

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}{2023-10-23}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2023-10-23}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2023-10-23}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2023-10-23}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2023-10-23}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2023-10-23}{ }
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 {2023-10-10}
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.

```

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} } }

```

(End of 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,

```

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

```

(End of definition for __kernel_backend_first_shipout:n.)

1.1 dvips backend

59 `<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.

```

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

```

(End of 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 `"`).

```

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

```

(End of 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.

```

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

```

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

```

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

```

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

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

```

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

```

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

```

86 </dvips>

```

1.2 LuaTeX and pdfTeX backends

87 `<*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:e`

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).

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

(End of definition for `_kernel_backend_literal_pdf:n`.)

`_kernel_backend_literal_page:n`
`_kernel_backend_literal_page:e`

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

```
99 \cs_new_protected:Npn \_kernel_backend_literal_page:n #1
100 {
101   <*luatex>
102   \tex_pdfextension:D literal ~
103   </luatex>
104   <*pdftex>
105   \tex_pdfliteral:D
106   </pdftex>
107   page { \exp_not:n {#1} }
108 }
109 \cs_new_protected:Npn \_kernel_backend_literal_page:e #1
110 {
111   <*luatex>
112   \tex_pdfextension:D literal ~
113   </luatex>
114   <*pdftex>
115   \tex_pdfliteral:D
116   </pdftex>
117   page {#1}
118 }
```

(End of definition for `_kernel_backend_literal_page:n`.)

`_kernel_backend_scope_begin:`
`_kernel_backend_scope_end:`

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

```
119 \cs_new_protected:Npn \_kernel_backend_scope_begin:
120 {
121   <*luatex>
122   \tex_pdfextension:D save \scan_stop:
123   </luatex>
124   <*pdftex>
```

```

125     \tex_pdfsave:D
126   </pdftex>
127   }
128   \cs_new_protected:Npn \__kernel_backend_scope_end:
129     {
130   <*luatex>
131     \tex_pdfextension:D restore \scan_stop:
132   </luatex>
133   <*pdftex>
134     \tex_pdfrestore:D
135   </pdftex>
136   }

```

(End of 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.

```

137   \cs_new_protected:Npn \__kernel_backend_matrix:n #1
138     {
139   <*luatex>
140     \tex_pdfextension:D setmatrix
141   </luatex>
142   <*pdftex>
143     \tex_pdfsetmatrix:D
144   </pdftex>
145     { \exp_not:n {#1} }
146   }
147   \cs_generate_variant:Nn \__kernel_backend_matrix:n { e }

```

(End of definition for __kernel_backend_matrix:n.)

```

148   </luatex | pdftex>

```

1.3 dvipdfmx backend

```

149   <*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.

```

150   \cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1
151     { \__kernel_backend_literal:n { pdf:literal~ #1 } }
152   \cs_generate_variant:Nn \__kernel_backend_literal_pdf:n { e }

```

(End of 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!

```

153   \cs_new_protected:Npn \__kernel_backend_literal_page:n #1
154     { \__kernel_backend_literal:n { pdf:literal~direct~ #1 } }

```

(End of 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
`_kernel_backend_scope_end:` `xdvidfpmx (x:)` as these are well-tested “in the wild”.

```

155 \cs_new_protected:Npn \_kernel_backend_scope_begin:
156 { \_kernel_backend_literal:n { x:gsave } }
157 \cs_new_protected:Npn \_kernel_backend_scope_end:
158 { \_kernel_backend_literal:n { x:grestore } }

```

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

```

159 </dviPDFmx | xetex>

```

1.4 dvisvgm backend

```

160 <*dvisvgm>

```

`_kernel_backend_literal_svg:n` Unlike the other backends, the requirements for making SVG files mean that we can’t
`_kernel_backend_literal_svg:e` 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.

```

161 \cs_new_protected:Npn \_kernel_backend_literal_svg:n #1
162 { \_kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }
163 \cs_generate_variant:Nn \_kernel_backend_literal_svg:n { e }

```

(End of definition for `_kernel_backend_literal_svg:n.`)

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

```

164 \int_new:N \g__kernel_backend_scope_int
165 \int_new:N \l__kernel_backend_scope_int

```

(End of definition for `\g__kernel_backend_scope_int` and `\l__kernel_backend_scope_int.`)

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

```

\_kernel_backend_scope:n 166 \cs_new_protected:Npn \_kernel_backend_scope_begin:
\_kernel_backend_scope:e 167 {
168   \_kernel_backend_literal_svg:n { <g> }
169   \int_set_eq:NN
170     \l__kernel_backend_scope_int
171     \g__kernel_backend_scope_int
172   \group_begin:
173     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
174 }
175 \cs_new_protected:Npn \_kernel_backend_scope_end:
176 {
177   \prg_replicate:nn
178     { \g__kernel_backend_scope_int }
179     { \_kernel_backend_literal_svg:n { </g> } }
180   \group_end:
181   \int_gset_eq:NN
182     \g__kernel_backend_scope_int
183     \l__kernel_backend_scope_int
184 }

```

```

185 \cs_new_protected:Npn \__kernel_backend_scope_begin:n #1
186 {
187   \__kernel_backend_literal_svg:n { <g ~ #1 > }
188   \int_set_eq:NN
189     \l__kernel_backend_scope_int
190     \g__kernel_backend_scope_int
191   \group_begin:
192     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
193 }
194 \cs_generate_variant:Nn \__kernel_backend_scope_begin:n { e }
195 \cs_new_protected:Npn \__kernel_backend_scope:n #1
196 {
197   \__kernel_backend_literal_svg:n { <g ~ #1 > }
198   \int_gincr:N \g__kernel_backend_scope_int
199 }
200 \cs_generate_variant:Nn \__kernel_backend_scope:n { e }

(End of definition for \__kernel_backend_scope_begin: and others.)

201 </dvisvgm>
202 </package>

```

2 l3backend-box implementation

```

203 <*package>
204 <@@=box>

```

2.1 dvips backend

```

205 <*dvips>

```

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

```

206 \cs_new_protected:Npn \__box_backend_clip:N #1
207 {
208   \__kernel_backend_scope_begin:
209   \__kernel_backend_align_begin:
210   \__kernel_backend_literal_postscript:n { matrix~currentmatrix }
211   \__kernel_backend_literal_postscript:n
212     { Resolution~72~div~VResolution~72~div~scale }
213   \__kernel_backend_literal_postscript:n { DVImag~dup~scale }
214   \__kernel_backend_literal_postscript:e
215     {
216       0 ~
217       \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
218       \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
219       \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
220       rectclip
221     }
222   \__kernel_backend_literal_postscript:n { setmatrix }
223   \__kernel_backend_align_end:

```

```

224 \hbox_overlap_right:n { \box_use:N #1 }
225 \__kernel_backend_scope_end:
226 \skip_horizontal:n { \box_wd:N #1 }
227 }

```

(End of definition for __box_backend_clip:N.)

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

```

228 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
229 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
230 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
231 {
232   \__kernel_backend_scope_begin:
233   \__kernel_backend_align_begin:
234   \__kernel_backend_literal_postscript:e
235   {
236     \fp_compare:nNnTF {#2} = \c_zero_fp
237     { 0 }
238     { \fp_eval:n { round ( -(#2) , 5 ) } } ~
239     rotate
240   }
241   \__kernel_backend_align_end:
242   \box_use:N #1
243   \__kernel_backend_scope_end:
244 }

```

(End of 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.

```

245 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
246 {
247   \__kernel_backend_scope_begin:
248   \__kernel_backend_align_begin:
249   \__kernel_backend_literal_postscript:e
250   {
251     \fp_eval:n { round ( #2 , 5 ) } ~
252     \fp_eval:n { round ( #3 , 5 ) } ~
253     scale
254   }
255   \__kernel_backend_align_end:
256   \hbox_overlap_right:n { \box_use:N #1 }
257   \__kernel_backend_scope_end:
258 }

```

(End of definition for __box_backend_scale:Nnn.)

```

259 </dvips>

```


2.2 LuaTeX and pdfTeX backends

260 `<*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.

```

261 \cs_new_protected:Npn \_box_backend_clip:N #1
262 {
263   \_kernel_backend_scope_begin:
264   \_kernel_backend_literal_pdf:e
265   {
266     0~
267     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
268     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
269     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
270     re~W~n
271   }
272   \hbox_overlap_right:n { \box_use:N #1 }
273   \_kernel_backend_scope_end:
274   \skip_horizontal:n { \box_wd:N #1 }
275 }

```

(End of definition for `_box_backend_clip:N`.)

`_box_backend_rotate:Nn`

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

`_box_backend_rotate_aux:Nn`

`\l__box_backend_cos_fp`

`\l__box_backend_sin_fp`

```

276 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
277 { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
278 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
279 {
280   \_kernel_backend_scope_begin:
281   \box_set_wd:Nn #1 { Opt }
282   \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
283   \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
284     { \fp_zero:N \l__box_backend_cos_fp }
285   \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
286   \_kernel_backend_matrix:e
287   {
288     \fp_use:N \l__box_backend_cos_fp \c_space_tl
289     \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp
290       { 0~0 }
291       {
292         \fp_use:N \l__box_backend_sin_fp
293         \c_space_tl
294         \fp_eval:n { -\l__box_backend_sin_fp }
295       }
296   }

```

```

297     \fp_use:N \l__box_backend_cos_fp
298   }
299   \box_use:N #1
300   \__kernel_backend_scope_end:
301 }
302 \fp_new:N \l__box_backend_cos_fp
303 \fp_new:N \l__box_backend_sin_fp

```

(End of 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.

```

304 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
305 {
306   \__kernel_backend_scope_begin:
307   \__kernel_backend_matrix:e
308   {
309     \fp_eval:n { round ( #2 , 5 ) } ~
310     0~0~
311     \fp_eval:n { round ( #3 , 5 ) }
312   }
313   \hbox_overlap_right:n { \box_use:N #1 }
314   \__kernel_backend_scope_end:
315 }

```

(End of definition for `__box_backend_scale:Nnn`.)

```

316 </luatex | pdftex>

```

2.3 dvipdfmx/X_YTeX backend

```

317 <*dvipdfmx | xetex>

```

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

```

318 \cs_new_protected:Npn \__box_backend_clip:N #1
319 {
320   \__kernel_backend_scope_begin:
321   \__kernel_backend_literal_pdf:e
322   {
323     0~
324     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
325     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
326     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
327     re~W~n
328   }
329   \hbox_overlap_right:n { \box_use:N #1 }
330   \__kernel_backend_scope_end:
331   \skip_horizontal:n { \box_wd:N #1 }
332 }

```

(End of 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.

```

333 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
334 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
335 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
336 {
337   \__kernel_backend_scope_begin:
338   \__kernel_backend_literal:e
339   {
340     x:rotate~
341     \fp_compare:nNnTF {#2} = \c_zero_fp
342     { 0 }
343     { \fp_eval:n { round ( #2 , 5 ) } }
344   }
345   \box_use:N #1
346   \__kernel_backend_scope_end:
347 }

```

(End of 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.

```

348 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
349 {
350   \__kernel_backend_scope_begin:
351   \__kernel_backend_literal:e
352   {
353     x:scale~
354     \fp_eval:n { round ( #2 , 5 ) } ~
355     \fp_eval:n { round ( #3 , 5 ) }
356   }
357   \hbox_overlap_right:n { \box_use:N #1 }
358   \__kernel_backend_scope_end:
359 }

```

(End of definition for __box_backend_scale:Nnn.)

```

360 </dvipdfmx | xetex>

```

2.4 dvisvgm backend

```

361 <*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!

```

362 \cs_new_protected:Npn \__box_backend_clip:N #1
363 {
364   \int_gincr:N \g__kernel_clip_path_int
365   \__kernel_backend_literal_svg:e

```

```

366     { < clipPath-id = " l3cp \int_use:N \g__kernel_clip_path_int " > }
367     \__kernel_backend_literal_svg:e
368     {
369         <
370         path ~ d =
371         "
372             M ~ 0 ~
373             \dim_to_decimal:n { -\box_dp:N #1 } ~
374             L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
375             \dim_to_decimal:n { -\box_dp:N #1 } ~
376             L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
377             \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
378             L ~ 0 ~
379             \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
380             Z
381         "
382     />
383     }
384     \__kernel_backend_literal_svg:n
385     { < /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.

```

386     \__kernel_backend_scope_begin:n
387     {
388         transform =
389         "
390             translate ( { ?x } , { ?y } ) ~
391             scale ( 1 , -1 )
392         "
393     }
394     \__kernel_backend_scope:e
395     {
396         clip-path =
397         "url ( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int ) "
398     }
399     \__kernel_backend_scope:n
400     {
401         transform =
402         "
403             scale ( -1 , 1 ) ~
404             translate ( { ?x } , { ?y } ) ~
405             scale ( -1 , -1 )
406         "
407     }
408     \box_use:N #1
409     \__kernel_backend_scope_end:
410 }
411 \int_new:N \g__kernel_clip_path_int

```

(End of definition for $\text{_box_backend_clip:N}$ and $\text{_g_kernel_clip_path_int.}$)

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

```

412 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
413 {
414   \__kernel_backend_scope_begin:e
415   {
416     transform =
417     "
418       rotate
419       ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
420     "
421   }
422   \box_use:N #1
423   \__kernel_backend_scope_end:
424 }

```

(End of 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).

```

425 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
426 {
427   \__kernel_backend_scope_begin:e
428   {
429     transform =
430     "
431       translate ( { ?x } , { ?y } ) ~
432       scale
433       (
434         \fp_eval:n { round ( -#2 , 5 ) } ,
435         \fp_eval:n { round ( -#3 , 5 ) }
436       ) ~
437       translate ( { ?x } , { ?y } ) ~
438       scale ( -1 )
439     "
440   }
441   \hbox_overlap_right:n { \box_use:N #1 }
442   \__kernel_backend_scope_end:
443 }

```

(End of definition for `__box_backend_scale:Nnn`.)

```

444 \</dvisvgm>
445 \</package>

```

3 l3backend-color implementation

```

446 \*package>
447 \@@=color>

```

Color support is split into parts: collecting data from $\text{\LaTeX} 2_{\epsilon}$, 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/XYTeX` in particular. Whilst it is in some ways convenient to use the same approach in multiple backends, the fact that `dvipdfmx/XYTeX` is PDF-based means it (largely) sticks closer to direct PDF output.

3.1 The color stack

For PDF-based engines, we have a color stack available inside the specials. This is used for concepts beyond color itself: it is needed to manage the graphics state generally. Although `dvipdfmx/XYTeX` have multiple color stacks in recent releases, the way these interact with the original single stack and with other graphic state operations means that currently it is not feasible to use the multiple stacks.

3.1.1 Common code

```
448 <*luatex | pdftex>
```

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

```
449 \int_new:N \l__color_backend_stack_int
```

(End of definition for `\l__color_backend_stack_int`.)

```
450 </luatex | pdftex>
```

3.1.2 LuaTeX and pdfTeX

```
451 <*luatex | pdftex>
```

`_kernel_color_backend_stack_init:Nnn`

```
452 \cs_new_protected:Npn \_kernel_color_backend_stack_init:Nnn #1#2#3
453 {
454   \int_const:Nn #1
455   {
456     <*luatex>
457     \tex_pdffeedback:D colorstackinit ~
458     </luatex>
459     <*pdftex>
460     \tex_pdfcolorstackinit:D
461     </pdftex>
462     \tl_if_blank:nF {#2} { #2 ~ }
463     {#3}
464   }
465 }
```

(End of definition for `_kernel_color_backend_stack_init:Nnn`.)

`_kernel_color_backend_stack_push:nn`

`_kernel_color_backend_stack_pop:n`

```
466 \cs_new_protected:Npn \_kernel_color_backend_stack_push:nn #1#2
467 {
468   <*luatex>
469   \tex_pdfextension:D colorstack ~
470   </luatex>
471   <*pdftex>
472   \tex_pdfcolorstack:D
473   </pdftex>
474   \int_eval:n {#1} ~ push ~ {#2}
```

```

475 }
476 \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
477 {
478   \*luatex
479   \tex_pdfextension:D colorstack ~
480   \*pdfTeX
481   \tex_pdfcolorstack:D
482   \int_eval:n {#1} ~ pop \scan_stop:
483 }

```

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

```

486 \</luatex | pdfTeX>

```

3.2 General color

3.2.1 dvips-style

```

487 \*dvips | dvisvgm>

```

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

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_named:n
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc
488 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
489 { \__color_backend_select:n { cmyk ~ #1 } }
490 \cs_new_protected:Npn \__color_backend_select_gray:n #1
491 { \__color_backend_select:n { gray ~ #1 } }
492 \cs_new_protected:Npn \__color_backend_select_named:n #1
493 { \__color_backend_select:n { ~ #1 } }
494 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
495 { \__color_backend_select:n { rgb ~ #1 } }
496 \cs_new_protected:Npn \__color_backend_select:n #1
497 {
498   \__kernel_backend_literal:n { color~push~ #1 }
499   \*dvips
500   \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
501   \</dvips>
502 }
503 \cs_new_protected:Npn \__color_backend_reset:
504 { \__kernel_backend_literal:n { color~pop } }

```

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

```

505 \</dvips | dvisvgm>

```

3.2.2 LuaTeX and pdfTeX

```

506 \*luatex | pdfTeX>

```

```

\l__color_backend_fill_tl
\l__color_backend_stroke_tl
507 \tl_new:N \l__color_backend_fill_tl
508 \tl_new:N \l__color_backend_stroke_tl
509 \tl_set:Nn \l__color_backend_fill_tl { 0 ~ g }
510 \tl_set:Nn \l__color_backend_stroke_tl { 0 ~ G }

```

(End of 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_rgb:n
\__color_backend_select:nn
\__color_backend_reset:
Store the values then pass to the stack.
511 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
512 { \__color_backend_select:nn { #1 ~ k } { #1 ~ K } }
513 \cs_new_protected:Npn \__color_backend_select_gray:n #1
514 { \__color_backend_select:nn { #1 ~ g } { #1 ~ G } }
515 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
516 { \__color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
517 \cs_new_protected:Npn \__color_backend_select:nn #1#2
518 {
519   \tl_set:Nn \l__color_backend_fill_tl {#1}
520   \tl_set:Nn \l__color_backend_stroke_tl {#2}
521   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int { #1 ~ #2 }
522 }
523 \cs_new_protected:Npn \__color_backend_reset:
524 { \__kernel_color_backend_stack_pop:n \l__color_backend_stack_int }

```

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

525 `</luatex | pdftex>`

3.2.3 dvipdfmx/XqTeX

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

526 `<*dvipdfmx | xetex>`

```

\__color_backend_select:n
\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_reset:
Using the single stack is relatively easy as there is only one route.
527 \cs_new_protected:Npn \__color_backend_select:n #1
528 { \__kernel_backend_literal:n { pdf : bc ~ [ #1 ] } }
529 \cs_new_eq:NN \__color_backend_select_cmyk:n \__color_backend_select:n
530 \cs_new_eq:NN \__color_backend_select_gray:n \__color_backend_select:n
531 \cs_new_eq:NN \__color_backend_select_rgb:n \__color_backend_select:n
532 \cs_new_protected:Npn \__color_backend_reset:
533 { \__kernel_backend_literal:n { pdf : ec } }

```

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

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

```

534 \cs_new_protected:Npn \__color_backend_select_named:n #1
535 {
536   \str_if_eq:nnTF {#1} { Black }
537   { \__color_backend_select_gray:n { 0 } }
538   { \msg_error:nnn { color } { unknown-named-color } {#1} }
539 }
540 \msg_new:nnn { color } { unknown-named-color }
541 { Named-color~'#1'~is~not~known. }

```

(End of definition for `__color_backend_select_named:n`.)

542 `</dvipdfmx | xetex>`

3.3 Separations

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

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

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

```
\g_color_backend_colorant_prop
```

```
544 \prop_new:N \g_color_backend_colorant_prop
```

(End of definition for \g_color_backend_colorant_prop.)

```
\_color_backend_devicen_colorants:n
```

```
\_color_backend_devicen_colorants:w
```

```
545 \cs_new:Npe \_color_backend_devicen_colorants:n #1
```

```
546 {
```

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

```
548   {
```

```
549     \c_space_tl
```

```
550     << ~
```

```
551     /Colorants ~
```

```
552     << ~
```

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

```
554     \exp_not:N \q_recursion_tail \c_space_tl
```

```
555     \exp_not:N \q_recursion_stop
```

```
556     >> ~
```

```
557   >>
```

```
558 }
```

```
559 }
```

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

```
561 {
```

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

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

```
564   {
```

```
565     #1 ~
```

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

```
567   }
```

```
568   \_color_backend_devicen_colorants:w
```

```
569 }
```

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

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

```
571 < *dvips >
```

```
\_color_backend_select_separation:nn
```

```
\_color_backend_select_devicen:nn
```

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

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

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

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

```
\_color_backend_select_iccbased:nn
```

No support.

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

(End of definition for `_color_backend_select_iccbased:nn`.)

