 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
3010 {
3011   \exp_args:Nx \__pdf_backend_object_write:nnn
3012   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
3013 }
3014 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
3015 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
3016 {
3017   \use:c { __pdf_backend_object_write_ #1 :nn }
3018   { \__pdf_backend_object_ref:n {#2} } {#3}
3019 }
3020 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
3021 {
3022   \__pdf_backend:x
3023   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
3024 }
3025 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
3026 {
3027   \__pdf_backend:x
3028   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
3029 }
3030 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
3031 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
3032 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
3033 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
3034 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
3035 {
3036   \__pdf_backend:x
3037   {
3038     #1 stream ~ #2 ~
3039     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
3040   }
```

```
3041 }
```

(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`

```
3042 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
3043 {
3044   \int_gincr:N \g__pdf_backend_object_int
3045   \exp_args:Nnx \use:c { \_pdf_backend_object_write_ #1 :nn }
3046   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
3047   {#2}
3048 }
3049 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }
```

(End definition for `_pdf_backend_object_now:nn`.)

`_pdf_backend_object_last:`

```
3050 \cs_new:Npn \_pdf_backend_object_last:
3051 { @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_YTeX.

```
3052 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
3053 { @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.

```
3054 \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.

```
3055 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
3056 {
3057   \int_gincr:N \g__pdf_backend_object_int
3058   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
3059   \_pdf_backend:x
3060   {
3061     ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
3062     width ~ \dim_eval:n {#1} ~
3063     height ~ \dim_eval:n {#2} ~
3064     depth ~ \dim_eval:n {#3} ~
3065     << /Type /Annot #4 >>
3066   }
3067 }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

`_pdf_backend_annotation_last:`

```
3068 \cs_new:Npn \_pdf_backend_annotation_last:
3069 { @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.

```
3070 \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:

```
3071 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
3072 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
3073 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
3074 { \_pdf_backend_link_begin:n {#1#2} }
3075 \cs_new_protected:Npx \_pdf_backend_link_begin:n #1
3076 {
3077   \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int
3078   \_pdf_backend:x
3079   {
3080     bann ~
3081     @pdf.lnk
3082     \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int
3083     \c_space_tl
3084     <<
3085     /Type /Annot
3086     #1
3087     >>
3088   }
3089 }
3090 \cs_new_protected:Npn \_pdf_backend_link_end:
3091 { \_pdf_backend:n { eann } }
```

(End definition for _pdf_backend_link_begin_goto:nnw and others.)

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

```
3092 \cs_new:Npn \_pdf_backend_link_last:
3093 { @pdf.lnk \int_use:N \g_pdf_backend_link_int }
```

(End definition for _pdf_backend_link_last:.)

_pdf_backend_link_margin:n Pass to dvipdfmx.

```
3094 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
3095 { \_kernel_backend_literal:x { dvipdfmx:config-g~ \dim_eval:n {#1} } }
```

(End definition for _pdf_backend_link_margin:n.)

_pdf_backend_destination:nn

_pdf_backend_destination:nnnn

_pdf_backend_destination_aux:nnnn

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.

```
3096 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
3097 {
3098   \_pdf_backend:x
3099   {
3100     dest ~ ( \exp_not:n {#1} )
3101     [
3102       @thispage
```



```

3103 \str_case:nnF {#2}
3104 {
3105   { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
3106   { fit } { /Fit }
3107   { fitb } { /FitB }
3108   { fitbh } { /FitBH }
3109   { fitbv } { /FitBV ~ @xpos }
3110   { fith } { /FitH ~ @ypos }
3111   { fitv } { /FitV ~ @xpos }
3112   { fitr } { /Fit }
3113 }
3114 { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
3115 ]
3116 }
3117 }
3118 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
3119 {
3120   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
3121   { \dim_eval:n {#2} } {#1} {#3} {#4}
3122 }
3123 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
3124 {
3125   \vbox_to_zero:n
3126   {
3127     \__kernel_kern:n {#4}
3128     \hbox:n
3129     {
3130       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
3131       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
3132     }
3133     \tex_vss:D
3134   }
3135   \__kernel_kern:n {#1}
3136   \vbox_to_zero:n
3137   {
3138     \__kernel_kern:n { -#3 }
3139     \hbox:n
3140     {
3141       \__pdf_backend:n
3142       {
3143         dest ~ (#2)
3144         [
3145           @thispage
3146           /FitR ~
3147           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
3148           @xpos ~ @ypos
3149         ]
3150       }
3151     }
3152     \tex_vss:D
3153   }
3154   \__kernel_kern:n { -#1 }
3155 }

```

(End definition for __pdf_backend_destination:nn, __pdf_backend_destination:nnnn, and __-

`pdf_backend_destination_aux:nnnn.)`

6.4.4 Structure

`_pdf_backend_compresslevel:n`
`_pdf_backend_compress_objects:n`

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

```

3156 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
3157   { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
3158 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
3159   {
3160     \bool_if:nF {#1}
3161     { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
3162   }

```

(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`

We start with the assumption that the default is active.

```

3163 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
3164   {
3165     \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
3166     \__kernel_backend_literal:x { pdf:majorversion~ \_pdf_backend_version_major: }
3167   }
3168 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
3169   {
3170     \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
3171     \__kernel_backend_literal:x { pdf:minorversion~ \_pdf_backend_version_minor: }
3172   }

```

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

`_pdf_backend_version_major:`
`_pdf_backend_version_minor:`

We start with the assumption that the default is active.

```

3173 \cs_new:Npn \_pdf_backend_version_major: { 1 }
3174 \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`
`_pdf_backend_emc:`

Simple wrappers. May need refinement: see <https://chat.stackexchange.com/transcript/message/49970158#49970158>.

```

3175 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
3176   { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
3177 \cs_new_protected:Npn \_pdf_backend_emc:
3178   { \__kernel_backend_literal_page:n { EMC } }

```

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

```

3179 </dvipdfmx | xetex>

```

6.5 dvisvgm backend

3180 $\langle *dvisvgm \rangle$

6.5.1 Catalogue entries

No-op.

3181 $\backslash cs_new_protected:Npn _pdf_backend_catalog_gput:nn \#1\#2 \{ \}$
3182 $\backslash cs_new_protected:Npn _pdf_backend_info_gput:nn \#1\#2 \{ \}$

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

6.5.2 Objects

All no-ops here.

3183 $\backslash cs_new_protected:Npn _pdf_backend_object_new:nn \#1\#2 \{ \}$
3184 $\backslash cs_new:Npn _pdf_backend_object_ref:n \#1 \{ \}$
3185 $\backslash cs_new_protected:Npn _pdf_backend_object_write:nn \#1\#2 \{ \}$
3186 $\backslash cs_new_protected:Npn _pdf_backend_object_write:nx \#1\#2 \{ \}$
3187 $\backslash cs_new_protected:Npn _pdf_backend_object_now:nn \#1\#2 \{ \}$
3188 $\backslash cs_new_protected:Npn _pdf_backend_object_now:nx \#1\#2 \{ \}$
3189 $\backslash cs_new:Npn _pdf_backend_object_last: \{ \}$
3190 $\backslash cs_new:Npn _pdf_backend_pageobject_ref:n \#1 \{ \}$

(End definition for $_pdf_backend_object_new:nn$ and others.)

6.5.3 Structure

These are all no-ops.

3191 $\backslash cs_new_protected:Npn _pdf_backend_compresslevel:n \#1 \{ \}$
3192 $\backslash cs_new_protected:Npn _pdf_backend_compress_objects:n \#1 \{ \}$

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

Data not available!

3193 $\backslash cs_new_protected:Npn _pdf_backend_version_major_gset:n \#1 \{ \}$
3194 $\backslash cs_new_protected:Npn _pdf_backend_version_minor_gset:n \#1 \{ \}$

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

Data not available!

3195 $\backslash cs_new:Npn _pdf_backend_version_major: \{ -1 \}$
3196 $\backslash cs_new:Npn _pdf_backend_version_minor: \{ -1 \}$

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

More no-ops.

3197 $\backslash cs_new_protected:Npn _pdf_backend_bdc:nn \#1\#2 \{ \}$
3198 $\backslash cs_new_protected:Npn _pdf_backend_emc: \{ \}$

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

3199 $\langle /dvisvgm \rangle$

3200 $\langle /package \rangle$

7 l3backend-opacity Implementation

```

3201 <*package>
3202 <@@=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.