```
\_color_backend_separation_init:nnnn
\_color_backend_separation_init:neenn
\_color_backend_separation_init_aux:nnnnnn
\_color_backend_separation_init_DeviceCMYK:nnn
\_color_backend_separation_init_DeviceGray:nnn
\_color_backend_separation_init_DeviceRGB:nnn
\_color_backend_separation_init_Device:Nn
\_color_backend_separation_init:nnn
\_color_backend_separation_init_count:n
\_color_backend_separation_init_count:w
\_color_backend_separation_init:nnnn
\_color_backend_separation_init:w
\_color_backend_separation_init:n
\_color_backend_separation_init:nw
\_color_backend_separation_init_CIELAB:nnn
```

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

```
576 \cs_new_protected:Npe \_color_backend_separation_init:nnnnn #1#2#3#4#5
577 {
578   \bool_if:NT \g__kernel_backend_header_bool
579   {
580     \exp_not:N \exp_args:Ne \_kernel_backend_first_shipout:n
581     {
582       \exp_not:N \_color_backend_separation_init_aux:nnnnnn
583       { \exp_not:N \int_use:N \g__color_model_int }
584       {#1} {#2} {#3} {#4} {#5}
585     }
586     \prop_gput:Nee \exp_not:N \g__color_backend_colorant_prop
587     { / \exp_not:N \str_convert_pdfname:n {#1} }
588     {
589       << ~
590       /setcolorspace ~ {} ~
591       >> ~ begin ~
592       color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
593       end
594     }
595   }
596 }
597 \cs_generate_variant:Nn \_color_backend_separation_init:nnnnn { nee }
598 \cs_new_protected:Npn \_color_backend_separation_init_aux:nnnnnn #1#2#3#4#5#6
599 {
600   \_kernel_backend_literal:e
601   {
602     !
603     TeXDict ~ begin ~
604     /color #1
605     {
606       [ ~
607       /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
608       [ ~ #3 ~ ] ~
609       {
610         \cs_if_exist_use:cF { \_color_backend_separation_init_ #3 :nnn }
611         { \_color_backend_separation_init:nnn }
612         {#4} {#5} {#6}
613       }
614       ] ~ setcolorspace
615       } ~ def ~
616     end
617   }
618 }
619 \cs_new:cpn { \_color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
620 { \_color_backend_separation_init_Device:Nn 4 {#3} }
621 \cs_new:cpn { \_color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
622 { \_color_backend_separation_init_Device:Nn 1 {#3} }
623 \cs_new:cpn { \_color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
```

```

624 { \_color_backend_separation_init_Device:Nn 2 {#3} }
625 \cs_new:Npn \_color_backend_separation_init_Device:Nn #1#2
626 {
627   #2 ~
628   \prg_replicate:nn {#1}
629   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
630   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
631 }

```

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.

```

632 \cs_new:Npn \_color_backend_separation_init:nnn #1#2#3
633 {
634   \exp_args:Ne \_color_backend_separation_init:nnnn
635   { \_color_backend_separation_init_count:n {#2} }
636   {#1} {#2} {#3}
637 }
638 \cs_new:Npn \_color_backend_separation_init_count:n #1
639 { \int_eval:n { 0 \_color_backend_separation_init_count:w #1 ~ \s_color_stop } }
640 \cs_new:Npn \_color_backend_separation_init_count:w #1 ~ #2 \s_color_stop
641 {
642   +1
643   \tl_if_blank:nF {#2}
644   { \_color_backend_separation_init_count:w #2 \s_color_stop }
645 }

```

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

```

646 \cs_new:Npn \_color_backend_separation_init:nnnn #1#2#3#4
647 {
648   \_color_backend_separation_init:w #3 ~ \s_color_stop #4 ~ \s_color_stop
649   \prg_replicate:nn {#1}
650   {
651     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
652     \int_eval:n { 3 * #1 } ~ index ~ mul ~
653     2 ~ index ~ add ~
654     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
655   }
656   \int_step_function:nnnN {#1} { -1 } { 1 }
657   \_color_backend_separation_init:n
658   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
659   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
660   \tl_if_blank:nF {#2}

```

```

661     { \_color_backend_separation_init:nw {#1} #2 ~ \s_color_stop }
662   }
663   \cs_new:Npn \_color_backend_separation_init:w
664     #1 ~ #2 \s_color_stop #3 ~ #4 \s_color_stop
665   {
666     #1 ~ #3 ~ 0 ~
667     \tl_if_blank:nF {#2}
668     { \_color_backend_separation_init:w #2 \s_color_stop #4 \s_color_stop }
669   }
670   \cs_new:Npn \_color_backend_separation_init:n #1
671   { \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.

```

672   \cs_new:Npn \_color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s_color_stop
673   {
674     #2 ~ #3 ~
675     2 ~ index ~ 2 ~ index ~ lt ~
676     { ~ pop ~ excl ~ pop ~ } ~
677     { ~
678       2 ~ index ~ 1 ~ index ~ gt ~
679       { ~ excl ~ pop ~ excl ~ pop ~ } ~
680       { ~ pop ~ pop ~ } ~
681       ifelse ~
682     }
683     ifelse ~
684     #1 ~ 1 ~ roll ~
685     \tl_if_blank:nF {#4}
686     { \_color_backend_separation_init:nw {#1} #4 \s_color_stop }
687   }

```

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.

```

688   \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
689   {
690     \_color_backend_separation_init:neenn
691     {#2}
692     {
693       /CIEBasedABC ~
694       << ~
695       /RangeABC ~ [ ~ \c_color_model_range_CIELAB_tl \c_space_tl ] ~
696       /DecodeABC ~
697       [ ~
698         { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
699         { ~ 500 ~ div ~ } ~ bind ~
700         { ~ 200 ~ div ~ } ~ bind ~
701       ] ~
702       /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
703       /DecodeLMN ~
704       [ ~
705         { ~
706           dup ~ 6 ~ 29 ~ div ~ ge ~
707           { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
708           { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~

```

```

709         ifelse ~
710         0.9505 ~ mul ~
711     } ~ bind ~
712     { ~
713         dup ~ 6 ~ 29 ~ div ~ ge ~
714         { ~ dup ~ dup ~ mul ~ mul ~ } ~
715         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
716         ifelse ~
717     } ~ bind ~
718     { ~
719         dup ~ 6 ~ 29 ~ div ~ ge ~
720         { ~ dup ~ dup ~ mul ~ mul ~ } ~
721         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
722         ifelse ~
723         1.0890 ~ mul ~
724     } ~ bind
725 ] ~
726 /WhitePoint ~
727 [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
728 >>
729 }
730 { \c__color_model_range_CIELAB_tl }
731 { 100 ~ 0 ~ 0 }
732 {#3}
733 }

```

(End of definition for `__color_backend_separation_init:nnnn` and others.)

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

```

734 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
735 {
736     \__kernel_backend_literal:e
737     {
738         !
739         TeXDict ~ begin ~
740         /color \int_use:N \g__color_model_int
741         {
742             [ ~
743             /DeviceN ~
744             [ ~ #1 ~ ] ~
745             #2 ~
746             { ~ #3 ~ } ~
747             \__color_backend_devicen_colorants:n {#1}
748             ] ~ setcolorspace
749             } ~ def ~
750         end
751     }
752 }

```

(End of definition for `__color_backend_devicen_init:nnn`.)

`__color_backend_iccbased_init:nnn` No support at present.

```

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

```

(End of definition for `_color_backend_iccbased_init:nnn`.)

754 `</dvips>`

755 `<*dvisvgm>`

`_color_backend_select_separation:nn` No support at present.

`_color_backend_select_devicen:nn` 756 `\cs_new_protected:Npn _color_backend_select_separation:nn #1#2 { }`

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

(End of 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` 758 `\cs_new_protected:Npn _color_backend_separation_init:nnnnn #1#2#3#4#5 { }`

759 `\cs_new_protected:Npn _color_backend_separation_init_CIELAB:nnnnnn #1#2#3 { }`

(End of 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.

760 `\cs_new_protected:Npn _color_backend_select_iccbased:nn #1#2`

761 `{`

762 `_kernel_backend_literal_svg:e`

763 `{`

764 `<style>`

765 `@color-profile ~`

766 `\str_if_eq:nnTF {#2} { cmyk }`

767 `{ device-cmyk }`

768 `{ --color \int_use:N \g_color_model_int }`

769 `\c_space_tl`

770 `{`

771 `src:("#1")`

772 `}`

773 `</style>`

774 `}`

775 `}`

(End of definition for `_color_backend_select_iccbased:nn`.)

776 `</dvisvgm>`

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

`_color_backend_select_separation:nn`

`_color_backend_select_devicen:nn`

778 `<*dvipdfmx | xetex>`

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

780 `{ _kernel_backend_literal:e { pdf : bc ~ \pdf_object_ref:n {#1} ~ [#2] } }`

781 `</dvipdfmx | xetex>`

782 `<*luatex | pdftex>`

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

784 `{ _color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }`

785 `</luatex | pdftex>`

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

787 `\cs_new_eq:NN _color_backend_select_iccbased:nn _color_backend_select_separation:nn`

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

`_color_backend_init_resource:n` Resource initiation comes up a few times. For dvipdfmx/X_YTeX, we skip this as at present it's handled by the backend.

```

788 \cs_new_protected:Npn \_color_backend_init_resource:n #1
789 {
790   <*luatex | pdftex>
791     \bool_lazy_and:nnT
792       { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
793       { \pdfmanagement_if_active_p: }
794     {
795       \use:e
796       {
797         \pdfmanagement_add:nnn
798           { Page / Resources / ColorSpace }
799           { #1 }
800           { \pdf_object_ref_last: }
801       }
802     }
803   </luatex | pdftex>
804 }

```

(End of definition for `_color_backend_init_resource:n`.)

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

```

805 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5
806 {
807   \pdf_object_unnamed_write:ne { dict }
808   {
809     /FunctionType ~ 2
810     /Domain ~ [0 ~ 1]
811     \tl_if_blank:nF {#3} { /Range ~ [#3] }
812     /C0 ~ [#4] ~
813     /C1 ~ [#5] /N ~ 1
814   }
815   \exp_args:Ne \_color_backend_separation_init:nn
816     { \str_convert_pdfname:n {#1} } {#2}
817   \_color_backend_init_resource:n { color \int_use:N \g__color_model_int }
818 }
819 \cs_new_protected:Npn \_color_backend_separation_init:nn #1#2
820 {
821   \use:e
822   {
823     \pdf_object_new:n { color \int_use:N \g__color_model_int }
824     \pdf_object_write:nnn { color \int_use:N \g__color_model_int } { array }
825     { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
826   }
827   \prop_gput:Nne \g__color_backend_colorant_prop { /#1 }
828   { \pdf_object_ref_last: }
829 }

```

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

```

830 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
831 {
832   \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
833   {
834     \pdf_object_new:n { __color_illuminant_CIELAB_ #1 }
835     \pdf_object_write:nne { __color_illuminant_CIELAB_ #1 } { array }
836     {
837       /Lab ~
838       <<
839       /WhitePoint ~
840       [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
841       /Range ~ [ \c__color_model_range_CIELAB_tl ]
842       >>
843     }
844   }
845   \__color_backend_separation_init:nnnnn
846   {#2}
847   { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
848   { \c__color_model_range_CIELAB_tl }
849   { 100 ~ 0 ~ 0 }
850   {#3}
851 }

```

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

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

```

852 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
853 {
854   \pdf_object_unnamed_write:ne { stream }
855   {
856     {
857       /FunctionType ~ 4 ~
858       /Domain ~
859       [ ~
860         \prg_replicate:nn
861         { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
862         { 0 ~ 1 ~ }
863       ] ~
864       /Range ~
865       [ ~
866         \str_case:nn {#2}
867         {
868           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
869           { /DeviceGray } { 0 ~ 1 }
870           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
871         } ~
872       ]
873     }
874     { {#3} }
875   }
876   \use:e
877   {

```



```

878     \pdf_object_new:n { color \int_use:N \g__color_model_int }
879     \pdf_object_write:nnn { color \int_use:N \g__color_model_int } { array }
880     {
881         /DeviceN ~
882         [ ~ #1 ~ ] ~
883         #2 ~
884         \pdf_object_ref_last:
885         \__color_backend_devicen_colorants:n {#1}
886     }
887 }
888 \__color_backend_init_resource:n { color \int_use:N \g__color_model_int }
889 }
890 \cs_new:Npn \__color_backend_devicen_init:w #1 ~ #2 \s__color_stop
891 {
892     + 1
893     \tl_if_blank:nF {#2}
894     { \__color_backend_devicen_init:w #2 \s__color_stop }
895 }

```

(End of 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.

```

896 \cs_new_protected:Npn \__color_backend_iccbased_init:nnn #1#2#3
897 {
898     \pdf_object_if_exist:nF { __color_icc_ #1 }
899     {
900         \pdf_object_new:n { __color_icc_ #1 }
901         \pdf_object_write:nne { __color_icc_ #1 } { fstream }
902         {
903             {
904                 /N ~ \exp_not:n { #2 } ~
905                 \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
906             }
907             {#1}
908         }
909     }
910     \pdf_object_unnamed_write:ne { array }
911     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
912     \__color_backend_init_resource:n { color \int_use:N \g__color_model_int }
913 }

```

(End of definition for __color_backend_iccbased_init:nnn.)

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

```

914 \cs_new_protected:Npn \__color_backend_iccbased_device:nnn #1#2#3
915 {
916     \pdf_object_if_exist:nF { __color_icc_ #1 }
917     {
918         \pdf_object_new:n { __color_icc_ #1 }
919         \pdf_object_write:nnn { __color_icc_ #1 } { fstream }
920         {
921             { /N ~ #3 }
922             {#1}

```

```

923     }
924   }
925   \pdf_object_unnamed_write:ne { array }
926   { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
927   \__color_backend_init_resource:n { Default #2 }
928 }

```

(End of definition for __color_backend_iccbased_device:nnn.)

```

929 </dvipdfmx | luatex | pdftex | xetex>

```

3.4 Fill and stroke color

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

```

930 < *dvipdfmx | xetex>

```

```

\__color_backend_fill:n
\__color_backend_fill_cmyk:n
\__color_backend_fill_gray:n
\__color_backend_fill_rgb:n
\__color_backend_stroke:n
  \__color_backend_stroke_cmyk:n
  \__color_backend_stroke_gray:n
  \__color_backend_stroke_rgb:n
931 \cs_new_protected:Npn \__color_backend_fill:n #1
932 { \__kernel_backend_literal:n { pdf : bc ~ fill ~ [ #1 ] } }
933 \cs_new_eq:NN \__color_backend_fill_cmyk:n \__color_backend_fill:n
934 \cs_new_eq:NN \__color_backend_fill_gray:n \__color_backend_fill:n
935 \cs_new_eq:NN \__color_backend_fill_rgb:n \__color_backend_fill:n
936 \cs_new_protected:Npn \__color_backend_stroke:n #1
937 { \__kernel_backend_literal:n { pdf : bc ~ stroke ~ [ #1 ] } }
938 \cs_new_eq:NN \__color_backend_stroke_cmyk:n \__color_backend_stroke:n
939 \cs_new_eq:NN \__color_backend_stroke_gray:n \__color_backend_stroke:n
940 \cs_new_eq:NN \__color_backend_stroke_rgb:n \__color_backend_stroke:n

```

(End of definition for __color_backend_fill:n and others.)

```

\__color_backend_fill_separation:nn
\__color_backend_stroke_separation:nn
  \__color_backend_fill_devicen:nn
  \__color_backend_stroke_devicen:nn
941 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
942 {
943   \__kernel_backend_literal:e
944   { pdf : bc ~ fill ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
945 }
946 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
947 {
948   \__kernel_backend_literal:e
949   { pdf : bc ~ stroke ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
950 }
951 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
952 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

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

```

\__color_backend_fill_reset:
  \__color_backend_stroke_reset:
953 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
954 \cs_new_eq:NN \__color_backend_stroke_reset: \__color_backend_reset:

```

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

955 \langle /dvipdfmx | xetex \rangle

956 \langle *luatex | pdftex \rangle

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

```

957 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
958   { \_color_backend_fill:n { #1 ~ k } }
959 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
960   { \_color_backend_fill:n { #1 ~ g } }
961 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
962   { \_color_backend_fill:n { #1 ~ rg } }
963 \cs_new_protected:Npn \_color_backend_fill:n #1
964   {
965     \tl_set:Nn \l__color_backend_fill_tl {#1}
966     \_kernel_color_backend_stack_push:nn \l__color_backend_stack_int
967       { #1 ~ \l__color_backend_stroke_tl }
968   }
969 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
970   { \_color_backend_stroke:n { #1 ~ K } }
971 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
972   { \_color_backend_stroke:n { #1 ~ G } }
973 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
974   { \_color_backend_stroke:n { #1 ~ RG } }
975 \cs_new_protected:Npn \_color_backend_stroke:n #1
976   {
977     \tl_set:Nn \l__color_backend_stroke_tl {#1}
978     \_kernel_color_backend_stack_push:nn \l__color_backend_stack_int
979       { \l__color_backend_fill_tl \c_space_tl #1 }
980   }

```

(End of 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
981 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
982   { \_color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
983 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
984   { \_color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
985 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
986 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

```

\_color_backend_fill_reset:
  \_color_backend_stroke_reset:
987 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
988 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:

```

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

989 \langle /luatex | pdftex \rangle

990 \langle *dvips \rangle

```

\__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

```

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

```

991 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
992   { \__color_backend_fill:n { cmyk ~ #1 } }
993 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
994   { \__color_backend_fill:n { gray ~ #1 } }
995 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
996   { \__color_backend_fill:n { rgb ~ #1 } }
997 \cs_new_protected:Npn \__color_backend_fill:n #1
998   {
999     \__kernel_backend_literal:n { color~push~ #1 }
1000   }
1001 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1002   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1003 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1004   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1005 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1006   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }

```

(End of 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

```

```

1007 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
1008   { \__color_backend_fill:n { separation ~ #1 ~ #2 } }
1009 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1010   { \__kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1011 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1012 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

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

```

\__color_backend_fill_reset:
  \__color_backend_stroke_reset:

```

```

1013 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1014 \cs_new_protected:Npn \__color_backend_stroke_reset: { }

```

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

```

1015 </dvips>
1016 <*dvisvgm>

```

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

```

1017 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1018   { \__color_backend_fill:n { cmyk ~ #1 } }
1019 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1020   { \__color_backend_fill:n { gray ~ #1 } }
1021 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1022   { \__color_backend_fill:n { rgb ~ #1 } }
1023 \cs_new_protected:Npn \__color_backend_fill:n #1
1024   {
1025     \__kernel_backend_literal:n { color~push~ #1 }
1026   }

```

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

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).

```

1027 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1028 { \__color_backend_cmyk:w #1 \s__color_stop }
1029 \cs_new_protected:Npn \__color_backend_stroke_cmyk:w
1030 #1 ~ #2 ~ #3 ~ #4 \s__color_stop
1031 {
1032   \use:e
1033   {
1034     \__color_backend:nnn
1035     { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1036     { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1037     { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }
1038   }
1039 }
1040 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1041 {
1042   \use:e
1043   {
1044     \__color_backend_stroke_gray_aux:n
1045     { \fp_eval:n { 100 * (#1) } }
1046   }
1047 }
1048 \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1049 { \__color_backend:nnn {#1} {#1} {#1} }
1050 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1051 { \__color_backend_rgb:w #1 \s__color_stop }
1052 \cs_new_protected:Npn \__color_backend_stroke_rgb:w
1053 #1 ~ #2 ~ #3 \s__color_stop
1054 {
1055   \use:e
1056   {
1057     \__color_backend:nnn
1058     { \fp_eval:n { 100 * (#1) } }
1059     { \fp_eval:n { 100 * (#2) } }
1060     { \fp_eval:n { 100 * (#3) } }
1061   }
1062 }
1063 \cs_new_protected:Npe \__color_backend:nnn #1#2#3
1064 {
1065   \__kernel_backend_scope:n
1066   {
1067     stroke =
1068     "
1069     rgb
1070     (
1071       #1 \c_percent_str ,
1072       #2 \c_percent_str ,
1073       #3 \c_percent_str
1074     )
1075     "
1076   }
1077 }

```

(End of definition for `_color_backend_stroke_cmyk:n` and others.)

At present, these are no-ops.

```

1078 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1079 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2 { }
1080 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1081 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

`_color_backend_fill_reset:`

`_color_backend_stroke_reset:`

```

1082 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
1083 \cs_new_protected:Npn \_color_backend_stroke_reset: { }

```

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

`_color_backend_devicen_init:nnn`

`_color_backend_iccbased_init:nnn`

No support at present.

```

1084 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3 { }
1085 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3 { }

```

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

1086 `</divisvgm>`

1087 `</package>`

3.5 Font handling integration

In LuaT_EX these colors should also be usable to color fonts, so luaotfload color handling is extended to include these.

```

1088 <*lua>

1089 local l = lpeg
1090 local spaces = l.P' '^0
1091 local digit16 = l.R('09', 'af', 'AF')
1092
1093 local octet = digit16 * digit16 / function(s)
1094   return string.format('%.3g ', tonumber(s, 16) / 255)
1095 end
1096
1097 if luaotfload and luaotfload.set_transparent_colorstack then
1098   local htmlcolor = l.Cs(octet * octet * octet * -1 * l.Cc'rg')
1099   local color_export = {
1100     token.create'tex_endlocalcontrol:D',
1101     token.create'tex_hpack:D',
1102     token.new(0, 1),
1103     token.create'color_export:nnN',
1104     token.new(0, 1),
1105     '',
1106     token.new(0, 2),
1107     token.new(0, 1),
1108     'backend',
1109     token.new(0, 2),
1110     token.create'l_tmpa_tl',
1111     token.create'exp_after:wN',
1112     token.create'__color_select:nn',

```

```

1113     token.create'l_tmpa_tl',
1114     token.new(0, 2),
1115 }
1116 local group_end = token.create'group_end:'
1117 local value = (1 - l.P'}')^0
1118 luatexbase.add_to_callback('luaotfload.parse_color', function (value)
1119 % Also allow HTML colors to preserve compatibility
1120     local html = htmlcolor:match(value)
1121     if html then return html end
1122
1123     tex.runtoks(function()
1124         token.get_next()
1125         color_export[6] = value
1126         tex.sprint(-2, color_export)
1127     end)
1128     local list = token.scan_list()
1129     if not list.head or list.head.next
1130         or list.head.subtype ~= node.subtype'pdf_colorstack' then
1131         error'Unexpected backend behavior'
1132     end
1133     local cmd = list.head.data
1134     node.free(list)
1135     return cmd
1136 end, 'l3color')
1137 end
1138 </lua>
1139 <*luatex>
1140 <*package>
1141 \lua_load_module:n {l3backend-luatex}
1142 </package>
1143 </luatex>

```

4 l3backend-draw implementation

```

1144 <*package>
1145 <@@=draw>

```

4.1 dvips backend

```

1146 <*dvips>

```

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

`__draw_backend_literal:e` 1147 \cs_new_eq:NN __draw_backend_literal:n __kernel_backend_literal_postscript:n
1148 \cs_generate_variant:Nn __draw_backend_literal:n { e }

(End of 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.

```

1149 \cs_new_protected:Npn \__draw_backend_begin:
1150 {
1151   \__kernel_backend_literal:n { ps::[begin] }
1152   \__draw_backend_literal:n { @beginspecial }
1153 }
1154 \cs_new_protected:Npn \__draw_backend_end:
1155 {
1156   \__draw_backend_literal:n { @endspecial }
1157   \__kernel_backend_literal:n { ps::[end] }
1158 }

```

(End of definition for `__draw_backend_begin:` and `__draw_backend_end:.`)

`__draw_backend_scope_begin:` Scope here may need to contain saved definitions, so the entire memory rather than just the graphic state has to be sent to the stack.

```

1159 \cs_new_protected:Npn \__draw_backend_scope_begin:
1160 { \__draw_backend_literal:n { save } }
1161 \cs_new_protected:Npn \__draw_backend_scope_end:
1162 { \__draw_backend_literal:n { restore } }

```

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

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

```

1163 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1164 {
1165   \__draw_backend_literal:e
1166   {
1167     \dim_to_decimal_in_bp:n {#1} ~
1168     \dim_to_decimal_in_bp:n {#2} ~ moveto
1169   }
1170 }
1171 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1172 {
1173   \__draw_backend_literal:e
1174   {
1175     \dim_to_decimal_in_bp:n {#1} ~
1176     \dim_to_decimal_in_bp:n {#2} ~ lineto
1177   }
1178 }
1179 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1180 {
1181   \__draw_backend_literal:e
1182   {
1183     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1184     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1185     moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1186   }

```



```

1187 }
1188 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1189 {
1190   \__draw_backend_literal:e
1191   {
1192     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1193     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1194     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1195     curveto
1196   }
1197 }

```

(End of 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
1198 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1199 { \bool_gset_true:N \g__draw_draw_eor_bool }
1200 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1201 { \bool_gset_false:N \g__draw_draw_eor_bool }
1202 \bool_new:N \g__draw_draw_eor_bool

```

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

```

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

```

```

1203 \cs_new_protected:Npn \__draw_backend_closepath:
1204 { \__draw_backend_literal:n { closepath } }
1205 \cs_new_protected:Npn \__draw_backend_stroke:
1206 {
1207   \__draw_backend_literal:n { gsave }
1208   \__draw_backend_literal:n { color.sc }
1209   \__draw_backend_literal:n { stroke }
1210   \__draw_backend_literal:n { grestore }
1211   \bool_if:NT \g__draw_draw_clip_bool
1212   {
1213     \__draw_backend_literal:e
1214     {
1215       \bool_if:NT \g__draw_draw_eor_bool { eo }
1216       clip
1217     }
1218   }
1219   \__draw_backend_literal:n { newpath }
1220   \bool_gset_false:N \g__draw_draw_clip_bool
1221 }
1222 \cs_new_protected:Npn \__draw_backend_closestroke:
1223 {
1224   \__draw_backend_closepath:
1225   \__draw_backend_stroke:
1226 }

```

```

1227 \cs_new_protected:Npn \__draw_backend_fill:
1228 {
1229   \__draw_backend_literal:e
1230   {
1231     \bool_if:NT \g__draw_draw_eor_bool { eo }
1232     fill
1233   }
1234   \bool_if:NT \g__draw_draw_clip_bool
1235   {
1236     \__draw_backend_literal:e
1237     {
1238       \bool_if:NT \g__draw_draw_eor_bool { eo }
1239       clip
1240     }
1241   }
1242   \__draw_backend_literal:n { newpath }
1243   \bool_gset_false:N \g__draw_draw_clip_bool
1244 }
1245 \cs_new_protected:Npn \__draw_backend_fillstroke:
1246 {
1247   \__draw_backend_literal:e
1248   {
1249     \bool_if:NT \g__draw_draw_eor_bool { eo }
1250     fill
1251   }
1252   \__draw_backend_literal:n { gsave }
1253   \__draw_backend_literal:n { color.sc }
1254   \__draw_backend_literal:n { stroke }
1255   \__draw_backend_literal:n { grestore }
1256   \bool_if:NT \g__draw_draw_clip_bool
1257   {
1258     \__draw_backend_literal:e
1259     {
1260       \bool_if:NT \g__draw_draw_eor_bool { eo }
1261       clip
1262     }
1263   }
1264   \__draw_backend_literal:n { newpath }
1265   \bool_gset_false:N \g__draw_draw_clip_bool
1266 }
1267 \cs_new_protected:Npn \__draw_backend_clip:
1268 { \bool_gset_true:N \g__draw_draw_clip_bool }
1269 \bool_new:N \g__draw_draw_clip_bool
1270 \cs_new_protected:Npn \__draw_backend_discardpath:
1271 {
1272   \bool_if:NT \g__draw_draw_clip_bool
1273   {
1274     \__draw_backend_literal:e
1275     {
1276       \bool_if:NT \g__draw_draw_eor_bool { eo }
1277       clip
1278     }
1279   }
1280   \__draw_backend_literal:n { newpath }

```

```

1281 \bool_gset_false:N \g__draw_draw_clip_bool
1282 }

```

(End of definition for `__draw_backend_closepath:` and others.)

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

```

1283 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1284 {
1285   \__draw_backend_literal:e
1286   {
1287     [
1288       \exp_args:Nf \use:n
1289       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1290     ] ~
1291     \dim_to_decimal_in_bp:n {#2} ~ setdash
1292   }
1293 }
1294 \cs_new:Npn \__draw_backend_dash:n #1
1295 { ~ \dim_to_decimal_in_bp:n {#1} }
1296 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1297 {
1298   \__draw_backend_literal:e
1299   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1300 }
1301 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1302 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1303 \cs_new_protected:Npn \__draw_backend_cap_but:
1304 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1305 \cs_new_protected:Npn \__draw_backend_cap_round:
1306 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1307 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1308 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1309 \cs_new_protected:Npn \__draw_backend_join_miter:
1310 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1311 \cs_new_protected:Npn \__draw_backend_join_round:
1312 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1313 \cs_new_protected:Npn \__draw_backend_join_bevel:
1314 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

(End of 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.

```

1315 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1316 {
1317   \__draw_backend_literal:n
1318   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1319 }

```

(End of 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).

```

1320 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
1321 {
1322   \_draw_backend_literal:n { @endspecial }
1323   \_draw_backend_literal:n { [end] }
1324   \_draw_backend_literal:n { [begin] }
1325   \_draw_backend_literal:n { save }
1326   \_draw_backend_literal:n { currentpoint }
1327   \_draw_backend_literal:n { currentpoint~translate }
1328   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1329   \_draw_backend_cm:nnnn { #2 } { #3 } { #4 } { #5 }
1330   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1331   \_draw_backend_literal:n { neg~exch~neg~exch~translate }
1332   \_draw_backend_literal:n { [end] }
1333   \hbox_overlap_right:n { \box_use:N #1 }
1334   \_draw_backend_literal:n { [begin] }
1335   \_draw_backend_literal:n { restore }
1336   \_draw_backend_literal:n { [end] }
1337   \_draw_backend_literal:n { [begin] }
1338   \_draw_backend_literal:n { @beginspecial }
1339 }

```

(End of definition for `_draw_backend_box_use:Nnnnn`.)

1340 `</dvips>`

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

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

1341 `<*dvipdfmx | luatex | pdftex | xetex>`

4.2.1 Drawing

`_draw_backend_literal:n`
`_draw_backend_literal:e`

Pass data through using a dedicated interface.

```

1342 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_pdf:n
1343 \cs_generate_variant:Nn \_draw_backend_literal:n { e }

```

(End of definition for `_draw_backend_literal:n`.)

`_draw_backend_begin:`
`_draw_backend_end:`

No special requirements here, so simply set up a drawing scope.

```

1344 \cs_new_protected:Npn \_draw_backend_begin:
1345 { \_draw_backend_scope_begin: }

```

```

1346 \cs_new_protected:Npn \__draw_backend_end:
1347 { \__draw_backend_scope_end: }

```

(End of definition for __draw_backend_begin: and __draw_backend_end:.)

__draw_backend_scope_begin: Use the backend-level scope mechanisms.

```

\__draw_backend_scope_end: 1348 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1349 \cs_new_eq:NN \__draw_backend_scope_end: \__kernel_backend_scope_end:

```

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

__draw_backend_moveto:nn Path creation operations all resolve directly to PDF primitive steps, with only the need to convert to bp.

```

\__draw_backend_lineto:nn 1350 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
\__draw_backend_curveto:nnnnnn 1351 {
\__draw_backend_rectangle:nnnn 1352 \__draw_backend_literal:e
1353 { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1354 }
1355 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1356 {
1357 \__draw_backend_literal:e
1358 { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1359 }
1360 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1361 {
1362 \__draw_backend_literal:e
1363 {
1364 \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1365 \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1366 \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1367 c
1368 }
1369 }
1370 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1371 {
1372 \__draw_backend_literal:e
1373 {
1374 \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1375 \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1376 re
1377 }
1378 }

```

(End of 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: 1379 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
\g__draw_draw_eor_bool 1380 { \bool_gset_true:N \g__draw_draw_eor_bool }
1381 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1382 { \bool_gset_false:N \g__draw_draw_eor_bool }
1383 \bool_new:N \g__draw_draw_eor_bool

```

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

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

```

\__draw_backend_closepath:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
1384 \cs_new_protected:Npn \__draw_backend_closepath:
1385 { \__draw_backend_literal:n { h } }
1386 \cs_new_protected:Npn \__draw_backend_stroke:
1387 { \__draw_backend_literal:n { S } }
1388 \cs_new_protected:Npn \__draw_backend_closestroke:
1389 { \__draw_backend_literal:n { s } }
1390 \cs_new_protected:Npn \__draw_backend_fill:
1391 {
1392   \__draw_backend_literal:e
1393   { f \bool_if:NT \g__draw_draw_eor_bool * }
1394 }
1395 \cs_new_protected:Npn \__draw_backend_fillstroke:
1396 {
1397   \__draw_backend_literal:e
1398   { B \bool_if:NT \g__draw_draw_eor_bool * }
1399 }
1400 \cs_new_protected:Npn \__draw_backend_clip:
1401 {
1402   \__draw_backend_literal:e
1403   { W \bool_if:NT \g__draw_draw_eor_bool * }
1404 }
1405 \cs_new_protected:Npn \__draw_backend_discardpath:
1406 { \__draw_backend_literal:n { n } }

```

(End of definition for __draw_backend_closepath: and others.)

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1407 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1408 {
1409   \__draw_backend_literal:e
1410   {
1411     [
1412       \exp_args:Nf \use:n
1413       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1414     ] ~
1415     \dim_to_decimal_in_bp:n {#2} ~ d
1416   }
1417 }
1418 \cs_new:Npn \__draw_backend_dash:n #1
1419 { ~ \dim_to_decimal_in_bp:n {#1} }
1420 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1421 {
1422   \__draw_backend_literal:e
1423   { \dim_to_decimal_in_bp:n {#1} ~ w }
1424 }
1425 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1426 { \__draw_backend_literal:e { #1 ~ M } }
1427 \cs_new_protected:Npn \__draw_backend_cap_butt:
1428 { \__draw_backend_literal:n { 0 ~ J } }
1429 \cs_new_protected:Npn \__draw_backend_cap_round:
1430 { \__draw_backend_literal:n { 1 ~ J } }
1431 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1432 { \__draw_backend_literal:n { 2 ~ J } }

```

```

1433 \cs_new_protected:Npn \__draw_backend_join_miter:
1434 { \__draw_backend_literal:n { 0 ~ j } }
1435 \cs_new_protected:Npn \__draw_backend_join_round:
1436 { \__draw_backend_literal:n { 1 ~ j } }
1437 \cs_new_protected:Npn \__draw_backend_join_bevel:
1438 { \__draw_backend_literal:n { 2 ~ j } }

```

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

__draw_backend_cm:nnnn
__draw_backend_cm_aux:nnnn

Another split here between LuaTeX/pdfTeX and dvipdfmx/X_YTeX. In the former, we have a direct method to maintain alignment: the backend can use a matrix itself. For dvipdfmx/X_YTeX, we can to decompose the matrix into rotations and a scaling, then use those operations as they are handled by the backend. (There is backend support for matrix operations in dvipdfmx/X_YTeX, but as a matched pair so not suitable for the “stand alone” transformation set up here.) The specials used here are from xdvipdfmx originally: they are well-tested, but probably equivalent to the pdf: versions!

```

1439 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1440 {
1441   <*luatex | pdftex>
1442     \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1443   </luatex | pdftex>
1444   <*dvipdfmx | xetex>
1445     \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1446     \__draw_backend_cm_aux:nnnn
1447   </dvipdfmx | xetex>
1448 }
1449 <*dvipdfmx | xetex>
1450 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1451 {
1452   \__kernel_backend_literal:e
1453   {
1454     x:rotate~
1455     \fp_compare:nNnTF {#1} = \c_zero_fp
1456       { 0 }
1457       { \fp_eval:n { round ( -#1 , 5 ) } }
1458   }
1459   \__kernel_backend_literal:e
1460   {
1461     x:scale~
1462     \fp_eval:n { round ( #2 , 5 ) } ~
1463     \fp_eval:n { round ( #3 , 5 ) }
1464   }
1465   \__kernel_backend_literal:e
1466   {
1467     x:rotate~
1468     \fp_compare:nNnTF {#4} = \c_zero_fp
1469       { 0 }
1470       { \fp_eval:n { round ( -#4 , 5 ) } }
1471   }
1472 }
1473 </dvipdfmx | xetex>

```

(End of 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.

```

1474 <*dvipdfmx | xetex>
1475 \cs_new_protected:Npn \_draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1476 {
1477   \use:e
1478   {
1479     \_draw_backend_cm_decompose_auxi:nnnnN
1480     { \fp_eval:n { (#1 + #4) / 2 } }
1481     { \fp_eval:n { (#1 - #4) / 2 } }
1482     { \fp_eval:n { (#3 + #2) / 2 } }
1483     { \fp_eval:n { (#3 - #2) / 2 } }
1484   }
1485   #5
1486 }
1487 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1488 {
1489   \use:e
1490   {
1491     \_draw_backend_cm_decompose_auxii:nnnnN
1492     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1493     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1494     { \fp_eval:n { atand ( #3 , #2 ) } }
1495     { \fp_eval:n { atand ( #4 , #1 ) } }
1496   }
1497   #5
1498 }
1499 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5

```



```

1500 {
1501   \use:e
1502   {
1503     \__draw_backend_cm_decompose_auxiii:nnnnN
1504     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1505     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1506     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1507     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1508   }
1509   #5
1510 }
1511 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1512 {
1513   \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1514   { #5 {#1} {#2} {#3} {#4} }
1515   { #5 {#1} {#3} {#2} {#4} }
1516 }
1517 </dviptfm | xetex>

```

(End of 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.

```

1518 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1519 {
1520   \__kernel_backend_scope_begin:
1521   < *luatex | pdftex >
1522   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1523   < /luatex | pdftex >
1524   < *dviptfm | xetex >
1525   \__kernel_backend_literal:n
1526   { pdf:btrans-matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1527   < /dviptfm | xetex >
1528   \hbox_overlap_right:n { \box_use:N #1 }
1529   < *dviptfm | xetex >
1530   \__kernel_backend_literal:n { pdf:etrans }
1531   < /dviptfm | xetex >
1532   \__kernel_backend_scope_end:
1533 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1534 < /dviptfm | luatex | pdftex | xetex >

```

4.3 dvisvgm backend

```

1535 < *dvisvgm >

```

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

```

\__draw_backend_literal:e
1536 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1537 \cs_generate_variant:Nn \__draw_backend_literal:n { e }

```

(End of definition for `_draw_backend_literal:n`.)

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

`_draw_backend_scope_end:` 1538 `\cs_new_eq:NN _draw_backend_scope_begin: _kernel_backend_scope_begin:`
1539 `\cs_new_eq:NN _draw_backend_scope_end: _kernel_backend_scope_end:`

(End of 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

1540 `\cs_new_protected:Npn _draw_backend_begin:`
1541 `{`
1542 `_kernel_backend_scope_begin:`
1543 `_kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }`
1544 `}`
1545 `\cs_new_eq:NN _draw_backend_end: _kernel_backend_scope_end:`

(End of 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`

1546 `\cs_new_protected:Npn _draw_backend_moveto:nn #1#2`
1547 `{`
1548 `_draw_backend_add_to_path:n`
1549 `{ M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }`
1550 `}`
1551 `\cs_new_protected:Npn _draw_backend_lineto:nn #1#2`
1552 `{`
1553 `_draw_backend_add_to_path:n`
1554 `{ L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }`
1555 `}`
1556 `\cs_new_protected:Npn _draw_backend_rectangle:nnnn #1#2#3#4`
1557 `{`
1558 `_draw_backend_add_to_path:n`
1559 `{`
1560 `M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}`
1561 `h ~ \dim_to_decimal:n {#3} ~`
1562 `v ~ \dim_to_decimal:n {#4} ~`
1563 `h ~ \dim_to_decimal:n { -#3 } ~`
1564 `Z`
1565 `}`
1566 `}`
1567 `\cs_new_protected:Npn _draw_backend_curveto:nnnnnn #1#2#3#4#5#6`
1568 `{`
1569 `_draw_backend_add_to_path:n`
1570 `{`
1571 `C ~`
1572 `\dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~`
1573 `\dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~`
1574 `\dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}`
1575 `}`

```

1576 }
1577 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1578 {
1579   \tl_gset:Nx \g__draw_backend_path_tl
1580   {
1581     \g__draw_backend_path_tl
1582     \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1583     #1
1584   }
1585 }
1586 \tl_new:N \g__draw_backend_path_tl

```

(End of 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:
1587 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1588 { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1589 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1590 { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

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

```

\__draw_backend_path:n Setting fill and stroke effects and doing clipping all has to be done using scopes. This
\__draw_backend_closepath: means setting up the various requirements in a shared auxiliary which deals with the
\__draw_backend_stroke: bits and pieces. Clipping paths are reused for path drawing: not essential but avoids
\__draw_backend_closestroke: constructing them twice. Discarding a path needs a separate function as it's not quite
\__draw_backend_fill: the same.
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int
1591 \cs_new_protected:Npn \__draw_backend_closepath:
1592 { \__draw_backend_add_to_path:n { Z } }
1593 \cs_new_protected:Npn \__draw_backend_path:n #1
1594 {
1595   \bool_if:NTF \g__draw_draw_clip_bool
1596   {
1597     \int_gincr:N \g__kernel_clip_path_int
1598     \__draw_backend_literal:e
1599     {
1600       < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1601       { ?nl }
1602       <path~d=" \g__draw_backend_path_tl "> { ?nl }
1603       < /clipPath > { ? nl }
1604       <
1605       use~xlink:href =
1606       "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1607       #1
1608       />
1609     }
1610     \__kernel_backend_scope:e
1611     {
1612       clip-path =
1613       "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int )"
1614     }
1615   }
1616   {
1617     \__draw_backend_literal:e

```

```

1618         { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1619     }
1620     \tl_gclear:N \g__draw_backend_path_tl
1621     \bool_gset_false:N \g__draw_draw_clip_bool
1622 }
1623 \int_new:N \g__draw_backend_path_int
1624 \cs_new_protected:Npn \__draw_backend_stroke:
1625 { \__draw_backend_path:n { style="fill:none" } }
1626 \cs_new_protected:Npn \__draw_backend_closestroke:
1627 {
1628     \__draw_backend_closepath:
1629     \__draw_backend_stroke:
1630 }
1631 \cs_new_protected:Npn \__draw_backend_fill:
1632 { \__draw_backend_path:n { style="stroke:none" } }
1633 \cs_new_protected:Npn \__draw_backend_fillstroke:
1634 { \__draw_backend_path:n { } }
1635 \cs_new_protected:Npn \__draw_backend_clip:
1636 { \bool_gset_true:N \g__draw_draw_clip_bool }
1637 \bool_new:N \g__draw_draw_clip_bool
1638 \cs_new_protected:Npn \__draw_backend_discardpath:
1639 {
1640     \bool_if:NT \g__draw_draw_clip_bool
1641     {
1642         \int_gincr:N \g__kernel_clip_path_int
1643         \__draw_backend_literal:e
1644         {
1645             < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1646             { ?nl }
1647             <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1648             < /clipPath >
1649         }
1650         \__kernel_backend_scope:e
1651         {
1652             clip-path =
1653             "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1654         }
1655     }
1656     \tl_gclear:N \g__draw_path_tl
1657     \bool_gset_false:N \g__draw_draw_clip_bool
1658 }

```

(End of definition for __draw_backend_path:n and others.)