```

3203 <*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.

```

3204 \cs_new_protected:Npn \__opacity_backend_select:n #1
3205 {
3206   \exp_args:Nx \__opacity_backend_select_aux:n
3207   { \fp_eval:n { min(max(0,#1),1) } }
3208 }
3209 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3210 {
3211   \__opacity_backend:nnn {#1} { fill } { ca }
3212   \__opacity_backend:nnn {#1} { stroke } { CA }
3213 }
3214 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3215 {
3216   \__opacity_backend:xnn
3217   { \fp_eval:n { min(max(0,#1),1) } }
3218   { fill }
3219   { ca }
3220 }
3221 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3222 {
3223   \__opacity_backend:xnn
3224   { \fp_eval:n { min(max(0,#1),1) } }
3225   { stroke }
3226   { CA }
3227 }
3228 \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3229 {
3230   \__kernel_backend_postscript:n
3231   {
3232     product ~ (Ghostscript) ~ search
3233     {
3234       pop ~ pop ~ pop ~
3235       #1 ~ .set #2 constantalpha
3236     }
3237     {
3238       pop ~
3239       mark ~
3240       /#3 ~ #1

```

```

3241         /SetTransparency ~
3242         pdfmark
3243     }
3244     ifelse
3245 }
3246 }
3247 \cs_generate_variant:Nn \_opacity_backend:nnn { x }

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

3248 </dvips>
3249 <*dvipdfmx | luatex | pdftex | xetex>

```

\c__opacity_backend_stack_int Set up a stack.

```

3250 \bool_lazy_and:nnT
3251 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3252 { \pdfmanagement_if_active_p:}
3253 {
3254     \_kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3255     { page ~ direct } { /opacity 1 ~ gs }
3256     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3257     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3258 }

(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
3259 \tl_new:N \l__opacity_backend_fill_tl
3260 \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:
3261 \cs_new_protected:Npn \_opacity_backend_select:n #1
3262 {
3263     \exp_args:Nx \_opacity_backend_select_aux:n
3264     { \fp_eval:n { min(max(0,#1),1) } }
3265 }
3266 \cs_new_protected:Npn \_opacity_backend_select_aux:n #1
3267 {
3268     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3269     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3270     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3271     { opacity #1 }
3272     { << /ca ~ #1 /CA ~ #1 >> }
3273     \_kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3274     { /opacity #1 ~ gs }
3275     \group_insert_after:N \_opacity_backend_reset:
3276 }
3277 \bool_lazy_and:nnF
3278 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3279 { \pdfmanagement_if_active_p:}
3280 {
3281     \cs_gset_protected:Npn \_opacity_backend_select_aux:n #1 { }
3282 }

```

```

3283 \cs_new_protected:Npn \__opacity_backend_reset:
3284 { \__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:.)

```

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

```

\__opacity_backend_fill:n
\__opacity_backend_stroke:n
  \__opacity_backend_fillstroke:nn
  \__opacity_backend_fillstroke:xx
3285 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3286 {
3287   \__opacity_backend_fill_stroke:xx
3288   { \fp_eval:n { min(max(0,#1),1) } }
3289   \l__opacity_backend_stroke_tl
3290 }
3291 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3292 {
3293   \__opacity_backend_fill_stroke:xx
3294   \l__opacity_backend_fill_tl
3295   { \fp_eval:n { min(max(0,#1),1) } }
3296 }
3297 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3298 {
3299   \str_if_eq:nnTF {#1} {#2}
3300   { \__opacity_backend_select_aux:n {#1} }
3301   {
3302     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3303     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3304     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3305     { opacity.fill #1 }
3306     { << /ca ~ #1 >> }
3307     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3308     { opacity.stroke #1 }
3309     { << /CA ~ #2 >> }
3310     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3311     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3312     \group_insert_after:N \__opacity_backend_reset:
3313   }
3314 }
3315 \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.)

```

3316 </dviptfm | luatex | pdftex | xetex>
3317 <*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.

```

\__opacity_backend_select:n
  \__opacity_backend_fill:n
\__opacity_backend_stroke:n
  \__opacity_backend:nn
3318 \cs_new_protected:Npn \__opacity_backend_select:n #1
3319 { \__opacity_backend:nn {#1} { } }
3320 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3321 { \__opacity_backend:nn {#1} { fill- } }
3322 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3323 { \__opacity_backend:nn { {#1} } { stroke- } }
3324 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3325 { \__kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

```

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

3326 `</dvisvgm>`

3327 `</package>`

8 l3backend-header Implementation

3328 `<*dvips & header>`

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

3329 `/color.sc { } def`

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

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

3330 `TeXDict begin`

3331 `/TeXcolorseparation { setcolor } def`

3332 `end`

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

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

3333 `true setglobal`

3334 `/pdf.globaldict 4 dict def`

3335 `false setglobal`

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

`pdf.cvs` Small utilities for PostScript manipulations. Conversion to DVI dimensions is done here
`pdf.dvi.pt` to allow for `Resolution`. The total height of a rectangle (an array) needs a little maths,
`pdf.pt.dvi` in contrast to simply extracting a value.

`pdf.rect.ht` 3336 `/pdf.cvs { 65534 string cvs } def`

3337 `/pdf.dvi.pt { 72.27 mul Resolution div } def`

3338 `/pdf.pt.dvi { 72.27 div Resolution mul } def`

3339 `/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` 3340 `/pdf.linkmargin { 1 pdf.pt.dvi } def`

`pdf.linkht.pad` 3341 `/pdf.linkdp.pad { 0 } def`

3342 `/pdf.linkht.pad { 0 } def`

(End definition for `pdf.linkmargin`, `pdf.linkdp.pad`, and `pdf.linkht.pad`. These functions are documented on page ??.)

`pdf.rect` Functions for marking the limits of an annotation/link, plus drawing the border. We
`pdf.save.ll` separate links for generic annotations to support adding a margin and setting a minimal
`pdf.save.ur` size.

`pdf.save.linkll` 3343 `/pdf.rect`

`pdf.save.linkur` 3344 `{ /Rect [pdf.llx pdf.lly pdf.urx pdf.ury] } def`

`pdf.llx` 3345 `/pdf.save.ll`

`pdf.lly` 3346 `{`

`pdf.urx`

`pdf.ury`

```

3347     currentpoint
3348     /pdf.lly exch def
3349     /pdf.llx exch def
3350   }
3351   def
3352 /pdf.save.ur
3353   {
3354     currentpoint
3355     /pdf.ury exch def
3356     /pdf.urx exch def
3357   }
3358   def
3359 /pdf.save.linkll
3360   {
3361     currentpoint
3362     pdf.linkmargin add
3363     pdf.linkdp.pad add
3364     /pdf.lly exch def
3365     pdf.linkmargin sub
3366     /pdf.llx exch def
3367   }
3368   def
3369 /pdf.save.linkur
3370   {
3371     currentpoint
3372     pdf.linkmargin sub
3373     pdf.linkht.pad sub
3374     /pdf.ury exch def
3375     pdf.linkmargin add
3376     /pdf.urx exch def
3377   }
3378   def

```

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

`pdf.dest.anchor` For finding the anchor point of a destination link. We make the use case a separate function as it comes up a lot, and as this makes it easier to adjust if we need additional effects. We also need a more complex approach to convert a co-ordinate pair correctly when defining a rectangle: this can otherwise be out when using a landscape page. (Thanks to Alexander Grahn for the approach here.)

```

pdf.dev.x 3379 /pdf.dest.anchor
pdf.dev.y 3380   {
pdf.tmpa 3381     currentpoint exch
pdf.tmpb 3382     pdf.dvi.pt 72 add
pdf.tmpc 3383     /pdf.dest.x exch def
pdf.tmpd 3384     pdf.dvi.pt
3385     vsize 72 sub exch sub
3386     /pdf.dest.y exch def
3387   }
3388   def
3389 /pdf.dest.point
3390   { pdf.dest.x pdf.dest.y } def
3391 /pdf.dest2device
3392   {

```



```

3393 /pdf.dest.y exch def
3394 /pdf.dest.x exch def
3395 matrix currentmatrix
3396 matrix defaultmatrix
3397 matrix invertmatrix
3398 matrix concatmatrix
3399 cvx exec
3400 /pdf.dev.y exch def
3401 /pdf.dev.x exch def
3402 /pdf.tmpd exch def
3403 /pdf.tmpc exch def
3404 /pdf.tmpb exch def
3405 /pdf.tmpa exch def
3406 pdf.dest.x pdf.tmpa mul
3407 pdf.dest.y pdf.tmpc mul add
3408 pdf.dev.x add
3409 pdf.dest.x pdf.tmpb mul
3410 pdf.dest.y pdf.tmpd mul add
3411 pdf.dev.y add
3412 }
3413 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	3414 /pdf.bordertracking false def
pdf.brokenlink.skip	3415 /pdf.bordertracking.begin
pdf.brokenlink.dict	3416 {
pdf.bordertracking.endpage	3417 SDict /pdf.bordertracking true put
pdf.bordertracking.continue	3418 SDict /pdf.leftboundary undef
pdf.originx	3419 SDict /pdf.rightboundary undef
pdf.originy	3420 /a where
	3421 {
	3422 /a
	3423 {
	3424 currentpoint pop
	3425 SDict /pdf.rightboundary known dup
	3426 {
	3427 SDict /pdf.rightboundary get 2 index lt
	3428 { not }
	3429 if
	3430 }
	3431 if
	3432 { pop }
	3433 { SDict exch /pdf.rightboundary exch put }
	3434 ifelse
	3435 moveto
	3436 currentpoint pop
	3437 SDict /pdf.leftboundary known dup
	3438 {
	3439 SDict /pdf.leftboundary get 2 index gt