<pre> __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: </pre>	<p>All of these ideas are properties of scopes in SVG. The only slight complexity is converting the dash array properly (doing any required maths).</p> <pre> 1659 \cs_new_protected:Npn __draw_backend_dash_pattern:nn #1#2 1660 { 1661 \use:e 1662 { 1663 __draw_backend_dash_aux:nn 1664 { \clist_map_function:nN {#1} __draw_backend_dash:n } 1665 { \dim_to_decimal:n {#2} } 1666 } </pre>
---	--

```

1667 }
1668 \cs_new:Npn \__draw_backend_dash:n #1
1669 { , \dim_to_decimal_in_bp:n {#1} }
1670 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1671 {
1672   \__kernel_backend_scope:e
1673   {
1674     stroke-dasharray =
1675     "
1676     \tl_if_empty:nTF {#1}
1677     { none }
1678     { \use_none:n #1 }
1679     " ~
1680     stroke-offset=" #2 "
1681   }
1682 }
1683 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1684 { \__kernel_backend_scope:e { stroke-width=" \dim_to_decimal:n {#1} " } }
1685 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1686 { \__kernel_backend_scope:e { stroke-miterlimit=" #1 " } }
1687 \cs_new_protected:Npn \__draw_backend_cap_but:
1688 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1689 \cs_new_protected:Npn \__draw_backend_cap_round:
1690 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1691 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1692 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1693 \cs_new_protected:Npn \__draw_backend_join_miter:
1694 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1695 \cs_new_protected:Npn \__draw_backend_join_round:
1696 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1697 \cs_new_protected:Npn \__draw_backend_join_bevel:
1698 { \__kernel_backend_scope:n { stroke-linejoin="bevel" } }

```

(End of 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.

```

1699 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1700 {
1701   \__kernel_backend_scope:n
1702   {
1703     transform =
1704     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1705   }
1706 }

```

(End of 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.

```

1707 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1708 {
1709   \__kernel_backend_scope_begin:
1710   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}

```

```

1711 \__kernel_backend_literal_svg:n
1712 {
1713   < g~
1714     stroke="none"~
1715     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1716   >
1717   }
1718   \box_set_wd:Nn #1 { Opt }
1719   \box_set_ht:Nn #1 { Opt }
1720   \box_set_dp:Nn #1 { Opt }
1721   \box_use:N #1
1722   \__kernel_backend_literal_svg:n { </g> }
1723   \__kernel_backend_scope_end:
1724 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```
1725 </dvisvgm>
```

```
1726 </package>
```

5 l3backend-graphics implementation

```

1727 <*package>
1728 <@@=graphics>

```

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

```

1729 \cs_new_protected:Npn \__graphics_backend_loaded:n #1
1730 {
1731   \cs_if_exist:NTF \hook_gput_code:nnn
1732   {
1733     \hook_gput_code:nnn
1734     { package / l3graphics / after }
1735     { backend }
1736     {#1}
1737   }
1738   {#1}
1739 }

```

(End of definition for __graphics_backend_loaded:n.)

5.1 dvips backend

```
1740 <*dvips>
```

\l_graphics_search_ext_seq

```

1741 \__graphics_backend_loaded:n
1742 { \seq_set_from_clist:Nn \l_graphics_search_ext_seq { .eps , .ps } }

```

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

_graphics_backend_getbb_eps:n

Simply use the generic function.

_graphics_backend_getbb_ps:n

```

1743 \__graphics_backend_loaded:n
1744 {
1745   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1746   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
1747 }

```

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

`__graphics_backend_include_eps:n`
`__graphics_backend_include_ps:n`

```

1748 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1749 {
1750   \__kernel_backend_literal:e
1751   {
1752     PSfile = #1 \c_space_tl
1753     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1754     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1755     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1756     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1757   }
1758 }
1759 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n

```

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

`__graphics_backend_get_pagecount:n`

```

1760 \__graphics_backend_loaded:n
1761 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }

```

(End of definition for `__graphics_backend_get_pagecount:n`.)

```

1762 </dvips>

```

5.2 LuaTeX and pdfTeX backends

1763 `< *luatex | pdftex >`

`\l_graphics_search_ext_seq`

```

1764 \__graphics_backend_loaded:n
1765 {
1766   \seq_set_from_clist:Nn
1767   \l_graphics_search_ext_seq
1768   { .pdf , .eps , .ps , .png , .jpg , .jpeg }
1769 }

```

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

`\l__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.

1770 `\tl_new:N \l__graphics_attr_tl`

(End of definition for `\l__graphics_attr_tl`.)

`__graphics_backend_getbb_jpg:n`
`__graphics_backend_getbb_jpeg:n`
`__graphics_backend_getbb_pdf:n`
`__graphics_backend_getbb_png:n`
`__graphics_backend_getbb_auxi:n`
`__graphics_backend_getbb_auxii:n`
`__graphics_backend_getbb_auxiii:n`
`__graphics_backend_dequote:w`

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

1771 `\cs_new_protected:Npn __graphics_backend_getbb_jpg:n #1`

```

1772 {
1773   \int_zero:N \l__graphics_page_int
1774   \tl_clear:N \l__graphics_pagebox_tl
1775   \tl_set:Ne \l__graphics_attr_tl
1776   {
1777     \tl_if_empty:NF \l__graphics_decodearray_str
1778     { :D \l__graphics_decodearray_str }
1779     \bool_if:NT \l__graphics_interpolate_bool
1780     { :I }
1781     \str_if_empty:NF \l__graphics_pdf_str
1782     { :X \l__graphics_pdf_str }
1783   }
1784   \__graphics_backend_getbb_auxi:n {#1}
1785 }
1786 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1787 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1788 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1789 {
1790   \tl_clear:N \l__graphics_decodearray_str
1791   \bool_set_false:N \l__graphics_interpolate_bool
1792   \tl_set:Ne \l__graphics_attr_tl
1793   {
1794     : \l__graphics_pagebox_tl
1795     \int_compare:nNnT \l__graphics_page_int > 1
1796     { :P \int_use:N \l__graphics_page_int }
1797     \str_if_empty:NF \l__graphics_pdf_str
1798     { :X \l__graphics_pdf_str }
1799   }
1800   \__graphics_backend_getbb_auxi:n {#1}
1801 }
1802 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1803 {
1804   \__graphics_bb_restore:eF { #1 \l__graphics_attr_tl }
1805   { \__graphics_backend_getbb_auxii:n {#1} }
1806 }

```

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.

```

1807 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1808 {
1809   \exp_args:Ne \__graphics_backend_getbb_auxiii:n
1810   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1811   \int_const:cn { c__graphics_ #1 \l__graphics_attr_tl _int }
1812   { \tex_the:D \tex_pdflastximage:D }
1813   \__graphics_bb_save:e { #1 \l__graphics_attr_tl }
1814 }
1815 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1816 {
1817   \tex_immediate:D \tex_pdfximage:D
1818   \bool_lazy_any:nT
1819   {
1820     { \l__graphics_interpolate_bool }

```



```

1821     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1822     { ! \str_if_empty_p:N \l__graphics_pdf_str }
1823   }
1824   {
1825     attr ~
1826     {
1827       \tl_if_empty:NF \l__graphics_decodearray_str
1828       { /Decode~[ \l__graphics_decodearray_str ] }
1829       \bool_if:NT \l__graphics_interpolate_bool
1830       { /Interpolate~true }
1831       \l__graphics_pdf_str
1832     }
1833   }
1834   \int_compare:nNnT \l__graphics_page_int > 0
1835   { page ~ \int_use:N \l__graphics_page_int }
1836   \tl_if_empty:NF \l__graphics_pagebox_tl
1837   { \l__graphics_pagebox_tl }
1838   {#1}
1839   \hbox_set:Nn \l__graphics_internal_box
1840   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1841   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1842   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1843   }
1844   \cs_new:Npn \__graphics_backend_dequote:w #1 " #2 " #3 \s__graphics_stop {#2}

```

(End of 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.

```

1845   \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1846   {
1847     \tex_pdfrefximage:D
1848     \int_use:c { c__graphics_ #1 \l__graphics_attr_tl_int }
1849   }
1850   \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1851   \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1852   \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

(End of 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
 \l__graphics_backend_dir_str
 \l__graphics_backend_name_str
 \l__graphics_backend_ext_str

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

```

1853   \sys_if_shell:T
1854   {
1855     \str_new:N \l__graphics_backend_dir_str
1856     \str_new:N \l__graphics_backend_name_str
1857     \str_new:N \l__graphics_backend_ext_str
1858     \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1859     {
1860       \file_parse_full_name:nNNN {#1}
1861       \l__graphics_backend_dir_str
1862       \l__graphics_backend_name_str

```

```

1863     \l__graphics_backend_ext_str
1864 \exp_args:Ne \__graphics_backend_getbb_eps:nn
1865 {
1866     \exp_args:Ne \__kernel_file_name_quote:n
1867     {
1868         \l__graphics_backend_name_str
1869         - \str_tail:N \l__graphics_backend_ext_str
1870         -converted-to.pdf
1871     }
1872 }
1873 {#1}
1874 }
1875 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_backend_getbb_eps:n
1876 \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1877 {
1878     \file_compare_timestamp:nNnT {#2} > {#1}
1879     {
1880         \sys_shell_now:n
1881         { repstopdf ~ #2 ~ #1 }
1882     }
1883     \tl_set:Nn \l__graphics_final_name_str {#1}
1884     \__graphics_backend_getbb_pdf:n {#1}
1885 }
1886 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1887 {
1888     \file_parse_full_name:nNNN {#1}
1889     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1890     \exp_args:Ne \__graphics_backend_include_pdf:n
1891     {
1892         \exp_args:Ne \__kernel_file_name_quote:n
1893         {
1894             \l__graphics_backend_name_str
1895             - \str_tail:N \l__graphics_backend_ext_str
1896             -converted-to.pdf
1897         }
1898     }
1899 }
1900 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1901 }

```

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

__graphics_backend_get_pagecount:n Simply load and store.

```

1902 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1903 {
1904     \tex_pdfximage:D {#1}
1905     \int_const:cn { c__graphics_ #1 _pages_int }
1906     { \int_use:N \tex_pdflastximagepages:D }
1907 }

```

(End of definition for __graphics_backend_get_pagecount:n.)

```

1908 </luatex | pdftex>

```

5.3 dvipdfmx backend

1909 $\langle *dvipdfmx | xetex \rangle$

$\backslash l_graphics_search_ext_seq$

```
1910 \__graphics_backend_loaded:n
1911 {
1912   \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1913   { .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }
1914 }
```

(End of 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.

$\backslash _graphics_backend_getbb_ps:n$

```
1915 \__graphics_backend_loaded:n
```

$\backslash _graphics_backend_getbb_jpg:n$

```
1916 {
```

$\backslash _graphics_backend_getbb_jpeg:n$

```
1917   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
```

$\backslash _graphics_backend_getbb_pdf:n$

```
1918   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
```

$\backslash _graphics_backend_getbb_png:n$

```
1919 }
```

$\backslash _graphics_backend_getbb_bmp:n$

```
1920  $\langle *dvipdfmx \rangle$ 
```

```
1921 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
```

```
1922 {
```

```
1923   \int_zero:N \l__graphics_page_int
```

```
1924   \tl_clear:N \l__graphics_pagebox_tl
```

```
1925   \__graphics_extract_bb:n {#1}
```

```
1926 }
```

```
1927 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
```

```
1928 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
```

```
1929 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
```

```
1930 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
```

```
1931 {
```

```
1932   \tl_clear:N \l__graphics_decodearray_str
```

```
1933   \bool_set_false:N \l__graphics_interpolate_bool
```

```
1934   \__graphics_extract_bb:n {#1}
```

```
1935 }
```

```
1936  $\langle /dvipdfmx \rangle$ 
```

(End of 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.

```
1937 \int_new:N \g__graphics_track_int
```

(End of 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.

$\backslash _graphics_backend_include_ps:n$

$\backslash _graphics_backend_include_jpg:n$

$\backslash _graphics_backend_include_jpeg:n$

$\backslash _graphics_backend_include_pdf:n$

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

$\backslash _graphics_backend_include_png:n$

```
1939 {
```

$\backslash _graphics_backend_include_bmp:n$

```
1940   \__kernel_backend_literal:e
```

```
1941   {
```

$\backslash _graphics_backend_include_auxi:n$

```
1942     PSfile = #1 \c_space_tl
```

$\backslash _graphics_backend_include_auxii:nn$

```
1943     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
```

$\backslash _graphics_backend_include_auxiii:enn$

```
1944     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
```

$\backslash _graphics_backend_include_auxiiii:nnn$

```
1945     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
```

```

1946         ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1947     }
1948 }
1949 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1950 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1951 { \__graphics_backend_include_auxi:nn {#1} { image } }
1952 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1953 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
1954 \cs_new_eq:NN \__graphics_backend_include_bmp:n \__graphics_backend_include_jpg:n
1955 \<dvipdfmx>
1956 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1957 { \__graphics_backend_include_auxi:nn {#1} { epdf } }
1958 \</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.

```

1959 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1960 {
1961     \__graphics_backend_include_auxii:enn
1962     {
1963         \tl_if_empty:NF \l__graphics_pagebox_tl
1964         { : \l__graphics_pagebox_tl }
1965         \int_compare:nNnT \l__graphics_page_int > 1
1966         { :P \int_use:N \l__graphics_page_int }
1967         \tl_if_empty:NF \l__graphics_decodearray_str
1968         { :D \l__graphics_decodearray_str }
1969         \bool_if:NT \l__graphics_interpolate_bool
1970         { :I }
1971     }
1972     {#1} {#2}
1973 }
1974 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1975 {
1976     \int_if_exist:cTF { c__graphics_ #2#1 _int }
1977     {
1978         \__kernel_backend_literal:e
1979         { pdf:usexobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
1980     }
1981     { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1982 }
1983 \cs_generate_variant:Nn \__graphics_backend_include_auxii:nnn { e }

```

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!

```

1984 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1985 {
1986     \int_gincr:N \g__graphics_track_int
1987     \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
1988     \__kernel_backend_literal:e
1989     {
1990         pdf:#3~

```

```

1991 @graphic \int_use:c { c__graphics_ #1#2 _int } ~
1992 \int_compare:nNnT \l__graphics_page_int > 1
1993 { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
1994 \tl_if_empty:NF \l__graphics_pagebox_tl
1995 {
1996   pagebox ~ \l__graphics_pagebox_tl \c_space_tl
1997   bbox ~
1998     \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1999     \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2000     \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2001     \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
2002   }
2003   (#1)
2004   \bool_lazy_or:nnT
2005     { \l__graphics_interpolate_bool }
2006     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
2007   {
2008     <<
2009     \tl_if_empty:NF \l__graphics_decodearray_str
2010     { /Decode~[ \l__graphics_decodearray_str ] }
2011     \bool_if:NT \l__graphics_interpolate_bool
2012     { /Interpolate~true }
2013     >>
2014   }
2015 }
2016 }

```

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

__graphics_backend_get_pagecount:n

```

2017 < *dvipdfmx >
2018 \__graphics_backend_loaded:n
2019 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }
2020 < /dvipdfmx >

```

(End of definition for __graphics_backend_get_pagecount:n.)

```

2021 < /dvipdfmx | xetex >

```

5.4 X_YTeX backend

```

2022 < *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.

```

2023 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2024 {
2025   \int_zero:N \l__graphics_page_int
2026   \tl_clear:N \l__graphics_pagebox_tl
2027   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2028 }
2029 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2030 \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: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

```

```

2031 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2032 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2033 {
2034   \tl_clear:N \l__graphics_decodearray_str
2035   \bool_set_false:N \l__graphics_interpolate_bool
2036   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2037 }
2038 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2039 {
2040   \int_compare:nNnTF \l__graphics_page_int > 1
2041     { \__graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2042     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2043 }
2044 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2045 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2046 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2047 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2048 {
2049   \tl_if_empty:NTF \l__graphics_pagebox_tl
2050     { \__graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2051     { \__graphics_backend_getbb_auxv:nNnn {#1} #2 {#3} {#4} }
2052 }
2053 }
2054 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2055 {
2056   \use:e
2057   {
2058     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2059     {
2060       #5
2061       \tl_if_blank:nF {#1}
2062         { \c_space_tl \__graphics_backend_getbb_pagebox:w #1 }
2063     }
2064   }
2065 }
2066 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2067 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2068 {
2069   \__graphics_bb_restore:nF {#1#3}
2070   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2071 }
2072 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2073 {
2074   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
2075   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2076   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2077   \__graphics_bb_save:n {#1#3}
2078 }
2079 \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

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

__graphics_backend_include_pdf:n For PDF graphics, properly supporting the pagebox concept in X_YTeX is best done using the \tex_XeTeXpdffile:D primitive. The syntax here is the same as for the graphic

measurement part, although we know at this stage that there must be some valid setting for `\l__graphics_pagebox_tl`.

```

2080 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2081 {
2082   \tex_XeTeXpdffile:D #1 ~
2083   \int_compare:nNnT \l__graphics_page_int > 0
2084     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2085     \exp_after:wN \__graphics_backend_getbb_pagebox:w \l__graphics_pagebox_tl
2086 }

```

(End of 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.

```

2087 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
2088 {
2089   \int_const:cn { c__graphics_ #1 _pages_int }
2090   {
2091     \int_max:nn
2092       { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2093       { 1 }
2094   }
2095 }

```

(End of definition for `__graphics_backend_get_pagecount:n`.)

```

2096 \xetex

```

5.5 dvisvgm backend

```

2097 \*dvisvgm

```

`\l_graphics_search_ext_seq`

```

2098 \__graphics_backend_loaded:n
2099 {
2100   \seq_set_from_clist:Nn
2101     \l_graphics_search_ext_seq
2102     { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }
2103 }

```

(End of 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.

```

2104 \cs_new_protected:Npn \__graphics_backend_getbb_svg:n #1
2105 {
2106   \__graphics_bb_restore:nF {#1}
2107   {
2108     \ior_open:Nn \l__graphics_internal_ior {#1}
2109     \ior_if_eof:NTF \l__graphics_internal_ior
2110       { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2111       {
2112         \dim_zero:N \l__graphics_llx_dim
2113         \dim_zero:N \l__graphics_lly_dim

```

```

2114 \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2115 \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2116 \ior_str_map_inline:Nn \l__graphics_internal_ior
2117 {
2118   \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2119   {
2120     \__graphics_backend_getbb_svg_auxi:nNn
2121     { width } \l__graphics_urx_dim {##1}
2122   }
2123   \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2124   {
2125     \__graphics_backend_getbb_svg_auxi:nNn
2126     { height } \l__graphics_ury_dim {##1}
2127   }
2128   \bool_lazy_and:nnF
2129   { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2130   { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2131   { \ior_map_break: }
2132 }
2133 \__graphics_bb_save:n {#1}
2134 }
2135 \ior_close:N \l__graphics_internal_ior
2136 }
2137 }
2138 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2139 {
2140   \use:e
2141   {
2142     \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxii:w
2143     ##1 \tl_to_str:n {#1} = ##2 \tl_to_str:n {#1} = ##3
2144     \s__graphics_stop
2145   }
2146   {
2147     \tl_if_blank:nF {##2}
2148     {
2149       \peek_remove_spaces:n
2150       {
2151         \peek_meaning:NTF ' % '
2152         { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2153         {
2154           \peek_meaning:NTF " % "
2155           { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2156           { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2157         }
2158       }
2159       ##2 \s__graphics_stop
2160     }
2161   }
2162   \use:e
2163   {
2164     \__graphics_backend_getbb_svg_auxii:w #3
2165     \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2166     \s__graphics_stop
2167   }

```



```

2168 }
2169 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2170 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop
2171 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2172 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2173 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2174 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2175 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2176 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2177 {
2178   \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2179   \l__graphics_internal_dim #2 bp \scan_stop:
2180   \dim_set_eq:NN #1 \l__graphics_internal_dim
2181 }
2182 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvii:w #1 \scan_stop: { }

```

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

_graphics_backend_getbb_eps:n
_graphics_backend_getbb_ps:n

Simply use the generic function.

```

2183 \__graphics_backend_loaded:n
2184 {
2185   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2186   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
2187 }

```

(End of 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.

```

2188 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2189 {
2190   \int_zero:N \l__graphics_page_int
2191   \tl_clear:N \l__graphics_pagebox_tl
2192   \__graphics_extract_bb:n {#1}
2193 }
2194 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2195 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n

```

(End of 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

```

2196 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2197 {
2198   \tl_clear:N \l__graphics_decodearray_str
2199   \bool_set_false:N \l__graphics_interpolate_bool
2200   \__graphics_extract_bb:n {#1}
2201 }

```

(End of 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.)

```

2202 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2203 { \__graphics_backend_include:nn { PSfile } {#1} }
2204 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n

```

```

2205 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2206 { \__graphics_backend_include:nn { pdf } {#1} }
2207 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
2208 {
2209   \__kernel_backend_literal:e
2210   {
2211     #1 = #2 \c_space_tl
2212     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2213     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2214     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2215     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2216   }
2217 }

```

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

```

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

```

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

```

2218 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2219 {
2220   \box_move_up:nn { \l__graphics_ury_dim }
2221   {
2222     \hbox:n
2223     {
2224       \__kernel_backend_literal:e
2225       {
2226         dvisvgm:img~
2227         \dim_to_decimal:n { \l__graphics_urx_dim } ~
2228         \dim_to_decimal:n { \l__graphics_ury_dim } ~
2229         \__graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2230       }
2231     }
2232   }
2233 }
2234 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_svg:n
2235 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_svg:n
2236 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_svg:n
2237 \cs_new:Npn \__graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2238 {#2}

```

(End of definition for __graphics_backend_include_svg:n and others.)

```

\__graphics_backend_get_pagecount:n

```

```

2239 \__graphics_backend_loaded:n
2240 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }

```

(End of definition for __graphics_backend_get_pagecount:n.)

```

2241 </dvisvgm>
2242 </package>

```

6 l3backend-pdf implementation

```
2243 <*package>
2244 <@@=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 most backends.

```
2245 <*!dvisvgm>

\l__pdf_internal_box

2246 \box_new:N \l__pdf_internal_box

(End of definition for \l__pdf_internal_box.)

2247 </!dvisvgm>
```

6.2 dvips backend

```
2248 <*dvips>

\__pdf_backend_pdfmark:n Used often enough it should be a separate function.
\__pdf_backend_pdfmark:e
2249 \cs_new_protected:Npn \__pdf_backend_pdfmark:n #1
2250 { \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2251 \cs_generate_variant:Nn \__pdf_backend_pdfmark:n { e }

(End of definition for \__pdf_backend_pdfmark:n.)
```

6.2.1 Catalogue entries

```
\__pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
2252 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2253 { \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2254 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2255 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }

(End of 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.
2256 \int_new:N \g__pdf_backend_object_int

(End of definition for \g__pdf_backend_object_int.)
```

```

__pdf_backend_object_new:n
__pdf_backend_object_ref:n
2257 \cs_new_protected:Npn __pdf_backend_object_new:n #1
2258 {
2259   \int_gincr:N \g__pdf_backend_object_int
2260   \int_const:cn
2261   { c__pdf_object_ \tl_to_str:n {#1} _int }
2262   { \g__pdf_backend_object_int }
2263 }
2264 \cs_new:Npn __pdf_backend_object_ref:n #1
2265 { { pdf.obj \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } } }

```

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

This is where we choose the actual type: some work to get things right. To allow code sharing with the anonymous version, we use an auxiliary.

```

__pdf_backend_object_write:nnn
__pdf_backend_object_write:nne
__pdf_backend_object_write_aux: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:nnn
2266 \cs_new_protected:Npn __pdf_backend_object_write:nnn #1#2#3
2267 {
2268   __pdf_backend_object_write_aux:nnn
2269   { __pdf_backend_object_ref:n {#1} }
2270   {#2} {#3}
2271 }
2272 \cs_generate_variant:Nn __pdf_backend_object_write:nnn { nne }
2273 \cs_new_protected:Npn __pdf_backend_object_write_aux:nnn #1#2#3
2274 {
2275   __pdf_backend_pdfmark:e
2276   {
2277     /objdef ~ #1
2278     /type
2279     \str_case:nn {#2}
2280     {
2281       { array } { /array }
2282       { dict } { /dict }
2283       { fstream } { /stream }
2284       { stream } { /stream }
2285     }
2286     /OBJ
2287   }
2288   \use:c { __pdf_backend_object_write_ #2 :nn } {#1} {#3}
2289 }
2290 \cs_new_protected:Npn __pdf_backend_object_write_array:nn #1#2
2291 {
2292   __pdf_backend_pdfmark:e
2293   { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2294 }
2295 \cs_new_protected:Npn __pdf_backend_object_write_dict:nn #1#2
2296 {
2297   __pdf_backend_pdfmark:e
2298   { #1 << \exp_not:n {#2} >> /PUT }
2299 }
2300 \cs_new_protected:Npn __pdf_backend_object_write_fstream:nn #1#2
2301 {
2302   \exp_args:Ne
2303   __pdf_backend_object_write_fstream:nnn {#1} #2
2304 }

```

```

2305 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2306 {
2307   \__kernel_backend_postscript:n
2308   {
2309     SDict ~ begin ~
2310     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2311     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2312     end
2313   }
2314 }
2315 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2316 {
2317   \exp_args:Ne
2318   \__pdf_backend_object_write_stream:nnn {#1} #2
2319 }
2320 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2321 {
2322   \__kernel_backend_postscript:n
2323   {
2324     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2325     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2326   }
2327 }

```

(End of definition for __pdf_backend_object_write:nnn and others.)

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

```

\__pdf_backend_object_now:ne
2328 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2329 {
2330   \int_gincr:N \g__pdf_backend_object_int
2331   \__pdf_backend_object_write_aux:nnn
2332   { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2333   {#1} {#2}
2334 }
2335 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last: Much like the annotation version.

```

2336 \cs_new:Npn \__pdf_backend_object_last:
2337 { { pdf.obj \int_use:N \g__pdf_backend_object_int } }

```

(End of definition for __pdf_backend_object_last:.)

__pdf_backend_pageobject_ref:n Page references are easy in dvips.

```

2338 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2339 { { Page #1 } }

```

(End of 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.
2340 \box_new:N \l__pdf_backend_content_box
(End of definition for \l__pdf_backend_content_box.)

\l__pdf_backend_model_box For creating model sizing for links.
2341 \box_new:N \l__pdf_backend_model_box
(End of definition for \l__pdf_backend_model_box.)

\g__pdf_backend_annotation_int Needed as objects which are not annotations could be created.
2342 \int_new:N \g__pdf_backend_annotation_int
(End of 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.
2343 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2344 {
2345   \exp_args:Nf \_pdf_backend_annotation_aux:nnnn
2346   { \dim_eval:n {#1} } {#2} {#3} {#4}
2347 }
2348 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2349 {
2350   \box_move_down:nn {#3}
2351   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2352   \box_move_up:nn {#2}
2353   {
2354     \hbox:n
2355     {
2356       \_kernel_kern:n {#1}
2357       \_kernel_backend_postscript:n { pdf.save.ur }
2358       \_kernel_kern:n { -#1 }
2359     }
2360   }
2361   \int_gincr:N \g__pdf_backend_object_int
2362   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2363   \_pdf_backend_pdfmark:e
2364   {
2365     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2366     pdf.rect
2367     #4 ~
2368     /ANN
2369   }
2370 }
(End of definition for \_pdf_backend_annotation:nnnn.)

```

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

```

2371 \cs_new:Npn \_pdf_backend_annotation_last:
2372 { { pdf.obj \int_use:N \g__pdf_backend_annotation_int } }

```

(End of definition for `_pdf_backend_annotation_last:.`)

`\g__pdf_backend_link_int` To track annotations which are links.

```

2373 \int_new:N \g__pdf_backend_link_int

```

(End of definition for `\g__pdf_backend_link_int.`)

`\g__pdf_backend_link_dict_tl` To pass information to the end-of-link function.

```

2374 \tl_new:N \g__pdf_backend_link_dict_tl

```

(End of definition for `\g__pdf_backend_link_dict_tl.`)

`\g__pdf_backend_link_sf_int` Needed to save/restore space factor, which is needed to deal with the face we need a box.

```

2375 \int_new:N \g__pdf_backend_link_sf_int

```

(End of definition for `\g__pdf_backend_link_sf_int.`)

`\g__pdf_backend_link_math_bool` Needed to save/restore math mode.

```

2376 \bool_new:N \g__pdf_backend_link_math_bool

```

(End of definition for `\g__pdf_backend_link_math_bool.`)

`\g__pdf_backend_link_bool` Track link formation: we cannot nest at all.

```

2377 \bool_new:N \g__pdf_backend_link_bool

```

(End of definition for `\g__pdf_backend_link_bool.`)

`\l__pdf_breaklink_pdfmark_tl` Swappable content for link breaking.

```

2378 \tl_new:N \l__pdf_breaklink_pdfmark_tl
2379 \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdfmark }

```

(End of definition for `\l__pdf_breaklink_pdfmark_tl.`)

`_pdf_breaklink_postscript:n` To allow dropping material unless link breaking is active.

```

2380 \cs_new_protected:Npn \_pdf_breaklink_postscript:n #1 { }

```

(End of definition for `_pdf_breaklink_postscript:n.`)

`__pdf_breaklink_usebox:N` Swappable box unpacking or use.

```

2381 \cs_new_eq:NN \_pdf_breaklink_usebox:N \box_use:N

```

(End of definition for `__pdf_breaklink_usebox:N.`)

```

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

```

Links are 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.

```

2382 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2383 {
2384   \__pdf_backend_link_begin:nw
2385   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2386 }
2387 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
2388 { \__pdf_backend_link_begin:nw {#1#2} }
2389 \cs_new_protected:Npn \__pdf_backend_link_begin:nw #1
2390 {
2391   \bool_if:NF \g__pdf_backend_link_bool
2392   { \__pdf_backend_link_begin_aux:nw {#1} }
2393 }

```

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.

```

2394 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2395 {
2396   \bool_gset_true:N \g__pdf_backend_link_bool
2397   \__kernel_backend_postscript:n
2398   { /pdf.link.dict ( #1 ) def }
2399   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2400   \__pdf_backend_link_sf_save:
2401   \mode_if_math:TF
2402   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2403   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2404   \hbox_set:Nw \l__pdf_backend_content_box
2405   \__pdf_backend_link_sf_restore:
2406   \bool_if:NT \g__pdf_backend_link_math_bool
2407   { \c_math_toggle_token }
2408 }
2409 \cs_new_protected:Npn \__pdf_backend_link_end:
2410 {
2411   \bool_if:NT \g__pdf_backend_link_bool
2412   { \__pdf_backend_link_end_aux: }
2413 }
2414 \cs_new_protected:Npn \__pdf_backend_link_end_aux:

```



```

2415 {
2416   \bool_if:NT \g__pdf_backend_link_math_bool
2417   { \c_math_toggle_token }
2418   \__pdf_backend_link_sf_save:
2419   \hbox_set_end:
2420   \__pdf_backend_link_minima:
2421   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2422   \exp_args:Ne \__pdf_backend_link_outerbox:n
2423   {
2424     \int_if_odd:nTF { \value { page } }
2425     { \oddsidemargin }
2426     { \evensidemargin }
2427   }
2428   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2429   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2430   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2431   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2432   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2433   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2434   {
2435     \hbox:n
2436     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2437   }
2438   \int_gincr:N \g__pdf_backend_object_int
2439   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2440   \__kernel_backend_postscript:e
2441   {
2442     mark
2443     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2444     \g__pdf_backend_link_dict_tl \c_space_tl
2445     pdf.rect
2446     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2447   }
2448   \__pdf_backend_link_sf_restore:
2449   \bool_gset_false:N \g__pdf_backend_link_bool
2450 }
2451 \cs_new_protected:Npn \__pdf_backend_link_minima:
2452 {
2453   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2454   \__kernel_backend_postscript:e
2455   {
2456     /pdf.linkdp.pad ~
2457     \dim_to_decimal:n
2458     {
2459       \dim_max:nn
2460       {
2461         \box_dp:N \l__pdf_backend_model_box
2462         - \box_dp:N \l__pdf_backend_content_box
2463       }
2464       { Opt }
2465     } ~
2466     pdf.pt.dvi ~ def
2467     /pdf.linkht.pad ~
2468     \dim_to_decimal:n

```

```

2469         {
2470             \dim_max:nn
2471             {
2472                 \box_ht:N \l__pdf_backend_model_box
2473                 - \box_ht:N \l__pdf_backend_content_box
2474             }
2475             { Opt }
2476         } ~
2477         pdf.pt.dvi ~ def
2478     }
2479 }
2480 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2481 {
2482     \__kernel_backend_postscript:e
2483     {
2484         /pdf.outerbox
2485         [
2486             \dim_to_decimal:n {#1} ~
2487             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2488             \dim_to_decimal:n { #1 + \textwidth } ~
2489             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2490         ]
2491         [ exch { pdf.pt.dvi } forall ] def
2492         /pdf.baselineskip ~
2493         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2494         { pdf.pt.dvi ~ def }
2495         { pop ~ pop }
2496         ifelse
2497     }
2498 }
2499 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2500 {
2501     \int_gset:Nn \g__pdf_backend_link_sf_int
2502     {
2503         \mode_if_horizontal:TF
2504         { \tex_spacefactor:D }
2505         { 0 }
2506     }
2507 }
2508 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2509 {
2510     \mode_if_horizontal:T
2511     {
2512         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2513         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2514     }
2515 }

```

(End of 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.

```

2516 \use_none:n
2517 {
2518   \cs_if_exist:NT \@makecol@hook
2519   {
2520     \tl_put_right:Nn \@makecol@hook
2521     {
2522       \box_if_empty:NF \l_shipout_box
2523       {
2524         \vbox_set:Nn \l_shipout_box
2525         {
2526           \__kernel_backend_postscript:n
2527           {
2528             pdf.globaldict /pdf.brokenlink.rect ~ known
2529             { pdf.bordertracking.continue }
2530             if
2531             }
2532             \vbox_unpack_drop:N \l_shipout_box
2533             \__kernel_backend_postscript:n
2534             { pdf.bordertracking.endpage }
2535           }
2536         }
2537       }
2538       \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2539       \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2540       \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2541     }
2542   }

```

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

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

```

2543 \cs_new:Npn \__pdf_backend_link_last:
2544 { { pdf.obj \int_use:N \g__pdf_backend_link_int } }

```

(End of definition for __pdf_backend_link_last:.)

__pdf_backend_link_margin:n Convert to big points and pass to PostScript.

```

2545 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2546 {
2547   \__kernel_backend_postscript:e
2548   {
2549     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2550   }
2551 }

```

(End of 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.

```

2552 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2553 {
2554   \__kernel_backend_postscript:n { pdf.dest.anchor }

```

```

2555 \__pdf_backend_pdfmark:e
2556 {
2557     /View
2558     [
2559         \str_case:nnF {#2}
2560         {
2561             { xyz } { /XYZ ~ pdf.dest.point ~ null }
2562             { fit } { /Fit }
2563             { fitb } { /FitB }
2564             { fitbh } { /FitBH ~ pdf.dest.y }
2565             { fitbv } { /FitBV ~ pdf.dest.x }
2566             { fith } { /FitH ~ pdf.dest.y }
2567             { fitv } { /FitV ~ pdf.dest.x }
2568             { fitr } { /Fit }
2569         }
2570         {
2571             /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2572         }
2573     ]
2574     /Dest ( \exp_not:n {#1} ) cvn
2575     /DEST
2576 }
2577 }
2578 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2579 {
2580     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2581     { \dim_eval:n {#2} } {#1} {#3} {#4}
2582 }
2583 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2584 {
2585     \vbox_to_zero:n
2586     {
2587         \__kernel_kern:n {#4}
2588         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2589         \tex_vss:D
2590     }
2591     \__kernel_kern:n {#1}
2592     \vbox_to_zero:n
2593     {
2594         \__kernel_kern:n { -#3 }
2595         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2596         \tex_vss:D
2597     }
2598     \__kernel_kern:n { -#1 }
2599     \__pdf_backend_pdfmark:n
2600     {
2601         /View
2602         [
2603             /FitR ~
2604             pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2605             pdf.urx ~ pdf.ury ~ pdf.dest2device
2606         ]
2607         /Dest ( #2 ) cvn
2608         /DEST

```

```

2609     }
2610 }

```

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

6.2.4 Structure

Doable for the usual ps2pdf method.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2611 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2612 {
2613   \int_compare:nNnT {#1} = 0
2614   {
2615     \__kernel_backend_literal_postscript:n
2616     {
2617       /setdistillerparams ~ where
2618       { pop << /CompressPages ~ false >> setdistillerparams }
2619       if
2620     }
2621   }
2622 }
2623 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2624 {
2625   \bool_if:nF {#1}
2626   {
2627     \__kernel_backend_literal_postscript:n
2628     {
2629       /setdistillerparams ~ where
2630       { pop << /CompressStreams ~ false >> setdistillerparams }
2631       if
2632     }
2633   }
2634 }

```

(End of definition for `__pdf_backend_compresslevel:n` and `__pdf_backend_compress_objects:n`.)

```

\__pdf_backend_version_major_gset:n
\__pdf_backend_version_minor_gset:n
2635 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2636 {
2637   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2638 }
2639 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2640 {
2641   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2642 }

```

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

`__pdf_backend_version_major:` Data not available!

```

\__pdf_backend_version_minor:
2643 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2644 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

```

(End of 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:
2645 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2646 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2647 \cs_new_protected:Npn \__pdf_backend_emc:
2648 { \__pdf_backend_pdfmark:n { /EMC } }

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

2649 </dvips>

```

6.3 LuaTeX and pdfTeX backend

```
2650 <*luatex | pdftex>
```

6.3.1 Annotations

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

```

\__pdf_backend_annotation:nnnn
2651 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2652 {
2653   <*luatex>
2654     \tex_pdfextension:D annot ~
2655   </luatex>
2656   <*pdftex>
2657     \tex_pdfannot:D
2658   </pdftex>
2659     width ~ \dim_eval:n {#1} ~
2660     height ~ \dim_eval:n {#2} ~
2661     depth ~ \dim_eval:n {#3} ~
2662     {#4}
2663   }

(End of definition for \__pdf_backend_annotation:nnnn.)

```

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.

```

\__pdf_backend_annotation_last:
2664 \cs_new:Npe \__pdf_backend_annotation_last:
2665 {
2666   \exp_not:N \int_value:w
2667   <*luatex>
2668     \exp_not:N \tex_pdffeedback:D lastannot ~
2669   </luatex>
2670   <*pdftex>
2671     \exp_not:N \tex_pdflastannot:D
2672   </pdftex>
2673   \c_space_tl 0 ~ R
2674   }

(End of definition for \__pdf_backend_annotation_last:.)

```

Links are all created using the same internals.

```

\__pdf_backend_link_begin_goto:nnw
\__pdf_backend_link_begin_user:nnw
\__pdf_backend_link_begin:nnnw
\__pdf_backend_link_end:
2675 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2676 { \__pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2677 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2

```

```

2678 { \_pdf_backend_link_begin:nnw {#1} { user } {#2} }
2679 \cs_new_protected:Npn \_pdf_backend_link_begin:nnw #1#2#3
2680 {
2681   \*luatex
2682   \tex_pdfextension:D startlink ~
2683   \*pdftex
2684   \tex_pdfstartlink:D
2685   \*pdftex
2686   attr {#1}
2687   #2 {#3}
2688 }
2689 \cs_new_protected:Npn \_pdf_backend_link_end:
2690 {
2691   \*luatex
2692   \tex_pdfextension:D endlink \scan_stop:
2693   \*pdftex
2694   \tex_pdfendlink:D
2695   \*pdftex
2696 }
2697
2698

```

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

_pdf_backend_link_last: Formatted for direct use.

```

2699 \cs_new:Npe \_pdf_backend_link_last:
2700 {
2701   \exp_not:N \int_value:w
2702   \*luatex
2703   \exp_not:N \tex_pdffeedback:D lastlink ~
2704   \*pdftex
2705   \exp_not:N \tex_pdflastlink:D
2706   \*pdftex
2707   \c_space_tl 0 ~ R
2708 }
2709

```

(End of definition for _pdf_backend_link_last:.)

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

```

2710 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2711 {
2712   \*luatex
2713   \tex_pdfvariable:D linkmargin
2714   \*pdftex
2715   \tex_pdflinkmargin:D
2716   \*pdftex
2717   \dim_eval:n {#1} \scan_stop:
2718 }
2719

```

(End of 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.

```

2720 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2721 {
2722   \*luatex
2723     \tex_pdfextension:D dest ~
2724   \*pdfTeX
2725     \tex_pdfdest:D
2726   \*pdfTeX
2727     name {#1}
2728     \str_case:nnF {#2}
2729     {
2730       { xyz } { xyz }
2731       { fit } { fit }
2732       { fitb } { fitb }
2733       { fitbh } { fitbh }
2734       { fitbv } { fitbv }
2735       { fith } { fith }
2736       { fitv } { fitv }
2737       { fitr } { fitr }
2738     }
2739     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2740   \scan_stop:
2741 }
2742
2743 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2744 {
2745   \*luatex
2746     \tex_pdfextension:D dest ~
2747   \*pdfTeX
2748     \tex_pdfdest:D
2749   \*pdfTeX
2750     name {#1}
2751     fitr ~
2752     width \dim_eval:n {#2} ~
2753     height \dim_eval:n {#3} ~
2754     depth \dim_eval:n {#4} \scan_stop:
2755 }
2756

```

(End of 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`

```

2757 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2758 {
2759   \*luatex
2760     \tex_pdfextension:D catalog
2761   \*pdfTeX
2762     \tex_pdfcatalog:D
2763   \*pdfTeX
2764

```



```

2765     { / #1 ~ #2 }
2766   }
2767   \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2768   {
2769     <*luatex>
2770     \tex_pdfextension:D info
2771   </luatex>
2772   <*pdftex>
2773   \tex_pdfinfo:D
2774   </pdftex>
2775   { / #1 ~ #2 }
2776 }

```

(End of 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.