```

3440         { not }
3441     if
3442     }
3443     if
3444     { pop }
3445     { SDict exch /pdf.leftboundary exch put }
3446     ifelse
3447 }
3448 put
3449 }
3450 if
3451 }
3452 def
3453 /pdf.bordertracking.end
3454 {
3455     /a where { /a { moveto } put } if
3456     /x where { /x { 0 exch rmoveto } put } if
3457     SDict /pdf.leftboundary known
3458     { pdf.outerbox 0 pdf.leftboundary put }
3459     if
3460     SDict /pdf.rightboundary known
3461     { pdf.outerbox 2 pdf.rightboundary put }
3462     if
3463     SDict /pdf.bordertracking false put
3464 }
3465 def
3466 /pdf.bordertracking.endpage
3467 {
3468     pdf.bordertracking
3469     {
3470         pdf.bordertracking.end
3471         true setglobal
3472         pdf.globaldict
3473         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3474         pdf.globaldict
3475         /pdf.brokenlink.skip pdf.baselineskip put
3476         pdf.globaldict
3477         /pdf.brokenlink.dict
3478         pdf.link.dict pdf.cvs put
3479         false setglobal
3480         mark pdf.link.dict cvx exec /Rect
3481         [
3482             pdf.llx
3483             pdf.lly
3484             pdf.outerbox 2 get pdf.linkmargin add
3485             currentpoint exch pop
3486             pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3487         ]
3488         /ANN pdf.pdfmark
3489     }
3490     if
3491 }
3492 def
3493 /pdf.bordertracking.continue

```

```

3494 {
3495   /pdf.link.dict pdf.globaldict
3496   /pdf.brokenlink.dict get def
3497   /pdf.outerbox pdf.globaldict
3498   /pdf.brokenlink.rect get def
3499   /pdf.baselineskip pdf.globaldict
3500   /pdf.brokenlink.skip get def
3501   pdf.globaldict dup dup
3502   /pdf.brokenlink.dict undef
3503   /pdf.brokenlink.skip undef
3504   /pdf.brokenlink.rect undef
3505   currentpoint
3506   /pdf.originy exch def
3507   /pdf.originx exch def
3508   /a where
3509   {
3510     /a
3511     {
3512       moveto
3513       SDict
3514       begin
3515         currentpoint pdf.originy ne exch
3516         pdf.originx ne or
3517         {
3518           pdf.save.linkll
3519           /pdf.lly
3520             pdf.lly pdf.outerbox 1 get sub def
3521           pdf.bordertracking.begin
3522         }
3523         if
3524         end
3525       }
3526       put
3527     }
3528   if
3529   /x where
3530   {
3531     /x
3532     {
3533       0 exch rmoveto
3534       SDict
3535       begin
3536         currentpoint
3537         pdf.originy ne exch pdf.originx ne or
3538         {
3539           pdf.save.linkll
3540           /pdf.lly
3541             pdf.lly pdf.outerbox 1 get sub def
3542           pdf.bordertracking.begin
3543         }
3544         if
3545         end
3546       }
3547       put

```

```

3548     }
3549     if
3550 }
3551 def

```

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

```

pdf.breaklink
pdf.breaklink.write
pdf.count
pdf.currentrect

```

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

```

3552 /pdf.breaklink
3553 {
3554     pop
3555     counttomark 2 mod 0 eq
3556     {
3557         counttomark /pdf.count exch def
3558         {
3559             pdf.count 0 eq { exit } if
3560             counttomark 2 roll
3561             1 index /Rect eq
3562             {
3563                 dup 4 array copy
3564                 dup dup
3565                 1 get
3566                 pdf.outerbox pdf.rect.ht
3567                 pdf.linkmargin 2 mul add sub
3568                 3 exch put
3569                 dup
3570                 pdf.outerbox 2 get
3571                 pdf.linkmargin add
3572                 2 exch put
3573                 dup dup
3574                 3 get
3575                 pdf.outerbox pdf.rect.ht
3576                 pdf.linkmargin 2 mul add add
3577                 1 exch put
3578                 /pdf.currentrect exch def
3579                 pdf.breaklink.write
3580                 {
3581                     pdf.currentrect
3582                     dup
3583                     pdf.outerbox 0 get
3584                     pdf.linkmargin sub
3585                     0 exch put
3586                     dup
3587                     pdf.outerbox 2 get
3588                     pdf.linkmargin add
3589                     2 exch put
3590                     dup dup
3591                     1 get
3592                     pdf.baselineskip add
3593                     1 exch put

```

```

3594         dup dup
3595         3 get
3596         pdf.baselineskip add
3597         3 exch put
3598         /pdf.currentrect exch def
3599         pdf.breaklink.write
3600     }
3601     1 index 3 get
3602     pdf.linkmargin 2 mul add
3603     pdf.outerbox pdf.rect.ht add
3604     2 index 1 get sub
3605     pdf.baselineskip div round cvi 1 sub
3606     exch
3607     repeat
3608     pdf.currentrect
3609     dup
3610         pdf.outerbox 0 get
3611         pdf.linkmargin sub
3612         0 exch put
3613     dup dup
3614         1 get
3615         pdf.baselineskip add
3616         1 exch put
3617     dup dup
3618         3 get
3619         pdf.baselineskip add
3620         3 exch put
3621     dup 2 index 2 get 2 exch put
3622     /pdf.currentrect exch def
3623     pdf.breaklink.write
3624     SDict /pdf.pdfmark.good false put
3625     exit
3626 }
3627 { pdf.count 2 sub /pdf.count exch def }
3628 ifelse
3629 }
3630 loop
3631 }
3632 if
3633 /ANN
3634 }
3635 def
3636 /pdf.breaklink.write
3637 {
3638     counttomark 1 sub
3639     index /_objdef eq
3640     {
3641         counttomark -2 roll
3642         dup wcheck
3643         {
3644             readonly
3645             counttomark 2 roll
3646         }
3647         { pop pop }

```

```

3648         ifelse
3649     }
3650     if
3651     counttomark 1 add copy
3652     pop pdf.currentrect
3653     /ANN pdfmark
3654 }
3655 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.

```

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

```

```

3694     pop
3695     {
3696         dup mark eq
3697         {
3698             pop
3699             exit
3700         }
3701         {
3702             pdf.pdfmark.dict
3703             begin def end
3704         }
3705         ifelse
3706     }
3707     loop
3708 }
3709 def

```

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

```

3710 </dvips & header>

```

Index

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

A	
<code>\AtBeginDvi</code>	57
B	
bool commands:	
<code>\bool_gset_false:N</code>	1225 , 1244 , 1267 , 1289 , 1305 , 1406 , 1645 , 1681 , 2430 , 2476
<code>\bool_gset_true:N</code>	1223 , 1292 , 1404 , 1660 , 2423 , 2429
<code>\bool_if:NTF</code>	67 , 678 , 1235 , 1239 , 1255 , 1258 , 1262 , 1273 , 1280 , 1284 , 1296 , 1300 , 1417 , 1422 , 1427 , 1619 , 1664 , 1803 , 1847 , 1986 , 2028 , 2418 , 2433 , 2438 , 2443
<code>\bool_if:nTF</code>	2652 , 2918 , 3160
<code>\bool_lazy_and:nnTF</code>	894 , 968 , 2145 , 3250 , 3277
<code>\bool_lazy_or:nnTF</code>	1839 , 2021
<code>\bool_new:N</code>	1226 , 1293 , 1407 , 1661 , 2403 , 2404
<code>\bool_set_false:N</code>	1814 , 1950 , 2052 , 2216
box commands:	
<code>\box_dp:N</code>	208 , 210 , 258 , 260 , 315 , 317 , 364 , 366 , 368 , 370 , 2455 , 2488 , 2489 , 2514
<code>\box_ht:N</code>	210 , 260 , 317 , 368 , 370 , 1859 , 2093 , 2460 , 2499 , 2500 , 2516
<code>\box_if_empty:N</code>	2549
<code>\box_move_down:nn</code>	2377 , 2455
<code>\box_move_up:nn</code>	2237 , 2379 , 2460
<code>\box_new:N</code>	2262 , 2367 , 2368
<code>\box_set_dp:Nn</code>	1744
<code>\box_set_ht:Nn</code>	1743
<code>\box_set_wd:Nn</code>	272 , 1742
<code>\box_use:N</code>	215 , 233 , 247 , 263 , 290 , 304 , 320 , 336 , 348 , 399 , 413 , 432 , 1357 , 1552 , 1745 , 2408
<code>\box_wd:N</code>	209 , 217 , 259 , 265 , 316 , 322 , 365 , 367 , 1858 , 2092
box internal commands:	
<code>__box_backend_clip:N</code>	197 , 252 , 309 , 353
<code>\l_box_backend_cos_fp</code>	267
<code>__box_backend_rotate:Nn</code>	219 , 267 , 324 , 403
<code>__box_backend_rotate_aux:Nn</code> ...	219 , 267 , 324
<code>__box_backend_scale:Nnn</code>	236 , 295 , 339 , 416
<code>\l_box_backend_sin_fp</code>	267
C	
clist commands:	
<code>\clist_map_function:nN</code>	1313 , 1437 , 1688
color internal commands:	
<code>__color_backend:nnn</code>	1109
<code>__color_backend_cmyk:w</code>	1110
<code>\g_color_backend_colorant_prop</code> ..	644 , 663 , 666 , 686 , 911
<code>__color_backend_devicen_-colorants:n</code>	645 , 847 , 966
<code>__color_backend_devicen_-colorants:w</code>	645
<code>__color_backend_devicen_-init:nnn</code>	834 , 936 , 1164
<code>__color_backend_devicen_init:w</code> ..	936
<code>__color_backend_fill:n</code>	1039 , 1066 , 1073 , 1091 , 1098
<code>__color_backend_fill_cmyk:n</code> ...	1039 , 1073 , 1098
<code>__color_backend_fill_devicen:nn</code> ..	1065 , 1090 , 1160
<code>__color_backend_fill_gray:n</code> ...	1039 , 1073 , 1098
<code>__color_backend_fill_rgb:n</code> ...	1039 , 1073 , 1098
<code>__color_backend_fill_separation:nn</code> ..	1065 , 1090 , 1160
<code>\l_color_backend_fill_tl</code>	618 , 628 , 1047 , 1062
<code>__color_backend_iccbased_-device:nnn</code>	1013
<code>__color_backend_iccbased_-init:nnn</code>	853 , 987 , 1164
<code>\c_color_backend_main_stack_int</code> ..	500
<code>__color_backend_pickup:N</code> ..	440 , 463
<code>__color_backend_pickup:w</code> ..	14 , 440 , 463
<code>__color_backend_reset:</code>	599 , 620 , 1050 , 1063 , 1082 , 1107
<code>__color_backend_rgb:w</code>	1133
<code>__color_backend_select:n</code> ..	599 , 673
<code>__color_backend_select:nn</code> ..	620 , 879