```

2777 \prop_new:N \g__pdf_backend_object_prop

```

(End of definition for \g__pdf_backend_object_prop.)

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

```

\__pdf_backend_object_ref:n
2778 \cs_new_protected:Npn \__pdf_backend_object_new:n #1
2779 {
2780   <*luatex>
2781   \tex_pdfextension:D obj ~
2782 </luatex>
2783 <*pdftex>
2784   \tex_pdfobj:D
2785 </pdftex>
2786   reserveobjnum ~
2787   \int_const:cn
2788   { c__pdf_object_ \tl_to_str:n {#1} _int }
2789 <*luatex>
2790   { \tex_pdffeedback:D lastobj }
2791 </luatex>
2792 <*pdftex>
2793   { \tex_pdflastobj:D }
2794 </pdftex>
2795 }
2796 \cs_new:Npn \__pdf_backend_object_ref:n #1
2797 { \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

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

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

```

\__pdf_backend_object_write:nne
\__pdf_backend_object_write:nn
\__pdf_exp_not_i:nn
\__pdf_exp_not_ii:nn
2798 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2799 {
2800   <*luatex>
2801   \tex_immediate:D \tex_pdfextension:D obj ~
2802 </luatex>
2803 <*pdftex>
2804   \tex_immediate:D \tex_pdfobj:D
2805 </pdftex>

```

```

2806     useobjnum ~
2807     \int_use:c
2808     { c__pdf_object_ \tl_to_str:n {#1} _int }
2809     \__pdf_backend_object_write:nn {#2} {#3}
2810   }
2811 \cs_new:Npn \__pdf_backend_object_write:nn #1#2
2812 {
2813   \str_case:nn {#1}
2814   {
2815     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2816     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2817     { fstream }
2818     {
2819       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2820       file ~ { \__pdf_exp_not_ii:nn #2 }
2821     }
2822     { stream }
2823     {
2824       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2825       { \__pdf_exp_not_ii:nn #2 }
2826     }
2827   }
2828 }
2829 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2830 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2831 \cs_new:Npn \__pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

```

(End of definition for __pdf_backend_object_write:nnn and others.)

__pdf_backend_object_now:nn
 __pdf_backend_object_now:ne

Much like writing, but direct creation.

```

2832 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2833 {
2834   <*luatex>
2835   \tex_immediate:D \tex_pdfextension:D obj ~
2836   </luatex>
2837   <*pdftex>
2838   \tex_immediate:D \tex_pdfobj:D
2839   </pdftex>
2840   \__pdf_backend_object_write:nn {#1} {#2}
2841 }
2842 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

Much like annotation.

```

2843 \cs_new:Npe \__pdf_backend_object_last:
2844 {
2845   \exp_not:N \int_value:w
2846   <*luatex>
2847   \exp_not:N \tex_pdffeedback:D lastobj ~
2848   </luatex>
2849   <*pdftex>
2850   \exp_not:N \tex_pdflastobj:D
2851   </pdftex>
2852   \c_space_tl 0 ~ R

```

```
2853 }
(End of definition for \_pdf_backend_object_last:.)
```

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

```
2854 \cs_new:Npe \_pdf_backend_pageobject_ref:n #1
2855 {
2856   \exp_not:N \int_value:w
2857   <*luatex>
2858   \exp_not:N \tex_pdffeedback:D pageref
2859   </luatex>
2860   <*pdftex>
2861   \exp_not:N \tex_pdfpageref:D
2862   </pdftex>
2863   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2864 }
(End of 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
2865 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2866 {
2867   \tex_global:D
2868   <*luatex>
2869   \tex_pdfvariable:D compresslevel
2870   </luatex>
2871   <*pdftex>
2872   \tex_pdfcompresslevel:D
2873   </pdftex>
2874   \int_value:w \int_eval:n {#1} \scan_stop:
2875 }
2876 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2877 {
2878   \bool_if:nTF {#1}
2879   { \_pdf_backend_objcompresslevel:n { 2 } }
2880   { \_pdf_backend_objcompresslevel:n { 0 } }
2881 }
2882 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2883 {
2884   \tex_global:D
2885   <*luatex>
2886   \tex_pdfvariable:D objcompresslevel
2887   </luatex>
2888   <*pdftex>
2889   \tex_pdfobjcompresslevel:D
2890   </pdftex>
2891   #1 \scan_stop:
2892 }
(End of definition for \_pdf_backend_compresslevel:n, \_pdf_backend_compress_objects:n, and
\_pdf_backend_objcompresslevel:n.)
```

_pdf_backend_version_major_gset:n
_pdf_backend_version_minor_gset:n

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

```

2893 \cs_new_protected:Npe \__pdf_backend_version_major_gset:n #1
2894 {
2895   <*luatex>
2896     \int_compare:nNnT \tex luatexversion:D > { 106 }
2897     {
2898       \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2899       \exp_not:N \int_eval:n {#1} \scan_stop:
2900     }
2901   </luatex>
2902   <*pdftex>
2903     \cs_if_exist:NT \tex_pdfmajorversion:D
2904     {
2905       \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2906       \exp_not:N \int_eval:n {#1} \scan_stop:
2907     }
2908   </pdftex>
2909 }
2910 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2911 {
2912   \tex_global:D
2913   <*luatex>
2914     \tex_pdfvariable:D minorversion
2915   </luatex>
2916   <*pdftex>
2917     \tex_pdfminorversion:D
2918   </pdftex>
2919   \int_eval:n {#1} \scan_stop:
2920 }

```

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

_pdf_backend_version_major:
_pdf_backend_version_minor:

As above.

```

2921 \cs_new:Npe \__pdf_backend_version_major:
2922 {
2923   <*luatex>
2924     \int_compare:nNnTF \tex luatexversion:D > { 106 }
2925     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2926     { 1 }
2927   </luatex>
2928   <*pdftex>
2929     \cs_if_exist:NTF \tex_pdfmajorversion:D
2930     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2931     { 1 }
2932   </pdftex>
2933 }
2934 \cs_new:Npn \__pdf_backend_version_minor:
2935 {
2936   \tex_the:D
2937   <*luatex>
2938     \tex_pdfvariable:D minorversion
2939   </luatex>
2940   <*pdftex>

```

```

2941 \tex_pdfminorversion:D
2942 </pdfTeX>
2943 }

```

(End of 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>.

```

2944 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2945 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2946 \cs_new_protected:Npn \_pdf_backend_emc:
2947 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2948 </luatex | pdfTeX>

```

6.4 dvipdfmx backend

```

2949 <*dvipdfmx | xetex>

```

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

```

\_pdf_backend:e
2950 \cs_new_protected:Npe \_pdf_backend:n #1
2951 { \__kernel_backend_literal:n { pdf: #1 } }
2952 \cs_generate_variant:Nn \_pdf_backend:n { e }

```

(End of definition for _pdf_backend:n.)

6.4.1 Catalogue entries

```

\_pdf_backend_catalog_gput:nn
\_pdf_backend_info_gput:nn
2953 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2954 { \_pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2955 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2956 { \_pdf_backend:n { docinfo << /#1 ~ #2 >> } }

```

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

6.4.2 Objects

\g_pdf_backend_object_int For tracking objects to allow finalisation.

```

\g_pdf_backend_object_prop
2957 \int_new:N \g_pdf_backend_object_int
2958 \prop_new:N \g_pdf_backend_object_prop

```

(End of definition for \g_pdf_backend_object_int and \g_pdf_backend_object_prop.)

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

```

\_pdf_backend_object_ref:n
2959 \cs_new_protected:Npn \_pdf_backend_object_new:n #1
2960 {
2961   \int_gincr:N \g_pdf_backend_object_int
2962   \int_const:cn
2963   { c_pdf_object_ \tl_to_str:n {#1} _int }
2964   { \g_pdf_backend_object_int }

```

```

2965 }
2966 \cs_new:Npn \__pdf_backend_object_ref:n #1
2967 { @pdf.obj \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } }

```

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

This is where we choose the actual type.

```

\__pdf_backend_object_write:nnn
\__pdf_backend_object_write:nne
\__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
2968 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2969 {
2970   \use:c { __pdf_backend_object_write_ #2 :nn }
2971   { \__pdf_backend_object_ref:n {#1} } {#3}
2972 }
2973 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2974 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2975 {
2976   \__pdf_backend:e
2977   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2978 }
2979 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2980 {
2981   \__pdf_backend:e
2982   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2983 }
2984 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2985 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2986 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2987 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2988 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2989 {
2990   \__pdf_backend:e
2991   {
2992     #1 stream ~ #2 ~
2993     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2994   }
2995 }

```

(End of definition for __pdf_backend_object_write:nnn and others.)

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

```

2996 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2997 {
2998   \int_gincr:N \g__pdf_backend_object_int
2999   \exp_args:Nne \use:c { __pdf_backend_object_write_ #1 :nn }
3000   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
3001   {#2}
3002 }
3003 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

3004 \cs_new:Npn \__pdf_backend_object_last:
3005 { @pdf.obj \int_use:N \g__pdf_backend_object_int }

```

(End of definition for __pdf_backend_object_last:.)

`_pdf_backend_pageobject_ref:n` Page references are easy in dvipdfmx/X_YTeX.

```

3006 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
3007 { @page #1 }

```

(End of 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.

```

3008 \int_new:N \g__pdf_backend_annotation_int

```

(End of definition for `\g__pdf_backend_annotation_int`.)

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

```

3009 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
3010 {
3011   \int_gincr:N \g__pdf_backend_object_int
3012   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
3013   \_pdf_backend:e
3014   {
3015     ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
3016     width ~ \dim_eval:n {#1} ~
3017     height ~ \dim_eval:n {#2} ~
3018     depth ~ \dim_eval:n {#3} ~
3019     << /Type /Annot #4 >>
3020   }
3021 }

```

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

`_pdf_backend_annotation_last:`

```

3022 \cs_new:Npn \_pdf_backend_annotation_last:
3023 { @pdf.obj \int_use:N \g__pdf_backend_annotation_int }

```

(End of definition for `_pdf_backend_annotation_last:`.)

`\g__pdf_backend_link_int` To track annotations which are links.

```

3024 \int_new:N \g__pdf_backend_link_int

```

(End of 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:`

```

3025 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
3026 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
3027 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
3028 { \_pdf_backend_link_begin:n {#1#2} }
3029 \cs_new_protected:Npe \_pdf_backend_link_begin:n #1
3030 {
3031   \exp_not:N \int_gincr:N \exp_not:N \g__pdf_backend_link_int
3032   \_pdf_backend:e
3033   {
3034     bann ~
3035     @pdf.lnk
3036     \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
3037     \c_space_tl

```

```

3038         <<
3039         /Type /Annot
3040         #1
3041         >>
3042     }
3043 }
3044 \cs_new_protected:Npn \__pdf_backend_link_end:
3045 { \__pdf_backend:n { eann } }

```

(End of definition for __pdf_backend_link_begin_goto:nnw and others.)

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

```

3046 \cs_new:Npn \__pdf_backend_link_last:
3047 { @pdf.lnk \int_use:N \g__pdf_backend_link_int }

```

(End of definition for __pdf_backend_link_last:.)

__pdf_backend_link_margin:n Pass to dvipdfmx.

```

3048 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
3049 { \__kernel_backend_literal:e { dvipdfmx:config~g~ \dim_eval:n {#1} } }

```

(End of definition for __pdf_backend_link_margin:n.)

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

```

3050 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
3051 {
3052     \__pdf_backend:e
3053     {
3054         dest ~ ( \exp_not:n {#1} )
3055         [
3056             @thispage
3057             \str_case:nnF {#2}
3058             {
3059                 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
3060                 { fit } { /Fit }
3061                 { fitb } { /FitB }
3062                 { fitbh } { /FitBH }
3063                 { fitbv } { /FitBV ~ @xpos }
3064                 { fith } { /FitH ~ @ypos }
3065                 { fitv } { /FitV ~ @xpos }
3066                 { fitr } { /Fit }
3067             }
3068             { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
3069         ]
3070     }
3071 }
3072 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
3073 {
3074     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
3075     { \dim_eval:n {#2} } {#1} {#3} {#4}
3076 }

```



```

3077 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
3078 {
3079   \vbox_to_zero:n
3080   {
3081     \__kernel_kern:n {#4}
3082     \hbox:n
3083     {
3084       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
3085       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
3086     }
3087     \tex_vss:D
3088   }
3089   \__kernel_kern:n {#1}
3090   \vbox_to_zero:n
3091   {
3092     \__kernel_kern:n { -#3 }
3093     \hbox:n
3094     {
3095       \__pdf_backend:n
3096       {
3097         dest ~ (#2)
3098         [
3099           @thispage
3100           /FitR ~
3101           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
3102           @xpos ~ @ypos
3103         ]
3104       }
3105     }
3106     \tex_vss:D
3107   }
3108   \__kernel_kern:n { -#1 }
3109 }

```

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

6.4.4 Structure

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

```

\_pdf_backend_compress_objects:n
3110 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
3111 { \__kernel_backend_literal:e { dvipdfmx:config~z~ \int_eval:n {#1} } }
3112 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
3113 {
3114   \bool_if:nF {#1}
3115   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
3116 }

```

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

_pdf_backend_version_major_gset:n We start with the assumption that the default is active.

```

\_pdf_backend_version_minor_gset:n
3117 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
3118 {
3119   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
3120   \__kernel_backend_literal:e { pdf:majorversion~ \__pdf_backend_version_major: }

```

```

3121 }
3122 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
3123 {
3124   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
3125   \__kernel_backend_literal:e { pdf:minorversion~ \__pdf_backend_version_minor: }
3126 }

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

```

__pdf_backend_version_major: We start with the assumption that the default is active.

```

\__pdf_backend_version_minor: 3127 \cs_new:Npn \__pdf_backend_version_major: { 1 }
3128 \cs_new:Npn \__pdf_backend_version_minor: { 5 }

```

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

6.4.5 Marked content

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

```

3129 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
3130 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
3131 \cs_new_protected:Npn \__pdf_backend_emc:
3132 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

3133 </dvipdfmx | xetex>

```

6.5 dvisvgm backend

```

3134 <*dvisvgm>

```

6.5.1 Annotations

__pdf_backend_annotation:nnnn

```

3135 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4 { }

```

(End of definition for __pdf_backend_annotation:nnnn.)

__pdf_backend_annotation_last:

```

3136 \cs_new:Npn \__pdf_backend_annotation_last: { }

```

(End of definition for __pdf_backend_annotation_last:.)

__pdf_backend_link_begin_goto:nnw

__pdf_backend_link_begin_user:nnw

__pdf_backend_link_begin:nnnw

__pdf_backend_link_end:

```

3137 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2 { }
3138 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2 { }
3139 \cs_new_protected:Npn \__pdf_backend_link_begin:nnnw #1#2#3 { }
3140 \cs_new_protected:Npn \__pdf_backend_link_end: { }

```

(End of definition for __pdf_backend_link_begin_goto:nnw and others.)

__pdf_backend_link_last:

```

3141 \cs_new:Npe \__pdf_backend_link_last: { }

```

(End of definition for __pdf_backend_link_last:.)

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

```

3142 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1 { }

(End of definition for \_pdf_backend_link_margin:n.)

\_pdf_backend_destination:nn
\_pdf_backend_destination:nmmn
3143 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2 { }
3144 \cs_new_protected:Npn \_pdf_backend_destination:nmmn #1#2#3#4 { }

(End of definition for \_pdf_backend_destination:nn and \_pdf_backend_destination:nmmn.)

```

6.5.2 Catalogue entries

`_pdf_backend_catalog_gput:nn` No-op.

`_pdf_backend_info_gput:nn`

```

3145 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2 { }
3146 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2 { }

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

```

6.5.3 Objects

`_pdf_backend_object_new:n` All no-ops here.

`_pdf_backend_object_ref:n`

`_pdf_backend_object_write:nmm`

`_pdf_backend_object_write:nne`

`_pdf_backend_object_now:nn`

`_pdf_backend_object_now:ne`

`_pdf_backend_object_last:`

`_pdf_backend_pageobject_ref:n`

```

3147 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1 { }
3148 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
3149 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3 { }
3150 \cs_new_protected:Npn \_pdf_backend_object_write:nne #1#2#3 { }
3151 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
3152 \cs_new_protected:Npn \_pdf_backend_object_now:ne #1#2 { }
3153 \cs_new:Npn \_pdf_backend_object_last: { }
3154 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1 { }

(End of definition for \_pdf_backend_object_new:n and others.)

```

6.5.4 Structure

`_pdf_backend_compresslevel:n` These are all no-ops.

`_pdf_backend_compress_objects:n`

```

3155 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1 { }
3156 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1 { }

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

```

`_pdf_backend_version_major_gset:n` Data not available!

`_pdf_backend_version_minor_gset:n`

```

3157 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
3158 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1 { }

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

```

`_pdf_backend_version_major:` Data not available!

`_pdf_backend_version_minor:`

```

3159 \cs_new:Npn \_pdf_backend_version_major: { -1 }
3160 \cs_new:Npn \_pdf_backend_version_minor: { -1 }

(End of definition for \_pdf_backend_version_major: and \_pdf_backend_version_minor:.)

```

```

\__pdf_backend_bdc:nn More no-ops.
\__pdf_backend_emc: 3161 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
3162 \cs_new_protected:Npn \__pdf_backend_emc: { }
(End of definition for \__pdf_backend_bdc:nn and \__pdf_backend_emc:.)
3163 </dvisvgm>

```

6.6 PDF Page size (media box)

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

```

3164 <*dvipdfmx | dvips>

\__pdf_backend_pagesize_gset:nn This is done as a backend literal, so we deal with it using the shipout hook.
3165 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2
3166 {
3167   \__kernel_backend_first_shipout:n
3168   {
3169     \__kernel_backend_literal:e
3170     {
3171       <*dvipdfmx>
3172         pdf:pagesize ~
3173         width ~ \dim_eval:n {#1} ~
3174         height ~ \dim_eval:n {#2}
3175       </dvipdfmx>
3176       <*dvips>
3177         papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
3178       </dvips>
3179     }
3180   }
3181 }
(End of definition for \__pdf_backend_pagesize_gset:nn.)
3182 </dvipdfmx | dvips>
3183 <*luatex | pdftex | xetex>

```

```

\__pdf_backend_pagesize_gset:nn Pass to the primitives.
3184 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2
3185 {
3186   \dim_gset:Nn \tex_pagewidth:D {#1}
3187   \dim_gset:Nn \tex_pageheight:D {#2}
3188 }
(End of definition for \__pdf_backend_pagesize_gset:nn.)
3189 </luatex | pdftex | xetex>
3190 <*dvisvgm>

```

```

\__pdf_backend_pagesize_gset:nn A no-op.
3191 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2 { }
(End of definition for \__pdf_backend_pagesize_gset:nn.)
3192 </dvisvgm>
3193 </package>

```

7 l3backend-opacity implementation

```

3194 <*package>
3195 <@@=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.

```

3196 <*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.

```

3197 \cs_new_protected:Npn \__opacity_backend_select:n #1
3198 {
3199   \exp_args:Ne \__opacity_backend_select_aux:n
3200   { \fp_eval:n { min(max(0,#1),1) } }
3201 }
3202 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3203 {
3204   \__opacity_backend:nnn {#1} { fill } { ca }
3205   \__opacity_backend:nnn {#1} { stroke } { CA }
3206 }
3207 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3208 {
3209   \__opacity_backend:enn
3210   { \fp_eval:n { min(max(0,#1),1) } }
3211   { fill }
3212   { ca }
3213 }
3214 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3215 {
3216   \__opacity_backend:enn
3217   { \fp_eval:n { min(max(0,#1),1) } }
3218   { stroke }
3219   { CA }
3220 }
3221 \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3222 {
3223   \__kernel_backend_postscript:n
3224   {
3225     product ~ (Ghostscript) ~ search
3226     {
3227       pop ~ pop ~ pop ~
3228       #1 ~ .set #2 constantalpha
3229     }
3230     {
3231       pop ~
3232       mark ~
3233       /#3 ~ #1

```

```

3234         /SetTransparency ~
3235         pdfmark
3236     }
3237     ifelse
3238 }
3239 }
3240 \cs_generate_variant:Nn \_opacity_backend:nnn { e }

```

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

```

3241 </dvips>
3242 <*dvipdfmx | luatex | pdftex | xetex>

```

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

```

3243 \bool_lazy_and:nnT
3244 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3245 { \pdfmanagement_if_active_p:}
3246 {
3247 <*luatex | pdftex>
3248     \_kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3249     { page ~ direct } { /opacity 1 ~ gs }
3250 </luatex | pdftex>
3251     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3252     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3253 }

```

(End of 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
3254 \tl_new:N \l__opacity_backend_fill_tl
3255 \tl_new:N \l__opacity_backend_stroke_tl

```

(End of 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:
3256 \cs_new_protected:Npn \_opacity_backend_select:n #1
3257 {
3258     \exp_args:Ne \_opacity_backend_select_aux:n
3259     { \fp_eval:n { min(max(0,#1),1) } }
3260 }
3261 \cs_new_protected:Npn \_opacity_backend_select_aux:n #1
3262 {
3263     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3264     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3265     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3266     { opacity #1 }
3267     { << /ca ~ #1 /CA ~ #1 >> }
3268 <*dvipdfmx | xetex>
3269     \_kernel_backend_literal_pdf:n
3270 </dvipdfmx | xetex>
3271 <*luatex | pdftex>
3272     \_kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3273 </luatex | pdftex>
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 {
3285   <*dvipdfmx | xetex>
3286   \__kernel_backend_literal_pdf:n
3287   { /opacity1 ~ gs }
3288   </dvipdfmx | xetex>
3289   <*luatex | pdftex>
3290   \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int
3291   </luatex | pdftex>
3292 }

```

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

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

```

3293 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3294 {
3295   \__opacity_backend_fill_stroke:ee
3296   { \fp_eval:n { min(max(0,#1),1) } }
3297   \l__opacity_backend_stroke_tl
3298 }
3299 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3300 {
3301   \__opacity_backend_fill_stroke:ee
3302   \l__opacity_backend_fill_tl
3303   { \fp_eval:n { min(max(0,#1),1) } }
3304 }
3305 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3306 {
3307   \str_if_eq:nnTF {#1} {#2}
3308   { \__opacity_backend_select_aux:n {#1} }
3309   {
3310     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3311     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3312     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3313     { opacity.fill #1 }
3314     { << /ca ~ #1 >> }
3315     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3316     { opacity.stroke #1 }
3317     { << /CA ~ #2 >> }
3318   }
3319   <*dvipdfmx | xetex>
3320   \__kernel_backend_literal_pdf:n
3321   </dvipdfmx | xetex>
3322   <*luatex | pdftex>
3323   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3324   </luatex | pdftex>

```

```

3324         { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3325         \group_insert_after:N \__opacity_backend_reset:
3326     }
3327 }
3328 \cs_generate_variant:Nn \__opacity_backend_fill_stroke:nn { ee }

(End of definition for \__opacity_backend_fill:n, \__opacity_backend_stroke:n, and \__opacity_
backend_fillstroke:nn.)