_color_backend_select_cmyk:n ..	_color_backend_stroke_devicen:nn
..... 599, 620 1065, 1090, 1160
_color_backend_select_devicen:nn	_color_backend_stroke_gray:n ..
..... 672, 856, 878 1039, 1073, 1109
_color_backend_select_gray:n ..	_color_backend_stroke_gray-
..... 599, 620, 637	aux:n 1109
_color_backend_select_iccbased:nn	_color_backend_stroke_rgb:n ...
..... 675, 860, 878 1039, 1073, 1109
_color_backend_select_named:n .	_color_backend_stroke_rgb:w . 1109
..... 599, 634	_color_backend_stroke_separation:nn
_color_backend_select_rgb:n 1065, 1090, 1160
..... 599, 620	\l_color_backend_stroke_tl
_color_backend_select_separation:nn 618, 629, 1049, 1060
..... 672, 856, 878	\g_color_model_int
_color_backend_separation-	... 683, 692, 840, 868, 902, 976, 1008
init:n 676	\c_color_model_range_CIELAB_tl .
_color_backend_separation- 795, 830, 925, 932
init:nn 882	color.sc 599, 3329
_color_backend_separation-	cs commands:
init:nnn 676	_cs_generate_variant:Nn .. 49, 63,
_color_backend_separation-	66, 99, 138, 143, 154, 185, 191, 550,
init:nnnn 676	586, 697, 1172, 1367, 1561, 2000,
_color_backend_separation-	2063, 2083, 2266, 2303, 2362, 2854,
init:nnnnn 676, 858, 882	2882, 2992, 3014, 3049, 3247, 3315
_color_backend_separation-	_cs_gset:Npx .. 2664, 2668, 3165, 3170
init:nw 676	_cs_gset_protected:Npn 532, 3281
_color_backend_separation-	_cs_if_exist:NTF . 27, 50, 441, 464,
init:w 676	520, 1003, 1026, 1755, 2545, 2943, 2969
_color_backend_separation-	_cs_if_exist_p:N . 895, 969, 3251, 3278
init_/DeviceCMYK:nnn 676	_cs_if_exist_use:NTF 38, 710
_color_backend_separation-	_cs_new:Npn 660, 719, 721, 723, 725,
init_/DeviceGray:nnn 676	732, 738, 740, 746, 763, 770, 772,
_color_backend_separation-	981, 1318, 1442, 1692, 1861, 2096,
init_/DeviceRGB:nnn 676	2254, 2281, 2363, 2365, 2398, 2570,
_color_backend_separation-	2670, 2671, 2824, 2855, 2856, 2974,
init_aux:nnnnnn 676	3007, 3050, 3052, 3068, 3092, 3173,
_color_backend_separation-	3174, 3184, 3189, 3190, 3195, 3196
init_CIELAB:nnn 676, 858, 882	_cs_new:Npx
_color_backend_separation-	... 645, 2691, 2726, 2883, 2894, 2961
init_CIELAB:nnnnnn 859	_cs_new_eq:NN
_color_backend_separation-	... 46, 57, 59, 674, 857, 880, 881,
init_count:n 676	1069, 1070, 1094, 1095, 1162, 1163,
_color_backend_separation-	1171, 1366, 1372, 1373, 1560, 1562,
init_count:w 676	1563, 1569, 1769, 1770, 1783, 1785,
_color_backend_separation-	1809, 1810, 1867, 1868, 1869, 1892,
init_Device:Nn 676	1917, 1934, 1935, 1944, 1945, 1946,
\g_color_backend_stack_int ... 500	1966, 1969, 1970, 1971, 2036, 2046,
\l_color_backend_stack_int	2047, 2048, 2202, 2203, 2211, 2212,
... 497, 522, 528, 630, 633, 1048, 1061	2221, 2251, 2252, 2253, 2257, 2408
_color_backend_stroke:n 1039, 1068	_cs_new_protected:Npn
_color_backend_stroke_cmyk:n 47, 54, 61, 64, 72,
..... 1039, 1073, 1109	78, 83, 85, 89, 100, 110, 119, 128,
_color_backend_stroke_cmyk:w 1109	141, 144, 146, 148, 152, 157, 166,
	176, 186, 197, 219, 221, 236, 252,

267, 269, 295, 309, 324, 326, 339,
 353, 403, 416, 440, 458, 463, 471,
 541, 551, 562, 576, 587, 599, 601,
 603, 605, 607, 614, 620, 622, 624,
 626, 632, 634, 672, 675, 698, 788,
 834, 853, 856, 858, 859, 860, 878,
 882, 907, 914, 936, 987, 1013, 1039,
 1041, 1043, 1045, 1052, 1054, 1056,
 1058, 1065, 1067, 1073, 1075, 1077,
 1079, 1084, 1086, 1088, 1090, 1092,
 1098, 1100, 1102, 1104, 1109, 1111,
 1122, 1130, 1132, 1134, 1160, 1161,
 1164, 1165, 1173, 1178, 1183, 1185,
 1187, 1195, 1203, 1212, 1222, 1224,
 1227, 1229, 1246, 1251, 1269, 1291,
 1294, 1307, 1320, 1325, 1327, 1329,
 1331, 1333, 1335, 1337, 1339, 1344,
 1368, 1370, 1374, 1379, 1384, 1394,
 1403, 1405, 1408, 1410, 1412, 1414,
 1419, 1424, 1429, 1431, 1444, 1449,
 1451, 1453, 1455, 1457, 1459, 1461,
 1463, 1474, 1499, 1511, 1523, 1535,
 1542, 1564, 1570, 1575, 1580, 1591,
 1601, 1611, 1613, 1615, 1617, 1648,
 1650, 1655, 1657, 1659, 1662, 1683,
 1694, 1707, 1709, 1711, 1713, 1715,
 1717, 1719, 1721, 1723, 1731, 1753,
 1772, 1795, 1811, 1823, 1828, 1836,
 1862, 1875, 1893, 1903, 1919, 1938,
 1947, 1955, 1967, 1973, 1976, 1991,
 2001, 2040, 2049, 2055, 2061, 2064,
 2071, 2084, 2089, 2097, 2104, 2121,
 2155, 2186, 2187, 2189, 2191, 2193,
 2199, 2205, 2213, 2219, 2222, 2224,
 2235, 2264, 2267, 2269, 2273, 2283,
 2304, 2309, 2314, 2319, 2329, 2334,
 2342, 2370, 2375, 2407, 2409, 2414,
 2416, 2421, 2436, 2441, 2478, 2507,
 2526, 2535, 2572, 2579, 2605, 2610,
 2638, 2650, 2662, 2666, 2672, 2674,
 2678, 2702, 2704, 2706, 2717, 2737,
 2747, 2770, 2784, 2794, 2805, 2826,
 2857, 2905, 2916, 2922, 2950, 2984,
 2986, 2993, 2995, 2999, 3009, 3015,
 3020, 3025, 3030, 3032, 3034, 3042,
 3055, 3071, 3073, 3090, 3094, 3096,
 3118, 3123, 3156, 3158, 3163, 3168,
 3175, 3177, 3181, 3182, 3183, 3185,
 3186, 3187, 3188, 3191, 3192, 3193,
 3194, 3197, 3198, 3204, 3209, 3214,
 3221, 3228, 3261, 3266, 3283, 3285,
 3291, 3297, 3318, 3320, 3322, 3324
 \cs_new_protected:Npx
 501, 676, 1145, 2933, 2990, 3075

\cs_set_eq:NN 2566, 2567
 \cs_set_protected:Npn 443, 466, 2159

D

dim commands:

\dim_compare:nNnTF 2135, 2140
 \dim_compare_p:nNn 2146, 2147
 \dim_eval:n 2373, 2608,
 2686, 2687, 2688, 2745, 2780, 2781,
 2782, 3062, 3063, 3064, 3095, 3121
 \dim_max:nn 2486, 2497
 \dim_set:Nn
 .. 1858, 1859, 2092, 2093, 2131, 2132
 \dim_set_eq:NN 2197
 \dim_to_decimal:n .. 364, 365, 366,
 367, 368, 370, 1573, 1578, 1584,
 1585, 1586, 1587, 1596, 1597, 1598,
 1689, 1708, 2244, 2245, 2484, 2495,
 2513, 2514, 2515, 2516, 2520, 2576
 \dim_to_decimal_in_bp:n
 208, 209, 210, 258, 259, 260,
 315, 316, 317, 1191, 1192, 1199,
 1200, 1207, 1208, 1216, 1217, 1218,
 1315, 1319, 1323, 1377, 1382, 1388,
 1389, 1390, 1398, 1399, 1439, 1443,
 1447, 1693, 1777, 1778, 1779, 1780,
 1960, 1961, 1962, 1963, 2015, 2016,
 2017, 2018, 2229, 2230, 2231, 2232
 \dim_zero:N 2129, 2130
 \c_max_dim
 .. 2131, 2132, 2135, 2140, 2146, 2147

draw internal commands:

__draw_align_currentpoint... .. 36
 __draw_backend_add_to_path:n ..
 1570, 1616
 __draw_backend_begin:
 1173, 1368, 1564
 __draw_backend_box_use:Nnnnn ..
 32, 1344, 1542, 1731
 __draw_backend_cap_but:
 1307, 1431, 1683
 __draw_backend_cap_rectangle: ..
 1307, 1431, 1683
 __draw_backend_cap_round:
 1307, 1431, 1683
 __draw_backend_clip: 1227, 1408, 1615
 __draw_backend_closepath:
 1227, 1408, 1615
 __draw_backend_closestroke: ...
 1227, 1408, 1615
 __draw_backend_cm:nnnn 1339, 1352,
 1353, 1354, 1463, 1546, 1723, 1734
 __draw_backend_cm_aux:nnnn .. 1463

<code>__draw_backend_cm_decompose:nnnnN</code>	<code>\g__draw_backend_path_int</code>	1630, 1647
..... 1469, 1498	<code>\g__draw_backend_path_tl</code> 1570, 1626, 1642, 1644, 1671
<code>__draw_backend_cm_decompose_</code>	<code>__draw_backend_rectangle:nnnn</code> 1187, 1374, 1570
<code>auxi:nnnnN</code>	<code>__draw_backend_scope_begin:</code> 1183, 1369, 1372, 1562
..... 1498	<code>__draw_backend_scope_end:</code> 1183, 1371, 1372, 1562
<code>__draw_backend_cm_decompose_</code>	<code>__draw_backend_stroke:</code> 1227, 1408, 1615
<code>auxii:nnnnN</code>	<code>\g__draw_draw_clip_bool</code>	.. 1227, 1615
..... 1498	<code>\g__draw_draw_eor_bool</code>	... 1222, 1239, 1255, 1262, 1273,
<code>__draw_backend_cm_decompose_</code>		1284, 1300, 1403, 1417, 1422, 1427
<code>auxiii:nnnnN</code>	<code>\g__draw_draw_path_int</code> 1615
..... 1498	<code>\g__draw_path_tl</code> 1680
<code>__draw_backend_curveto:nnnnnn</code>		
..... 1187, 1374, 1570		
<code>__draw_backend_dash:n</code>		
..... 1307, 1431, 1683		
<code>__draw_backend_dash_aux:nn</code>		
.. 1683		
<code>__draw_backend_dash_pattern:nn</code>		
..... 1307, 1431, 1683		
<code>__draw_backend_discardpath:</code>		
..... 1227, 1408, 1615		
<code>__draw_backend_end:</code>		
..... 1173, 1368, 1564		
<code>__draw_backend_evenodd_rule:</code>		
..... 1222, 1403, 1611		
<code>__draw_backend_fill:</code>		
..... 1227, 1408, 1615		
<code>__draw_backend_fillstroke:</code>		
..... 1227, 1408, 1615		
<code>__draw_backend_join_bevel:</code>		
..... 1307, 1431, 1683		
<code>__draw_backend_join_miter:</code>		
..... 1307, 1431, 1683		
<code>__draw_backend_join_round:</code>		
..... 1307, 1431, 1683		
<code>__draw_backend_lineto:nn</code>		
..... 1187, 1374, 1570		
<code>__draw_backend_linewidth:n</code>		
..... 1307, 1431, 1683		
<code>__draw_backend_literal:n</code>		
..... 1171, 1176, 1180, 1184,		
1186, 1189, 1197, 1205, 1214, 1228,		
1231, 1232, 1233, 1234, 1237, 1243,		
1253, 1260, 1266, 1271, 1276, 1277,		
1278, 1279, 1282, 1288, 1298, 1304,		
1309, 1322, 1326, 1328, 1330, 1332,		
1334, 1336, 1338, 1341, 1346, 1347,		
1348, 1349, 1350, 1351, 1355, 1356,		
1358, 1359, 1360, 1361, 1362, 1366,		
1376, 1381, 1386, 1396, 1409, 1411,		
1413, 1416, 1421, 1426, 1430, 1433,		
1446, 1450, 1452, 1454, 1456, 1458,		
1460, 1462, 1560, 1622, 1641, 1667		
<code>__draw_backend_miterlimit:n</code>		
..... 1307, 1431, 1683		
<code>__draw_backend_moveto:nn</code>		
..... 1187, 1374, 1570		
<code>__draw_backend_nonzero_rule:</code>		
..... 1222, 1403, 1611		
<code>__draw_backend_path:n</code>		
..... 1615		
	E	
	<code>\errmessage</code> 38
	<code>\evensidemargin</code> 2453
	exp commands:	
	<code>\exp_after:wN</code> 450, 2102
	<code>\exp_args:Ne</code> 734, 1830, 1883, 1909, 2607, 3120
	<code>\exp_args:Nf</code> 1312, 1436, 2372
	<code>\exp_args:NMf</code> 220, 268, 325
	<code>\exp_args:Nnx</code> 2359, 3045
	<code>\exp_args:NV</code> 445
	<code>\exp_args:Nx</code>	.. 680, 892, 1881, 1907,
		2316, 2331, 2449, 3011, 3206, 3263
	<code>\exp_last_unbraced:Nx</code> 454, 468
	<code>\exp_not:N</code> 503,
		504, 512, 514, 647, 653, 654, 655,
		682, 683, 686, 687, 692, 2693, 2695,
		2698, 2728, 2730, 2733, 2885, 2887,
		2890, 2896, 2898, 2901, 2938, 2939,
		2945, 2946, 2965, 2970, 3077, 3082
	<code>\exp_not:n</code>	48, 97, 108, 136, 995, 2307,
		2312, 2601, 2840, 2841, 2855, 2856,
		2868, 2869, 3023, 3028, 3039, 3100
	<code>\ExplBackendFileDate</code> 1
	F	
	file commands:	
	<code>\file_compare_timestamp:nNnTF</code>	.. 1895
	<code>\file_parse_full_name:nNNN</code>	1877, 1905
	<code>\fmtversion</code> 52
	fp commands:	
	<code>\fp_compare:nNnTF</code> 227, 274, 280, 332, 1479, 1492, 1537
	<code>\fp_eval:n</code>	.. 220, 229, 242, 243, 268,
		285, 300, 302, 325, 334, 345, 346,
		410, 425, 426, 1117, 1118, 1119,

1127, 1140, 1141, 1142, 1481, 1486,
 1487, 1494, 1504, 1505, 1506, 1507,
 1516, 1517, 1518, 1519, 1528, 1529,
 1530, 1531, 2598, 2767, 3114, 3207,
 3217, 3224, 3264, 3288, 3295, 3325
 \fp_new:N 293, 294
 \fp_set:Nn 273, 276
 \fp_use:N 279, 283, 288
 \fp_zero:N 275
 \c_zero_fp 227, 274, 280, 332, 1479, 1492