3329 </dviptfm | luatex | pdftex | xetex>
3330 <*dvisvgm>

```

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

```

\__opacity_backend_fill:n
\__opacity_backend_stroke:n
\__opacity_backend:nn
3331 \cs_new_protected:Npn \__opacity_backend_select:n #1
3332 { \__opacity_backend:nn {#1} { } }
3333 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3334 { \__opacity_backend:nn {#1} { fill- } }
3335 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3336 { \__opacity_backend:nn { {#1} } { stroke- } }
3337 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3338 { \__kernel_backend_scope:e { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

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

3339 </dvisvgm>
3340 </package>

```

7.1 Font handling integration

In LuaTeX we want to use these functions also for transparent fonts to avoid interference between both uses of transparency.

```

3341 <*lua>

First we need to check if pdfmanagement is active from Lua.

3342 local pdfmanagement_active do
3343   local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3344   local cmd = pdfmanagement_if_active_p.cmdname
3345   if cmd == 'undefined_cs' then
3346     pdfmanagement_active = false
3347   else
3348     token.put_next(pdfmanagement_if_active_p)
3349     pdfmanagement_active = token.scan_int() ~= 0
3350   end
3351 end

3352
3353 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3354   luaotfload.set_transparent_colorstack(token.create'c__opacity_backend_stack_int'.index)
3355
3356   local transparent_register = {
3357     token.create'pdfmanagement_add:nnn',
3358     token.new(0, 1),
3359     'Page/Resources/ExtGState',
3360     token.new(0, 2),

```



```

3361     token.new(0, 1),
3362     '',
3363     token.new(0, 2),
3364     token.new(0, 1),
3365     '<</ca ',
3366     '',
3367     '/CA ',
3368     '',
3369     '>>',
3370     token.new(0, 2),
3371 }
3372 luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)
3373     value = (octet * -1):match(value)
3374     if not value then
3375         tex.error'Invalid transparency value'
3376         return
3377     end
3378     value = value:sub(1, -2)
3379     local result = 'opacity' .. value
3380     tex.runtoks(function()
3381         transparent_register[6], transparent_register[10], transparent_register[12] = result,
3382         tex.sprint(-2, transparent_register)
3383     end)
3384     return '/' .. result .. ' gs'
3385 end, 'l3opacity')
3386 end
3387 </lua>

```

8 l3backend-header implementation

```

3388 <*dvips & header>

```

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

```

3389 /color.sc { } def

```

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

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

```

3390 TeXDict begin
3391 /TeXcolorseparation { setcolor } def
3392 end

```

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

pdf.globaldict A small global dictionary for backend use.

```

3393 true setglobal
3394 /pdf.globaldict 4 dict def
3395 false setglobal

```

(End of 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	<pre> 3396 /pdf.cvs { 65534 string cvs } def 3397 /pdf.dvi.pt { 72.27 mul Resolution div } def 3398 /pdf.pt.dvi { 72.27 div Resolution mul } def 3399 /pdf.rect.ht { dup 1 get neg exch 3 get add } def </pre> <p>(End of definition for pdf.cvs and others. These functions are documented on page ??.)</p>
pdf.linkmargin	Settings which are defined up-front in SDict.
pdf.linkdp.pad	<pre> 3400 /pdf.linkmargin { 1 pdf.pt.dvi } def </pre>
pdf.linkht.pad	<pre> 3401 /pdf.linkdp.pad { 0 } def 3402 /pdf.linkht.pad { 0 } def </pre> <p>(End of definition for pdf.linkmargin, pdf.linkdp.pad, and pdf.linkht.pad. These functions are documented on page ??.)</p>
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	<pre> 3403 /pdf.rect </pre>
pdf.save.linkur	<pre> 3404 { /Rect [pdf.llx pdf.lly pdf.urx pdf.ury] } def </pre>
pdf.llx	<pre> 3405 /pdf.save.ll </pre>
pdf.lly	<pre> 3406 { </pre>
pdf.urx	<pre> 3407 currentpoint </pre>
pdf.ury	<pre> 3408 /pdf.lly exch def 3409 /pdf.llx exch def 3410 } </pre>
	<pre> 3411 def 3412 /pdf.save.ur </pre>
	<pre> 3413 { </pre>
	<pre> 3414 currentpoint </pre>
	<pre> 3415 /pdf.ury exch def 3416 /pdf.urx exch def 3417 } </pre>
	<pre> 3418 def 3419 /pdf.save.linkll </pre>
	<pre> 3420 { </pre>
	<pre> 3421 currentpoint 3422 pdf.linkmargin add 3423 pdf.linkdp.pad add 3424 /pdf.lly exch def 3425 pdf.linkmargin sub 3426 /pdf.llx exch def 3427 } </pre>
	<pre> 3428 def 3429 /pdf.save.linkur </pre>
	<pre> 3430 { </pre>
	<pre> 3431 currentpoint 3432 pdf.linkmargin sub 3433 pdf.linkht.pad sub 3434 /pdf.ury exch def 3435 pdf.linkmargin add </pre>

```

3436     /pdf.urx exch def
3437   }
3438   def

```

(End of 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 3439 /pdf.dest.anchor
pdf.dev.y 3440 {
pdf.tmpa 3441   currentpoint exch
pdf.tmpb 3442   pdf.dvi.pt 72 add
pdf.tmpc 3443   /pdf.dest.x exch def
pdf.tmpd 3444   pdf.dvi.pt
3445   vsize 72 sub exch sub
3446   /pdf.dest.y exch def
3447 }
3448 def
3449 /pdf.dest.point
3450 { pdf.dest.x pdf.dest.y } def
3451 /pdf.dest2device
3452 {
3453   /pdf.dest.y exch def
3454   /pdf.dest.x exch def
3455   matrix currentmatrix
3456   matrix defaultmatrix
3457   matrix invertmatrix
3458   matrix concatmatrix
3459   cvx exec
3460   /pdf.dev.y exch def
3461   /pdf.dev.x exch def
3462   /pdf.tmpd exch def
3463   /pdf.tmpc exch def
3464   /pdf.tmpb exch def
3465   /pdf.tmpa exch def
3466   pdf.dest.x pdf.tmpa mul
3467     pdf.dest.y pdf.tmpc mul add
3468     pdf.dev.x add
3469   pdf.dest.x pdf.tmpb mul
3470     pdf.dest.y pdf.tmpd mul add
3471     pdf.dev.y add
3472 }
3473 def

```

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

pdf.bordertracking To know where a breakable link can go, we need to track the boundary rectangle. That can be done by hooking into a and x operations: those names have to be retained. The boundary is stored at the end of the operation. Special effort is needed at the start and end of pages (or rather galleys), such that everything works properly.

```

pdf.bordertracking.begin 3474 /pdf.bordertracking false def
pdf.bordertracking.end
pdf.leftboundary
pdf.rightboundary
pdf.brokenlink.rect
pdf.brokenlink.skip
pdf.brokenlink.dict
pdf.bordertracking.endpage
pdf.bordertracking.continue
pdf.originx
pdf.originy

```

```

3475 /pdf.bordertracking.begin
3476 {
3477   SDict /pdf.bordertracking true put
3478   SDict /pdf.leftboundary undef
3479   SDict /pdf.rightboundary undef
3480   /a where
3481   {
3482     /a
3483     {
3484       currentpoint pop
3485       SDict /pdf.rightboundary known dup
3486       {
3487         SDict /pdf.rightboundary get 2 index lt
3488         { not }
3489         if
3490       }
3491       if
3492       { pop }
3493       { SDict exch /pdf.rightboundary exch put }
3494       ifelse
3495       moveto
3496       currentpoint pop
3497       SDict /pdf.leftboundary known dup
3498       {
3499         SDict /pdf.leftboundary get 2 index gt
3500         { not }
3501         if
3502       }
3503       if
3504       { pop }
3505       { SDict exch /pdf.leftboundary exch put }
3506       ifelse
3507     }
3508     put
3509   }
3510   if
3511 }
3512 def
3513 /pdf.bordertracking.end
3514 {
3515   /a where { /a { moveto } put } if
3516   /x where { /x { 0 exch rmoveto } put } if
3517   SDict /pdf.leftboundary known
3518   { pdf.outerbox 0 pdf.leftboundary put }
3519   if
3520   SDict /pdf.rightboundary known
3521   { pdf.outerbox 2 pdf.rightboundary put }
3522   if
3523   SDict /pdf.bordertracking false put
3524 }
3525 def
3526 /pdf.bordertracking.endpage
3527 {
3528   pdf.bordertracking

```

```

3529 {
3530     pdf.bordertracking.end
3531     true setglobal
3532     pdf.globaldict
3533     /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3534     pdf.globaldict
3535     /pdf.brokenlink.skip pdf.baselineskip put
3536     pdf.globaldict
3537     /pdf.brokenlink.dict
3538     pdf.link.dict pdf.cvs put
3539     false setglobal
3540     mark pdf.link.dict cvx exec /Rect
3541     [
3542         pdf.llx
3543         pdf.lly
3544         pdf.outerbox 2 get pdf.linkmargin add
3545         currentpoint exch pop
3546         pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3547     ]
3548     /ANN pdf.pdfmark
3549 }
3550 if
3551 }
3552 def
3553 /pdf.bordertracking.continue
3554 {
3555     /pdf.link.dict pdf.globaldict
3556     /pdf.brokenlink.dict get def
3557     /pdf.outerbox pdf.globaldict
3558     /pdf.brokenlink.rect get def
3559     /pdf.baselineskip pdf.globaldict
3560     /pdf.brokenlink.skip get def
3561     pdf.globaldict dup dup
3562     /pdf.brokenlink.dict undef
3563     /pdf.brokenlink.skip undef
3564     /pdf.brokenlink.rect undef
3565     currentpoint
3566     /pdf.originy exch def
3567     /pdf.originx exch def
3568     /a where
3569     {
3570         /a
3571         {
3572             moveto
3573             SDict
3574             begin
3575                 currentpoint pdf.originy ne exch
3576                 pdf.originx ne or
3577                 {
3578                     pdf.save.linkll
3579                     /pdf.lly
3580                     pdf.lly pdf.outerbox 1 get sub def
3581                     pdf.bordertracking.begin
3582                 }

```

```

3583         if
3584         end
3585     }
3586     put
3587 }
3588 if
3589 /x where
3590 {
3591     /x
3592     {
3593         0 exch rmoveto
3594         SDict
3595         begin
3596         currentpoint
3597         pdf.originy ne exch pdf.originx ne or
3598         {
3599             pdf.save.linkll
3600             /pdf.lly
3601             pdf.lly pdf.outerbox 1 get sub def
3602             pdf.bordertracking.begin
3603         }
3604         if
3605         end
3606     }
3607     put
3608 }
3609 if
3610 }
3611 def

```

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

<pre> pdf.breaklink pdf.breaklink.write pdf.count pdf.currentrect </pre>	<p>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.</p>
--	---

```

3612 /pdf.breaklink
3613 {
3614     pop
3615     counttomark 2 mod 0 eq
3616     {
3617         counttomark /pdf.count exch def
3618         {
3619             pdf.count 0 eq { exit } if
3620             counttomark 2 roll
3621             1 index /Rect eq
3622             {
3623                 dup 4 array copy
3624                 dup dup
3625                 1 get
3626                 pdf.outerbox pdf.rect.ht
3627                 pdf.linkmargin 2 mul add sub
3628                 3 exch put

```

```

3629     dup
3630     pdf.outerbox 2 get
3631     pdf.linkmargin add
3632     2 exch put
3633     dup dup
3634     3 get
3635     pdf.outerbox pdf.rect.ht
3636     pdf.linkmargin 2 mul add add
3637     1 exch put
3638     /pdf.currentrect exch def
3639     pdf.breaklink.write
3640     {
3641         pdf.currentrect
3642         dup
3643         pdf.outerbox 0 get
3644         pdf.linkmargin sub
3645         0 exch put
3646         dup
3647         pdf.outerbox 2 get
3648         pdf.linkmargin add
3649         2 exch put
3650         dup dup
3651         1 get
3652         pdf.baselineskip add
3653         1 exch put
3654         dup dup
3655         3 get
3656         pdf.baselineskip add
3657         3 exch put
3658         /pdf.currentrect exch def
3659         pdf.breaklink.write
3660     }
3661     1 index 3 get
3662     pdf.linkmargin 2 mul add
3663     pdf.outerbox pdf.rect.ht add
3664     2 index 1 get sub
3665     pdf.baselineskip div round cvi 1 sub
3666     exch
3667     repeat
3668     pdf.currentrect
3669     dup
3670     pdf.outerbox 0 get
3671     pdf.linkmargin sub
3672     0 exch put
3673     dup dup
3674     1 get
3675     pdf.baselineskip add
3676     1 exch put
3677     dup dup
3678     3 get
3679     pdf.baselineskip add
3680     3 exch put
3681     dup 2 index 2 get 2 exch put
3682     /pdf.currentrect exch def

```

```

3683         pdf.breaklink.write
3684         SDict /pdf.pdfmark.good false put
3685         exit
3686     }
3687     { pdf.count 2 sub /pdf.count exch def }
3688     ifelse
3689 }
3690 loop
3691 }
3692 if
3693 /ANN
3694 }
3695 def
3696 /pdf.breaklink.write
3697 {
3698     counttomark 1 sub
3699     index /_objdef eq
3700     {
3701         counttomark -2 roll
3702         dup wcheck
3703         {
3704             readonly
3705             counttomark 2 roll
3706         }
3707         { pop pop }
3708         ifelse
3709     }
3710     if
3711     counttomark 1 add copy
3712     pop pdf.currentrect
3713     /ANN pdfmark
3714 }
3715 def

```

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

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

```

3716 /pdf.pdfmark
3717 {
3718     SDict /pdf.pdfmark.good true put
3719     dup /ANN eq
3720     {
3721         pdf.pdfmark.store
3722         pdf.pdfmark.dict
3723         begin
3724             Subtype /Link eq
3725             currentdict /Rect known and
3726             SDict /pdf.outerbox known and
3727             SDict /pdf.baselineskip known and
3728             {

```



```

3729         Rect 3 get
3730         pdf.linkmargin 2 mul add
3731         pdf.outerbox pdf.rect.ht add
3732         Rect 1 get sub
3733         pdf.baselineskip div round cvi 0 gt
3734         { pdf.breaklink }
3735         if
3736     }
3737     if
3738     end
3739     SDict /pdf.outerbox undef
3740     SDict /pdf.baselineskip undef
3741     currentdict /pdf.pdfmark.dict undef
3742 }
3743 if
3744 pdf.pdfmark.good
3745 { pdfmark }
3746 { cleartomark }
3747 ifelse
3748 }
3749 def
3750 /pdf.pdfmark.store
3751 {
3752     /pdf.pdfmark.dict 65534 dict def
3753     counttomark 1 add copy
3754     pop
3755     {
3756         dup mark eq
3757         {
3758             pop
3759             exit
3760         }
3761         {
3762             pdf.pdfmark.dict
3763             begin def end
3764         }
3765     } ifelse
3766 }
3767 loop
3768 }
3769 def

```

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

```

3770 </dvips & header>

```

Index

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

- A**
- \AtBeginDvi 56
- B**
- bool commands:
- \bool_gset_false:N
..... 1201, 1220, 1243, 1265,
1281, 1382, 1621, 1657, 2403, 2449
 - \bool_gset_true:N
.. 1199, 1268, 1380, 1636, 2396, 2402
 - \bool_if:NTF 66,
578, 1211, 1215, 1231, 1234, 1238,
1249, 1256, 1260, 1272, 1276, 1393,
1398, 1403, 1595, 1640, 1779, 1829,
1969, 2011, 2391, 2406, 2411, 2416
 - \bool_if:nTF 2625, 2878, 3114
 - \bool_lazy_and:nnTF
..... 791, 2128, 3243, 3277
 - \bool_lazy_any:nTF 1818
 - \bool_lazy_or:nnTF 2004
 - \bool_new:N
.. 1202, 1269, 1383, 1637, 2376, 2377
 - \bool_set_false:N
..... 1791, 1933, 2035, 2199
- box commands:
- \box_dp:N
. 217, 219, 267, 269, 324, 326, 373,
375, 377, 379, 2428, 2461, 2462, 2487
 - \box_ht:N 219, 269, 326, 377,
379, 1842, 2076, 2433, 2472, 2473, 2489
 - \box_if_empty:NTF 2522
 - \box_move_down:nn 2350, 2428
 - \box_move_up:nn 2220, 2352, 2433
 - \box_new:N 2246, 2340, 2341
 - \box_set_dp:Nn 1720
 - \box_set_ht:Nn 1719
 - \box_set_wd:Nn 281, 1718
 - \box_use:N 224, 242,
256, 272, 299, 313, 329, 345, 357,
408, 422, 441, 1333, 1528, 1721, 2381
 - \box_wd:N 218, 226,
268, 274, 325, 331, 374, 376, 1841, 2075
- box internal commands:
- __box_backend_clip:N
206, 206, 261, 261, 318, 318, 362, 362
 - \l__box_backend_cos_fp 276
 - __box_backend_rotate:Nn
228, 228, 276, 276, 333, 333, 412, 412
 - __box_backend_rotate_aux:Nn 228,
229, 230, 276, 277, 278, 333, 334, 335
 - __box_backend_scale:Nnn
245, 245, 304, 304, 348, 348, 425, 425
 - \l__box_backend_sin_fp 276
- C**
- clist commands:
- \clist_map_function:nN
..... 1289, 1413, 1664
- color internal commands:
- __color_backend:nnn
..... 1027, 1034, 1049, 1057, 1063
 - __color_backend_cmyk:w 1028
 - \g__color_backend_colorant_prop .
..... 544, 563, 566, 586, 827
 - __color_backend_devicen_
colorants:n 545, 545, 747, 885
 - __color_backend_devicen_
colorants:w 545, 553, 560, 568
 - __color_backend_devicen_
init:nnn
..... 734, 734, 852, 852, 1084, 1084
 - __color_backend_devicen_init:w .
..... 852, 861, 890, 894
 - __color_backend_fill:n 931,
931, 933, 934, 935, 957, 958, 960,
962, 963, 982, 991, 992, 994, 996,
997, 1008, 1017, 1018, 1020, 1022, 1023
 - __color_backend_fill_cmyk:n 931,
933, 957, 957, 991, 991, 1017, 1017
 - __color_backend_fill_devicen:nn
..... 941,
951, 981, 985, 1007, 1011, 1078, 1080
 - __color_backend_fill_gray:n 931,
934, 957, 959, 991, 993, 1017, 1019
 - __color_backend_fill_reset: 953,
953, 987, 987, 1013, 1013, 1082, 1082
 - __color_backend_fill_rgb:n 931,
935, 957, 961, 991, 995, 1017, 1021
 - __color_backend_fill_separation:nn
..... 941, 941, 951, 981, 981,
985, 1007, 1007, 1011, 1078, 1078, 1080
 - \l__color_backend_fill_tl
..... 507, 519, 965, 979
 - __color_backend_iccbased_
device:nnn 914, 914