G

graphics commands:
 \l_graphics_search_ext_seq 1765, 1788, 1927, 2115
 graphics internal commands:
 __graphics_backend_dequote:w . 1795
 \l__graphics_backend_dir_str . 1870
 \l__graphics_backend_ext_str . 1870
 __graphics_backend_get_pagecount:n
 1784, 1919, 2034, 2104, 2256
 __graphics_backend_getbb_auxi:n
 1795
 __graphics_backend_getbb_-
 auxi:nN 2040
 __graphics_backend_getbb_-
 auxii:n 1795
 __graphics_backend_getbb_-
 auxii:nnN 2040
 __graphics_backend_getbb_-
 auxiii:n 1795
 __graphics_backend_getbb_-
 auxiii:nNnn 2040
 __graphics_backend_getbb_-
 auxiv:nnNnn 2040
 __graphics_backend_getbb_-
 auxv:nNnn 2040
 __graphics_backend_getbb_-
 auxvi:nNnn 2087, 2089
 __graphics_backend_getbb_bmp:n .
 1932, 2040
 __graphics_backend_getbb_eps:n .
 1767, 1870, 1932, 2200
 __graphics_backend_getbb_eps:nm
 1870
 __graphics_backend_getbb_eps:nn
 1881, 1893
 __graphics_backend_getbb_jpeg:n
 1795, 1932, 2040, 2205
 __graphics_backend_getbb_jpg:n .
 1795, 1932, 2040, 2205
 __graphics_backend_getbb_-
 pagebox:w 2040, 2102
 __graphics_backend_getbb_pdf:n .
 1795, 1901, 1932, 2040, 2213
 __graphics_backend_getbb_png:n .
 1795, 1932, 2040, 2205
 __graphics_backend_getbb_ps:n .
 1767, 1870, 1932, 2200
 __graphics_backend_getbb_svg:n 2121
 __graphics_backend_getbb_svg_-
 auxi:nNn 2121
 __graphics_backend_getbb_svg_-
 auxii:w 2121
 __graphics_backend_getbb_svg_-
 auxiii:Nw 2121
 __graphics_backend_getbb_svg_-
 auxiv:Nw 2121
 __graphics_backend_getbb_svg_-
 auxv:Nw 2121
 __graphics_backend_getbb_svg_-
 auxvi:Nn 2121
 __graphics_backend_getbb_svg_-
 auxvii:w 2121
 __graphics_backend_include:nn 2219
 __graphics_backend_include_-
 auxi:nn 1955
 __graphics_backend_include_-
 auxii:nnn 1955
 __graphics_backend_include_-
 auxiii:nnn 1955
 __graphics_backend_include_-
 bmp:n 1955
 __graphics_backend_include_-
 dequote:w 2235
 __graphics_backend_include_-
 eps:n 1772, 1870, 1955, 2219
 __graphics_backend_include_-
 jpeg:n 1862, 1969, 2235
 __graphics_backend_include_-
 jpg:n 1862, 1955, 2235
 __graphics_backend_include_-
 jpseg:n 1955
 __graphics_backend_include_-
 pdf:n . 1862, 1907, 1955, 2097, 2219
 __graphics_backend_include_-
 png:n 1862, 1955, 2235
 __graphics_backend_include_ps:n
 1772, 1870, 1955, 2219
 __graphics_backend_include_-
 svg:n 2235
 __graphics_backend_loaded:n ...
 ... 1753, 1765, 1767, 1784, 1788,
 1927, 1932, 2035, 2115, 2200, 2256
 \l__graphics_backend_name_str . 1870
 __graphics_bb_restore:nTF
 1825, 2086, 2123

`__graphics_bb_save:n` 1834, 2094, 2150
`\l__graphics_decodearray_str` ...
 1801, 1802,
 1813, 1841, 1845, 1846, 1949, 1984,
 1985, 2023, 2026, 2027, 2051, 2215
`__graphics_extract_bb:n`
 1942, 1951, 2209, 2217
`\l__graphics_final_name_str` .. 1900
`__graphics_get_pagecount:n`
 1785, 2036, 2257
`\l__graphics_graphics_attr_tl` ...
 1794, 1799,
 1806, 1815, 1825, 1832, 1834, 1865
`\l__graphics_internal_box`
 1856, 1858, 1859, 2091, 2092, 2093
`\l__graphics_internal_dim` 2196, 2197
`\l__graphics_internal_ior`
 2125, 2126, 2133, 2152
`\l__graphics_interpolate_bool` ...
 1803, 1814, 1840, 1847,
 1950, 1986, 2022, 2028, 2052, 2216
`\l__graphics_llx_dim`
 1777, 1960, 2015, 2129, 2229
`\l__graphics_lly_dim`
 1778, 1961, 2016, 2130, 2230
`\l__graphics_page_int`
 1797, 1818, 1819, 1851,
 1852, 1940, 1982, 1983, 2009, 2010,
 2042, 2057, 2058, 2100, 2101, 2207
`\l__graphics_pagebox_tl`
 55, 1798, 1817,
 1853, 1854, 1941, 1980, 1981, 2011,
 2013, 2043, 2066, 2067, 2102, 2208
`__graphics_read_bb:n`
 1769, 1770, 1934, 1935, 2202, 2203
`\g__graphics_track_int`
 1954, 2003, 2004
`\l__graphics_urx_dim`
 1779, 1858, 1962, 2017, 2092,
 2131, 2135, 2138, 2146, 2231, 2244
`\l__graphics_ury_dim`
 1780, 1859, 1963, 2018, 2093, 2132,
 2140, 2143, 2147, 2232, 2237, 2245
group commands:
`\group_begin:` 163, 182
`\group_end:` 171
`\group_insert_after:N`
 1050, 1063, 1082, 1107, 3275, 3312

H

hbox commands:
`\hbox:n` 2239, 2378, 2381,
 2456, 2462, 2615, 2622, 3128, 3139

`\hbox_overlap_right:n` 215,
 247, 263, 304, 320, 348, 432, 1357, 1552
`\hbox_set:Nn` .. 1856, 2091, 2448, 2480
`\hbox_set:Nw` 2431
`\hbox_set_end:` 2446
`\hbox_unpack:N` 2567
hook commands:
`\hook_gput_code:nnn` .. 55, 1755, 1757

I

int commands:
`\int_compare:nNnTF`
 1818, 1851, 1982, 2009,
 2057, 2100, 2539, 2640, 2936, 2964
`\int_const:Nn` . 504, 530, 564, 1832,
 1922, 2004, 2106, 2276, 2814, 3002
`\int_eval:n`
 546, 556, 584, 595, 730, 739, 752,
 754, 758, 771, 2664, 2668, 2914,
 2939, 2946, 2959, 3157, 3165, 3170
`\int_gincr:N` 189, 355,
 503, 1621, 1666, 2003, 2275, 2344,
 2388, 2465, 3001, 3044, 3057, 3077
`\int_gset:Nn` 164, 183, 2528
`\int_gset_eq:NN` 172, 2389, 2466, 3058
`\int_if_exist:NnTF` 1993
`\int_if_odd:nTF` 2451
`\int_max:nn` 2108
`\int_new:N` 155, 156,
 402, 497, 500, 1647, 1954, 2271,
 2369, 2400, 2402, 2997, 3054, 3070
`\int_set:Nn` 522
`\int_set_eq:NN` ... 160, 179, 528, 2540
`\int_step_function:nnnN` 756
`\int_use:N` 357,
 388, 512, 523, 538, 683, 692, 840,
 868, 902, 976, 1008, 1624, 1630,
 1637, 1669, 1677, 1819, 1852, 1865,
 1923, 1983, 1996, 2008, 2010, 2101,
 2109, 2282, 2347, 2360, 2364, 2392,
 2399, 2470, 2571, 2825, 2835, 3008,
 3046, 3051, 3061, 3069, 3082, 3093
`\int_value:w`
 2693, 2728, 2885, 2896, 2914
`\int_zero:N` ... 1797, 1940, 2042, 2207
ior commands:
`\ior_close:N` 2152
`\ior_if_eof:NnTF` 2126
`\ior_map_break:` 2148
`\ior_open:Nn` 2125
`\ior_str_map_inline:Nn` 2133

K

kernel internal commands:

`__kernel_backend_align_begin:` . . .
 [72](#), [200](#), [224](#), [239](#)
`__kernel_backend_align_end:` . . .
 [72](#), [214](#), [232](#), [246](#)
`__kernel_backend_first_shipout:n`
 [50](#), [69](#), [507](#), [680](#)
`\g_kernel_backend_header_bool` . .
 [67](#), [678](#)
`__kernel_backend_literal:n`
 [46](#), [62](#), [65](#), [70](#), [74](#), [81](#),
 [84](#), [86](#), [142](#), [145](#), [147](#), [149](#), [153](#), [329](#),
 [342](#), [509](#), [534](#), [535](#), [543](#), [553](#), [609](#),
 [615](#), [700](#), [836](#), [1081](#), [1106](#), [1175](#),
 [1181](#), [1476](#), [1483](#), [1489](#), [1549](#), [1554](#),
 [1774](#), [1957](#), [1995](#), [2005](#), [2226](#), [2241](#),
 [2991](#), [3095](#), [3157](#), [3161](#), [3166](#), [3171](#)
`__kernel_backend_literal_page:n`
 [100](#), [144](#), [2985](#), [2987](#), [3176](#), [3178](#)
`__kernel_backend_literal_pdf:n` .
 [89](#), [141](#), [255](#), [312](#), [1366](#)
`__kernel_backend_literal_-`
 `postscript:n`
 [61](#), [75](#), [76](#), [80](#), [201](#), [202](#), [204](#),
 [205](#), [213](#), [225](#), [240](#), [1171](#), [2642](#), [2654](#)
`__kernel_backend_literal_svg:n` .
 [152](#), [159](#), [170](#), [178](#), [188](#),
 [356](#), [358](#), [375](#), [862](#), [1560](#), [1735](#), [1746](#)
`__kernel_backend_matrix:n`
 [128](#), [277](#), [298](#), [1466](#)
`__kernel_backend_postscript:n` . .
 [64](#),
 [611](#), [1085](#), [1087](#), [1089](#), [1093](#), [2265](#),
 [2321](#), [2336](#), [2378](#), [2384](#), [2424](#), [2456](#),
 [2463](#), [2467](#), [2481](#), [2509](#), [2553](#), [2560](#),
 [2566](#), [2574](#), [2581](#), [2615](#), [2622](#), [3230](#)
`__kernel_backend_scope:n` [157](#), [385](#),
 [390](#), [1147](#), [1567](#), [1612](#), [1614](#), [1634](#),
 [1674](#), [1696](#), [1708](#), [1710](#), [1712](#), [1714](#),
 [1716](#), [1718](#), [1720](#), [1722](#), [1725](#), [3325](#)
`__kernel_backend_scope_begin:` . .
 [83](#), [110](#), [146](#), [157](#),
 [199](#), [223](#), [238](#), [254](#), [271](#), [297](#), [311](#),
 [328](#), [341](#), [1372](#), [1544](#), [1562](#), [1566](#), [1733](#)
`__kernel_backend_scope_begin:n` .
 [157](#), [377](#), [405](#), [418](#)
`__kernel_backend_scope_end:` [83](#),
 [110](#), [146](#), [157](#), [216](#), [234](#), [248](#), [264](#),
 [291](#), [305](#), [321](#), [337](#), [349](#), [400](#), [414](#),
 [433](#), [532](#), [1373](#), [1556](#), [1563](#), [1569](#), [1747](#)
`\g_kernel_backend_scope_int` . . .
 [155](#), [162](#), [164](#), [169](#), [173](#), [181](#), [183](#), [189](#)

`\l__kernel_backend_scope_int` . . .
 [155](#), [161](#), [174](#), [180](#)
`\g_kernel_clip_path_int`
 [353](#), [1621](#), [1624](#), [1637](#), [1666](#), [1669](#), [1677](#)
`__kernel_color_backend_stack_-`
 `init:Nnn` [500](#), [562](#), [3254](#)
`__kernel_color_backend_stack_-`
 `pop:n` [541](#), [576](#), [633](#), [3284](#)
`__kernel_color_backend_stack_-`
 `push:nn` [541](#),
 [576](#), [630](#), [1048](#), [1061](#), [3273](#), [3310](#)
`__kernel_dependency_version_-`
 `check:Nn` [1](#)
`__kernel_dependency_version_-`
 `check:nn` [27](#), [29](#)
`__kernel_file_name_quote:n`
 [1883](#), [1909](#)
`__kernel_kern:n`
 [2383](#), [2385](#), [2614](#), [2618](#),
 [2621](#), [2625](#), [3127](#), [3135](#), [3138](#), [3154](#)

M

`\MessageBreak` [40](#)
mode commands:
`\mode_if_horizontal:TF` . . . [2530](#), [2537](#)
`\mode_if_math:TF` [2428](#)
msg commands:
`\msg_error:nnn` [638](#), [2127](#)
`\msg_new:nnn` [640](#)

O

`\oddsidemargin` [2452](#)
opacity internal commands:
`__opacity_backend:nn` [3318](#)
`__opacity_backend:nnn` [3204](#)
`__opacity_backend_fill:n`
 [3204](#), [3285](#), [3318](#)
`__opacity_backend_fill_stroke:nn`
 [3287](#), [3293](#), [3297](#), [3315](#)
`\l_opacity_backend_fill_tl`
 [3259](#), [3268](#), [3294](#), [3302](#)
`__opacity_backend_fillstroke:nn`
 [3285](#)
`__opacity_backend_reset:` [3261](#), [3312](#)
`__opacity_backend_select:n`
 [3204](#), [3261](#), [3318](#)
`__opacity_backend_select_aux:n` .
 [3204](#), [3261](#), [3300](#)
`\c_opacity_backend_stack_int` . . .
 [3250](#), [3273](#), [3284](#), [3310](#)
`__opacity_backend_stroke:n`
 [3204](#), [3285](#), [3318](#)
`\l_opacity_backend_stroke_tl` . . .
 [3259](#), [3269](#), [3289](#), [3303](#)

P

pdf commands:

`\pdf_object_if_exist:nTF` 916, 989, 1015
`\pdf_object_new:nn` ... 918, 991, 1017
`\pdf_object_ref:n` 931, 1002, 1025
`\pdf_object_ref_last:`
 .. 903, 910, 912, 965, 977, 1009, 1033
`\pdf_object_unnamed_write:nn` ...
 884, 909, 938, 960, 1001, 1024
`\pdf_object_write:nn` . 919, 992, 1018

pdf internal commands:

`__pdf_backend:n` 2990,
 2994, 2996, 3022, 3027, 3036, 3059,
 3078, 3091, 3098, 3130, 3131, 3141
`__pdf_backend_annotation:nnnn` ..
 2370, 2678, 3055
`__pdf_backend_annotation_`
 `aux:nnnn` 2372, 2375
`\g__pdf_backend_annotation_int` ..
 .. 2369, 2389, 2399, 3054, 3058, 3069
`__pdf_backend_annotation_last:` ..
 2398, 2691, 3068
`__pdf_backend_bdc:nn`
 2672, 2984, 3175, 3197
`__pdf_backend_catalog_gput:nn` ..
 2267, 2784, 2993, 3181
`__pdf_backend_compress_objects:n`
 2638, 2905, 3156, 3191
`__pdf_backend_compresslevel:n` ..
 2638, 2905, 3156, 3191
`\l__pdf_backend_content_box` 2367,
 2431, 2455, 2458, 2460, 2489, 2500
`__pdf_backend_destination:nn` ...
 2579, 2747, 3096
`__pdf_backend_destination:nnnn` ..
 2579, 2747, 3096
`__pdf_backend_destination_`
 `aux:nnnn` 2579, 3096
`__pdf_backend_emc:`
 2672, 2984, 3175, 3197
`__pdf_backend_info_gput:nn`
 2267, 2784, 2993, 3181
`__pdf_backend_link:nw` 2409
`__pdf_backend_link_aux:nw` ... 2409
`__pdf_backend_link_begin:n` .. 3071
`__pdf_backend_link_begin:nnnw` 2702
`__pdf_backend_link_begin:nw` ...
 2411, 2415, 2416
`__pdf_backend_link_begin_aux:nw`
 2419, 2421
`__pdf_backend_link_begin_`
 `goto:nnw` 2409, 2702, 3071

`__pdf_backend_link_begin_`
 `user:nnw` 2409, 2702, 3071
`\g__pdf_backend_link_bool`
 2404, 2418, 2423, 2438, 2476
`\g__pdf_backend_link_dict_tl` ...
 2401, 2426, 2471
`__pdf_backend_link_end:`
 2409, 2702, 3071
`__pdf_backend_link_end_aux:` . 2409
`\g__pdf_backend_link_int`
 2400, 2466,
 2470, 2571, 3070, 3077, 3082, 3093
`__pdf_backend_link_last:`
 2570, 2726, 3092
`__pdf_backend_link_margin:n` ...
 2572, 2737, 3094
`\g__pdf_backend_link_math_bool` ..
 2403, 2429, 2430, 2433, 2443
`__pdf_backend_link_minima:` .. 2409
`__pdf_backend_link_outerbox:n` 2409
`\g__pdf_backend_link_sf_int`
 2402, 2528, 2539, 2540
`__pdf_backend_link_sf_restore:` 2409
`__pdf_backend_link_sf_save:` . 2409
`\l__pdf_backend_model_box` . 2368,
 2448, 2480, 2488, 2499, 2514, 2516
`__pdf_backend_objcompresslevel:n`
 2905
`\g__pdf_backend_object_int`
 2271, 2275, 2278,
 2344, 2347, 2360, 2364, 2388, 2389,
 2392, 2465, 2466, 2997, 3001, 3004,
 3044, 3046, 3051, 3057, 3058, 3061
`__pdf_backend_object_last:`
 2363, 2883, 3050, 3183
`__pdf_backend_object_new:nn` ...
 2273, 2805, 2999, 3183
`__pdf_backend_object_now:nn` ...
 2342, 2857, 3042, 3183
`\g__pdf_backend_object_prop`
 2271, 2279, 2290, 2300,
 2804, 2822, 2838, 2997, 3005, 3012
`__pdf_backend_object_ref:n` 2273,
 2287, 2301, 2805, 2999, 3018, 3183
`__pdf_backend_object_write:nn` ..
 2283, 2826, 3009, 3183
`__pdf_backend_object_write:nnn` 3009
`__pdf_backend_object_write_`
 `array:nn` 2283, 3009
`__pdf_backend_object_write_`
 `dict:nn` 2283, 3009
`__pdf_backend_object_write_`
 `fstream:nn` 2283, 3009

_pdf_backend_object_write_- fstream:nnn	2317, 2319	pdf.globaldict	3333
_pdf_backend_object_write_- stream:nn	2283, 3009	pdf.leftboundary	3414
_pdf_backend_object_write_- stream:nnn	2283	pdf.link.dict	2409
_pdf_backend_object_write_- stream:nnnn	3009	pdf.linkdp.pad	2409, 3340
_pdf_backend_pageobject_ref:n	2365, 2894, 3052, 3183	pdf.linkht.pad	2409, 3340
_pdf_backend_pdfmark:n	2264, 2268, 2270, 2285, 2306, 2311, 2345, 2390, 2582, 2626, 2673, 2675	pdf.linkmargin	3340
_pdf_backend_version_major:...	2664, 2670, 2961, 3165, 3166, 3173, 3195	pdf.llx	2409, 3343
_pdf_backend_version_major_- gset:n	2662, 2933, 3163, 3193	pdf.lly	2409, 3343
_pdf_backend_version_minor:...	2668, 2670, 2961, 3170, 3171, 3173, 3195	pdf.originx	3414
_pdf_backend_version_minor_- gset:n	2662, 2933, 3163, 3193	pdf.originy	3414
\l_pdf_breaklink_pdfmark_tl	2405, 2473, 2565	pdf.outerbox	2409, 3656
_pdf_breaklink_postscript:n	2407, 2457, 2459, 2566	pdf.pdfmark	3656
_pdf_breaklink_usebox:N	2408, 2458, 2567	pdf.pdfmark.dict	3656
_pdf_exp_not_i:nn	2826, 2872, 2877	pdf.pdfmark.good	3656
_pdf_exp_not_ii:nn	2826, 2873, 2878	pdf.pt.dvi	3336
\l_pdf_internal_box	2262	pdf.rect	3343
pdf.baselineskip	2409, 3656	pdf.rect.ht	3336
pdf.bordertracking	3414	pdf.rightboundary	3414
pdf.bordertracking.begin	3414	pdf.save.linkll	3343
pdf.bordertracking.continue	3414	pdf.save.linkur	3343
pdf.bordertracking.end	3414	pdf.save.ll	3343
pdf.bordertracking.endpage	3414	pdf.save.ur	3343
pdf.breaklink	3552	pdf.tmpa	3379
pdf.breaklink.write	3552	pdf.tmpb	3379
pdf.brokenlink.dict	3414	pdf.tmpc	3379
pdf.brokenlink.rect	3414	pdf.tmpd	3379
pdf.brokenlink.skip	3414	pdf.urx	3343
pdf.count	3552	pdf.ury	2409, 3343
pdf.currentrect	3552	pdfmanagement commands:	
pdf.cvs	3336	_pdfmanagement_add:nnn	900, 974, 1003, 1007, 1026, 1030, 3256, 3270, 3304, 3307
pdf.dest.anchor	3379	_pdfmanagement_if_active_p:	895, 896, 969, 970, 3251, 3252, 3278, 3279
pdf.dest.point	3379	peek commands:	
pdf.dest.x	3379	_peek_meaning:Ntf	2168, 2171
pdf.dest.y	3379	_peek_remove_spaces:n	2166
pdf.dest2device	3379	prg commands:	
pdf.dev.x	3379	_prg_replicate:nn	168, 728, 749, 759, 944
pdf.dev.y	3379	prop commands:	
pdf.dvi.pt	3336	_prop_gput:Nnn	686, 911, 2279, 2822, 3005
		_prop_if_in:NnTF	663
		_prop_item:Nn	666, 2290, 2300, 2838, 3012
		_prop_new:N	644, 2272, 2804, 2998
		\ProvidesExplFile	2
		Q	
		quark commands:	
		_quark_if_recursion_tail_stop:n	662
		_q_recursion_stop	655
		_q_recursion_tail	654

S

scan commands:
 \scan_stop: 113, 122,
 595, 2196, 2199, 2720, 2745, 2768,
 2782, 2914, 2931, 2939, 2946, 2959
scan internal commands:
 \s__color_stop
 455, 458, 469, 472, 739, 740,
 744, 748, 761, 764, 768, 772, 786,
 945, 981, 985, 1110, 1112, 1133, 1135
 \s__graphics_stop
 1831, 1861, 2161, 2176,
 2183, 2187, 2189, 2191, 2246, 2254
separation 3330
seq commands:
 \seq_set_from_clist:Nn
 1766, 1790, 1929, 2117
skip commands:
 \skip_horizontal:n 217, 265, 322
str commands:
 \c_hash_str 388, 1630, 1637, 1677
 \c_percent_str 1153, 1154, 1155
 \str_case:nn 950, 2349, 2865
 \str_case:nnTF 2586, 2756, 3103
 \str_case:e:nn 2289, 2837
 \str_convert_pdfname:n 687, 707, 893
 \str_if_eq:nnTF
 474, 477, 480, 483, 636, 866, 3299
 \str_new:N 1872, 1873, 1874
 \str_tail:N 1886, 1912
sys commands:
 \sys_if_shell:TF 1870
 \sys_shell_now:n 1897

T

T_EX and L^AT_EX 2_ε commands:
 @cclv 2549, 2551, 2559
 @ifl@t@r 50, 52
 @makecol@hook 2543
 \current@color 14, 445, 450, 455, 469
 \special 2
tex commands:
 \tex_afterassignment:D 2195
 \tex_baselineskip:D 2520
 \tex_endinput:D 44
 \tex_global:D
 2907, 2924, 2938, 2945, 2952
 \tex_immediate:D
 1838, 1921, 2829, 2832, 2860, 2863
 \tex luatexversion:D 2936, 2964
 \tex_pdfannot:D 2684
 \tex_pdfcatalog:D 2790
 \tex_pdfcolorstack:D 582, 593
 \tex_pdfcolorstackinit:D 570

 \tex_pdfcompresslevel:D 2912
 \tex_pdfdest:D 2753, 2776
 \tex_pdfendlink:D 2723
 \tex_pdfextension:D
 92, 103, 113, 122, 131,
 579, 590, 2681, 2709, 2720, 2750,
 2773, 2787, 2797, 2808, 2829, 2860
 \tex_pdffeedback:D
 567, 2695, 2730, 2817, 2887, 2898
 \tex_pdfinfo:D 2800
 \tex_pdflastannot:D 2698
 \tex_pdflastlink:D 2733
 \tex_pdflastobj:D 2820, 2890
 \tex_pdflastximage:D 1833, 1857
 \tex_pdflastximagepages:D 1923
 \tex_pdflinkmargin:D 2743
 \tex_pdfliteral:D 95, 106
 \tex_pdfmajorversion:D
 2943, 2945, 2969, 2970
 \tex_pdfminorversion:D 2957, 2981
 \tex_pdfobj:D 2811, 2832, 2863
 \tex_pdfobjcompresslevel:D 2929
 \tex_pdfpageref:D 2901
 \tex_pdfrefximage:D 1857, 1864
 \tex_pdfrestore:D 125
 \tex_pdfsave:D 116
 \tex_pdfsetmatrix:D 134
 \tex_pdfstartlink:D 2712
 \tex_pdfvariable:D 2740,
 2909, 2926, 2938, 2954, 2965, 2978
 \tex_pdfximage:D 1838, 1921
 \tex_spacefactor:D 2531, 2540
 \tex_special:D 46
 \tex_the:D 1833, 2965, 2970, 2976
 \tex_vss:D 2616, 2623, 3133, 3152
 \tex_XeTeXpdffile:D 2053, 2099
 \tex_XeTeXpdfpagecount:D 2109
 \tex_XeTeXpicfile:D 2044
TeXcolorseparation 3330
\textwidth 2515
tl commands:
 \c_space_tl 279, 284,
 287, 513, 538, 649, 654, 692, 795,
 869, 1062, 1606, 1776, 1777, 1778,
 1779, 1959, 1960, 1961, 1962, 2010,
 2013, 2015, 2016, 2017, 2018, 2079,
 2101, 2228, 2229, 2230, 2231, 2471,
 2700, 2735, 2892, 2903, 3061, 3083
 \tl_clear:N 1798, 1806, 1813,
 1941, 1949, 2043, 2051, 2208, 2215
 \tl_gclear:N 1644, 1680
 \tl_gset:Nn 1603, 2426
 \tl_if_blank:nTF 514, 572, 647,
 743, 760, 767, 785, 888, 984, 2078, 2164

<code>\tl_if_empty:N</code>	NTF . 1606, 1801, 1845, 1853, 1980, 1984, 2011, 2026, 2066	
<code>\tl_if_empty:n</code>	TF 996, 1700	
<code>\tl_if_empty_p:N</code> 1841, 2023	
<code>\tl_if_head_is_space:N</code>	TF 445	
<code>\tl_new:N</code> 618, 619, 1610, 1794, 2401, 2405, 3259, 3260	
<code>\tl_put_right:Nn</code> 2547	
<code>\tl_set:Nn</code> 447, 459, 475, 478, 481, 485, 488, 628, 629, 1047, 1060, 1799, 1815, 1900, 2406, 2565, 3268, 3269, 3302, 3303	
<code>\tl_to_str:n</code> 2160, 2182, 2277, 2282, 2815, 2825, 2836, 3003, 3008	
<code>\tl_use:N</code> 827, 924	
token commands:		
<code>\c_math_toggle_token</code> 2434, 2444	
		U
	use commands:	
	<code>\use:N</code> 43, 2299, 2359, 3017, 3045
	<code>\use:n</code> 59, 450, 485, 505, 898, 972, 1005, 1028, 1114, 1124, 1137, 1312, 1436, 1501, 1513, 1525, 1685, 2073, 2157, 2179
	<code>\use_none:n</code> 1702, 2543
		V
	<code>\value</code> 2451
	vbox commands:	
	<code>\vbox_set:Nn</code> 2551
	<code>\vbox_to_zero:n</code>	2612, 2619, 3125, 3136
	<code>\vbox_unpack_drop:N</code> 2559