_color_backend_iccbased_- 576, 688, 758, 805, 830
init:nnn	
..... 753, 753, 896, 896, 1084, 1085	
_color_backend_init_resource:n	
..... 788, 788, 817, 888, 912, 927	
_color_backend_reset:	
..... 488, 503, 511, 523,	
527, 532, 953, 954, 987, 988, 1013, 1082	
_color_backend_rgb:w	1051
_color_backend_select:n	
..... 488, 489, 491, 493,	
495, 496, 527, 527, 529, 530, 531, 573	
_color_backend_select:nn	
..... 511, 512, 514, 516, 517, 784	
_color_backend_select_cmyk:n ..	
..... 488, 488, 511, 511, 527, 529	
_color_backend_select_devicen:nn	
..... 572, 574, 756, 757, 778, 786	
_color_backend_select_gray:n ..	
..... 488, 490, 511, 513, 527, 530, 537	
_color_backend_select_iccbased:nn	
..... 575, 575, 760, 760, 778, 787	
_color_backend_select_named:n ..	
..... 488, 492, 534, 534	
_color_backend_select_rgb:n ..	
..... 488, 494, 511, 515, 527, 531	
_color_backend_select_separation:nn	
..... 572, 572, 574,	
756, 756, 757, 778, 779, 783, 786, 787	
_color_backend_separation_-	
init:n	576, 657, 670
_color_backend_separation_-	
init:nn	805, 815, 819
_color_backend_separation_-	
init:nnn	576, 611, 632
_color_backend_separation_-	
init:nnnn	576, 634, 646
_color_backend_separation_-	
init:nnnnn	576,
576, 597, 690, 758, 758, 805, 805, 845	
_color_backend_separation_-	
init:nw	576, 661, 672, 686
_color_backend_separation_-	
init:w	576, 648, 663, 668
_color_backend_separation_-	
init_/DeviceCMYK:nnn	576
_color_backend_separation_-	
init_/DeviceGray:nnn	576
_color_backend_separation_-	
init_/DeviceRGB:nnn	576
_color_backend_separation_-	
init_aux:nnnnnn	576, 582, 598
_color_backend_separation_-	
init_CIELAB:nnn	
_color_backend_separation_-	
init_CIELAB:nnnnnn	759
_color_backend_separation_-	
init_count:n	576, 635, 638
_color_backend_separation_-	
init_count:w	576, 639, 640, 644
_color_backend_separation_-	
init_Device:Nn	
..... 576, 620, 622, 624, 625	
\l_color_backend_stack_int	
..... 449, 521, 524, 966, 978	
_color_backend_stroke:n	
..... 931, 936, 938,	
939, 940, 957, 970, 972, 974, 975, 984	
_color_backend_stroke_cmyk:n ..	
..... 931,	
938, 957, 969, 991, 1001, 1027, 1027	
_color_backend_stroke_cmyk:w ..	
..... 1027, 1029	
_color_backend_stroke_devicen:nn	
..... 941,	
952, 981, 986, 1007, 1012, 1078, 1081	
_color_backend_stroke_gray:n ..	
..... 931,	
939, 957, 971, 991, 1003, 1027, 1040	
_color_backend_stroke_gray_-	
aux:n	1027, 1044, 1048
_color_backend_stroke_reset: ..	
..... 953,	
954, 987, 988, 1013, 1014, 1082, 1083	
_color_backend_stroke_rgb:n ..	
..... 931,	
940, 957, 973, 991, 1005, 1027, 1050	
_color_backend_stroke_rgb:w ..	
..... 1027, 1052	
_color_backend_stroke_separation:nn	
..... 941, 946, 952, 981, 983,	
986, 1007, 1009, 1012, 1078, 1079, 1081	
\l_color_backend_stroke_tl	
..... 507, 520, 967, 977	
\g_color_model_int	583, 592, 740,
768, 817, 823, 824, 878, 879, 888, 912	
\c_color_model_range_CIELAB_tl ..	
..... 695, 730, 841, 848	
color.sc	488, 3389
cs commands:	
\cs_generate_variant:Nn	
... 62, 65, 98, 147, 152, 163, 194,	
200, 597, 1148, 1343, 1537, 1983,	
2046, 2066, 2251, 2272, 2335, 2829,	
2842, 2952, 2973, 3003, 3240, 3328	
\cs_gset:Npe ..	2637, 2641, 3119, 3124
\cs_gset_protected:Npn	3281

\cs_if_exist:NTF	1437, 1439, 1450, 1475, 1487, 1499,
..... 27, 49, 1731, 2518, 2903, 2929	1511, 1518, 1540, 1546, 1551, 1556,
\cs_if_exist_p:N	1567, 1577, 1587, 1589, 1591, 1593,
..... 792, 3244, 3278	1624, 1626, 1631, 1633, 1635, 1638,
\cs_if_exist_use:NTF	1659, 1670, 1683, 1685, 1687, 1689,
..... 38, 610	1691, 1693, 1695, 1697, 1699, 1707,
\cs_new:Npe	1729, 1748, 1771, 1788, 1802, 1807,
..... 545, 2664, 2699, 2843, 2854, 2921, 3141	1815, 1845, 1858, 1876, 1886, 1902,
\cs_new:Npn	1921, 1930, 1938, 1950, 1956, 1959,
..... 560, 619, 621, 623, 625, 632, 638,	1974, 1984, 2023, 2032, 2038, 2044,
640, 646, 663, 670, 672, 890, 1294,	2047, 2054, 2067, 2072, 2080, 2087,
1418, 1668, 1844, 2079, 2237, 2264,	2104, 2138, 2169, 2170, 2172, 2174,
2336, 2338, 2371, 2543, 2643, 2644,	2176, 2182, 2188, 2196, 2202, 2205,
2796, 2811, 2830, 2831, 2934, 2966,	2207, 2218, 2249, 2252, 2254, 2257,
3004, 3006, 3022, 3046, 3127, 3128,	2266, 2273, 2290, 2295, 2300, 2305,
3136, 3148, 3153, 3154, 3159, 3160	2315, 2320, 2328, 2343, 2348, 2380,
\cs_new_eq:NN	2382, 2387, 2389, 2394, 2409, 2414,
..... 46, 56, 58,	2451, 2480, 2499, 2508, 2545, 2552,
529, 530, 531, 574, 757, 786, 787,	2578, 2583, 2611, 2623, 2635, 2639,
933, 934, 935, 938, 939, 940, 951,	2645, 2647, 2651, 2675, 2677, 2679,
952, 953, 954, 985, 986, 987, 988,	2690, 2710, 2720, 2743, 2757, 2767,
1011, 1012, 1013, 1080, 1081, 1082,	2778, 2798, 2832, 2865, 2876, 2882,
1147, 1342, 1348, 1349, 1536, 1538,	2910, 2944, 2946, 2953, 2955, 2959,
1539, 1545, 1745, 1746, 1759, 1761,	2968, 2974, 2979, 2984, 2986, 2988,
1786, 1787, 1850, 1851, 1852, 1875,	2996, 3009, 3025, 3027, 3044, 3048,
1900, 1917, 1918, 1927, 1928, 1929,	3050, 3072, 3077, 3110, 3112, 3117,
1949, 1952, 1953, 1954, 2019, 2029,	3122, 3129, 3131, 3135, 3137, 3138,
2030, 2031, 2185, 2186, 2194, 2195,	3139, 3140, 3142, 3143, 3144, 3145,
2204, 2234, 2235, 2236, 2240, 2381	3146, 3147, 3149, 3150, 3151, 3152,
\cs_new_protected:Npe	3155, 3156, 3157, 3158, 3161, 3162,
..... 576, 1063, 2893, 2950, 3029	3165, 3184, 3191, 3197, 3202, 3207,
\cs_new_protected:Npn	3214, 3221, 3256, 3261, 3283, 3293,
..... 47, 53, 60, 63, 71, 77, 82,	3299, 3305, 3331, 3333, 3335, 3337
84, 88, 99, 109, 119, 128, 137, 150,	\cs_set_eq:NN
153, 155, 157, 161, 166, 175, 185, 2539, 2540
195, 206, 228, 230, 245, 261, 276,	\cs_set_protected:Npn
278, 304, 318, 333, 335, 348, 362, 2142
412, 425, 452, 466, 476, 488, 490,	
492, 494, 496, 503, 511, 513, 515,	
517, 523, 527, 532, 534, 572, 575,	
598, 688, 734, 753, 756, 758, 759,	
760, 779, 783, 788, 805, 819, 830,	
852, 896, 914, 931, 936, 941, 946,	
957, 959, 961, 963, 969, 971, 973,	
975, 981, 983, 991, 993, 995, 997,	
1001, 1003, 1005, 1007, 1009, 1014,	
1017, 1019, 1021, 1023, 1027, 1029,	
1040, 1048, 1050, 1052, 1078, 1079,	
1083, 1084, 1085, 1149, 1154, 1159,	
1161, 1163, 1171, 1179, 1188, 1198,	
1200, 1203, 1205, 1222, 1227, 1245,	
1267, 1270, 1283, 1296, 1301, 1303,	
1305, 1307, 1309, 1311, 1313, 1315,	
1320, 1344, 1346, 1350, 1355, 1360,	
1370, 1379, 1381, 1384, 1386, 1388,	
1390, 1395, 1400, 1405, 1407, 1420,	
1425, 1427, 1429, 1431, 1433, 1435,	

D

dim commands:

\dim_compare:nNnTF	2118, 2123
\dim_compare_p:nNn	2129, 2130
\dim_eval:n	
... 2346, 2581, 2659, 2660, 2661,	
2718, 2753, 2754, 2755, 3016, 3017,	
3018, 3049, 3075, 3173, 3174, 3177	
\dim_gset:Nn	3186, 3187
\dim_max:nn	2459, 2470
\dim_set:Nn	
.. 1841, 1842, 2075, 2076, 2114, 2115	
\dim_set_eq:NN	2180
\dim_to_decimal:n ..	373, 374, 375,
376, 377, 379, 1549, 1554, 1560,	
1561, 1562, 1563, 1572, 1573, 1574,	
1665, 1684, 2227, 2228, 2457, 2468,	
2486, 2487, 2488, 2489, 2493, 2549	

<code>\dim_to_decimal_in_bp:n</code>	<code>__draw_backend_dash_pattern:nn</code> ..
.... 217, 218, 219, 267, 268, 269,	.. 1283, 1283, 1407, 1407, 1659, 1659
324, 325, 326, 1167, 1168, 1175,	<code>__draw_backend_discardpath:</code> ...
1176, 1183, 1184, 1192, 1193, 1194,	.. 1203, 1270, 1384, 1405, 1591, 1638
1291, 1295, 1299, 1353, 1358, 1364,	<code>__draw_backend_end:</code>
1365, 1366, 1374, 1375, 1415, 1419,	.. 1149, 1154, 1344, 1346, 1540, 1545
1423, 1669, 1753, 1754, 1755, 1756,	<code>__draw_backend_evenodd_rule:</code> ...
1943, 1944, 1945, 1946, 1998, 1999,	.. 1198, 1198, 1379, 1379, 1587, 1587
2000, 2001, 2212, 2213, 2214, 2215	<code>__draw_backend_fill:</code>
<code>\dim_zero:N</code> 1203, 1227, 1384, 1390, 1591, 1631
2112, 2113	<code>__draw_backend_fillstroke:</code>
<code>\c_max_dim</code> 1203, 1245, 1384, 1395, 1591, 1633
.. 2114, 2115, 2118, 2123, 2129, 2130	<code>__draw_backend_join_bevel:</code>
draw internal commands:	.. 1283, 1313, 1407, 1437, 1659, 1697
<code>__draw_align_currentpoint:...</code> .. 36	<code>__draw_backend_join_miter:</code>
<code>__draw_backend_add_to_path:n</code> 1283, 1309, 1407, 1433, 1659, 1693
..... 1546,	<code>__draw_backend_join_round:</code>
1548, 1553, 1558, 1569, 1577, 1592	.. 1283, 1311, 1407, 1435, 1659, 1695
<code>__draw_backend_begin:</code>	<code>__draw_backend_lineto:nn</code>
.. 1149, 1149, 1344, 1344, 1540, 1540	.. 1163, 1171, 1350, 1355, 1546, 1551
<code>__draw_backend_box_use:Nnnnn</code> ...	<code>__draw_backend_linewidth:n</code>
31, 1320, 1320, 1518, 1518, 1707, 1707	.. 1283, 1296, 1407, 1420, 1659, 1683
<code>__draw_backend_cap_butt:</code>	<code>__draw_backend_literal:n</code>
.. 1283, 1303, 1407, 1427, 1659, 1687 1147, 1147, 1148, 1152,
<code>__draw_backend_cap_rectangle:</code> ..	1156, 1160, 1162, 1165, 1173, 1181,
.. 1283, 1307, 1407, 1431, 1659, 1691	1190, 1204, 1207, 1208, 1209, 1210,
<code>__draw_backend_cap_round:</code>	1213, 1219, 1229, 1236, 1242, 1247,
.. 1283, 1305, 1407, 1429, 1659, 1689	1252, 1253, 1254, 1255, 1258, 1264,
<code>__draw_backend_clip:</code>	1274, 1280, 1285, 1298, 1302, 1304,
.. 1203, 1267, 1384, 1400, 1591, 1635	1306, 1308, 1310, 1312, 1314, 1317,
<code>__draw_backend_closepath:</code>	1322, 1323, 1324, 1325, 1326, 1327,
..... 1203, 1203,	1331, 1332, 1334, 1335, 1336, 1337,
1224, 1384, 1384, 1591, 1591, 1628	1338, 1342, 1342, 1343, 1352, 1357,
<code>__draw_backend_closestroke:</code> ...	1362, 1372, 1385, 1387, 1389, 1392,
.. 1203, 1222, 1384, 1388, 1591, 1626	1397, 1402, 1406, 1409, 1422, 1426,
<code>__draw_backend_cm:nnnn</code>	1428, 1430, 1432, 1434, 1436, 1438,
... 1315, 1315, 1328, 1329, 1330,	1536, 1536, 1537, 1598, 1617, 1643
1439, 1439, 1522, 1699, 1699, 1710	<code>__draw_backend_miterlimit:n</code> ...
<code>__draw_backend_cm_aux:nnnn</code> 1283, 1301, 1407, 1425, 1659, 1685
..... 1439, 1446, 1450	<code>__draw_backend_moveto:nn</code>
<code>__draw_backend_cm_decompose:nnnnN</code>	.. 1163, 1163, 1350, 1350, 1546, 1546
..... 1445, 1474, 1475	<code>__draw_backend_nonzero_rule:</code> ...
<code>__draw_backend_cm_decompose_-</code>	.. 1198, 1200, 1379, 1381, 1587, 1589
auxi:nnnnN	<code>__draw_backend_path:n</code>
1474, 1479, 1487 1591, 1593, 1625, 1632, 1634
<code>__draw_backend_cm_decompose_-</code>	<code>\g__draw_backend_path_int</code> 1606, 1623
auxii:nnnnN	<code>\g__draw_backend_path_tl</code>
1474, 1491, 1499 1546, 1602, 1618, 1620, 1647
<code>__draw_backend_cm_decompose_-</code>	<code>__draw_backend_rectangle:nnnn</code> ..
auxiii:nnnnN 1163, 1179, 1350, 1370, 1546, 1556
1474, 1503, 1511	<code>__draw_backend_scope_begin:</code> 1159,
<code>__draw_backend_curveto:nnnnnn</code> ..	1159, 1345, 1348, 1348, 1538, 1538
.. 1163, 1188, 1350, 1360, 1546, 1567	<code>__draw_backend_scope_end:</code> 1159,
<code>__draw_backend_dash:n</code>	1161, 1347, 1348, 1349, 1538, 1539
..... 1283, 1289, 1294,	
1407, 1413, 1418, 1659, 1664, 1668	
<code>__draw_backend_dash_aux:nn</code>	
..... 1659, 1663, 1670	

`__draw_backend_stroke:` [1203](#), [1205](#),
[1225](#), [1384](#), [1386](#), [1591](#), [1624](#), [1629](#)
`\g__draw_draw_clip_bool` . . . [1203](#), [1591](#)
`\g__draw_draw_eor_bool`
. . . [1198](#), [1215](#), [1231](#), [1238](#), [1249](#),
[1260](#), [1276](#), [1379](#), [1393](#), [1398](#), [1403](#)
`\g__draw_draw_path_int` [1591](#)
`\g__draw_path_tl` [1656](#)

E

`\errmessage` [38](#)
`\evensidemargin` [2426](#)
exp commands:
`\exp_after:wN` [2085](#)
`\exp_args:Ne` [580](#), [634](#), [815](#),
[1809](#), [1864](#), [1866](#), [1890](#), [1892](#), [2302](#),
[2317](#), [2422](#), [2580](#), [3074](#), [3199](#), [3258](#)
`\exp_args:Nf` [1288](#), [1412](#), [2345](#)
`\exp_args:Nne` [2999](#)
`\exp_args:NNf` [229](#), [277](#), [334](#)
`\exp_not:N` [547](#), [553](#), [554](#), [555](#), [580](#),
[582](#), [583](#), [586](#), [587](#), [592](#), [2666](#), [2668](#),
[2671](#), [2701](#), [2703](#), [2706](#), [2845](#), [2847](#),
[2850](#), [2856](#), [2858](#), [2861](#), [2898](#), [2899](#),
[2905](#), [2906](#), [2925](#), [2930](#), [3031](#), [3036](#)
`\exp_not:n` [48](#), [96](#), [107](#), [145](#),
[904](#), [2293](#), [2298](#), [2574](#), [2815](#), [2816](#),
[2830](#), [2831](#), [2977](#), [2982](#), [2993](#), [3054](#)
`\ExplBackendFileDate` [1](#)

F

file commands:
`\file_compare_timestamp:nNnTF` . . . [1878](#)
`\file_parse_full_name:nNNN` [1860](#), [1888](#)
`\fmtversion` [51](#)
fp commands:
`\fp_compare:nNnTF`
. . . [236](#), [283](#), [289](#), [341](#), [1455](#), [1468](#), [1513](#)
`\fp_eval:n` [229](#), [238](#), [251](#), [252](#), [277](#),
[294](#), [309](#), [311](#), [334](#), [343](#), [354](#), [355](#),
[419](#), [434](#), [435](#), [1035](#), [1036](#), [1037](#),
[1045](#), [1058](#), [1059](#), [1060](#), [1457](#), [1462](#),
[1463](#), [1470](#), [1480](#), [1481](#), [1482](#), [1483](#),
[1492](#), [1493](#), [1494](#), [1495](#), [1504](#), [1505](#),
[1506](#), [1507](#), [2571](#), [2740](#), [3068](#), [3200](#),
[3210](#), [3217](#), [3259](#), [3296](#), [3303](#), [3338](#)
`\fp_new:N` [302](#), [303](#)
`\fp_set:Nn` [282](#), [285](#)
`\fp_use:N` [288](#), [292](#), [297](#)
`\fp_zero:N` [284](#)
`\c_zero_fp` [236](#), [283](#), [289](#), [341](#), [1455](#), [1468](#)

G

graphics commands:
`\l_graphics_search_ext_seq`
. [1741](#), [1764](#), [1910](#), [2098](#)
graphics internal commands:
`\l__graphics_attr_tl` [1770](#),
[1775](#), [1792](#), [1804](#), [1811](#), [1813](#), [1848](#)
`__graphics_backend_dequote:w`
. [1771](#), [1810](#), [1844](#)
`\l__graphics_backend_dir_str` . . . [1853](#)
`\l__graphics_backend_ext_str` . . . [1853](#)
`__graphics_backend_get_pagecount:n`
. [1760](#), [1761](#), [1902](#), [1902](#),
[2017](#), [2019](#), [2087](#), [2087](#), [2239](#), [2240](#)
`__graphics_backend_getbb_auxi:n`
. [1771](#), [1784](#), [1800](#), [1802](#)
`__graphics_backend_getbb_-`
`auxi:nN` [2023](#), [2027](#), [2036](#), [2038](#)
`__graphics_backend_getbb_-`
`auxii:n` [1771](#), [1805](#), [1807](#)
`__graphics_backend_getbb_-`
`auxiii:nnN` [2023](#), [2041](#), [2044](#), [2046](#)
`__graphics_backend_getbb_-`
`auxiii:n` [1771](#), [1809](#), [1815](#)
`__graphics_backend_getbb_-`
`auxiii:nNnn` [2023](#), [2042](#), [2045](#), [2047](#)
`__graphics_backend_getbb_-`
`auxiv:nnNnn` [2023](#), [2050](#), [2054](#), [2066](#)
`__graphics_backend_getbb_-`
`auxv:nNnn` [2023](#), [2051](#), [2058](#), [2067](#)
`__graphics_backend_getbb_-`
`auxvi:nNnn` [2070](#), [2072](#)
`__graphics_backend_getbb_bmp:n`
. [1915](#), [1929](#), [2023](#), [2031](#)
`__graphics_backend_getbb_eps:n`
. [1743](#), [1745](#), [1853](#),
[1858](#), [1875](#), [1915](#), [1917](#), [2183](#), [2185](#)
`__graphics_backend_getbb_eps:nm`
. [1853](#)
`__graphics_backend_getbb_eps:nn`
. [1864](#), [1876](#)
`__graphics_backend_getbb_jpeg:n`
. [1771](#), [1786](#),
[1915](#), [1927](#), [2023](#), [2029](#), [2188](#), [2194](#)
`__graphics_backend_getbb_jpg:n`
[1771](#), [1771](#), [1786](#), [1787](#), [1915](#), [1921](#),
[1927](#), [1928](#), [1929](#), [2023](#), [2023](#), [2029](#),
[2030](#), [2031](#), [2188](#), [2188](#), [2194](#), [2195](#)
`__graphics_backend_getbb_-`
`pagebox:w` [2023](#), [2062](#), [2079](#), [2085](#)
`__graphics_backend_getbb_pdf:n`
. [1771](#), [1788](#), [1884](#),
[1915](#), [1930](#), [2023](#), [2032](#), [2196](#), [2196](#)

```

\__graphics_backend_getbb_png:n .
    ..... 1771, 1787,
    1915, 1928, 2023, 2030, 2188, 2195
\__graphics_backend_getbb_ps:n ..
    ..... 1743, 1746,
    1853, 1875, 1915, 1918, 2183, 2186
\__graphics_backend_getbb_svg:n .
    ..... 2104, 2104
\__graphics_backend_getbb_svg_-
auxi:nNn ... 2104, 2120, 2125, 2138
\__graphics_backend_getbb_svg_-
auxii:w .... 2104, 2142, 2164, 2169
\__graphics_backend_getbb_svg_-
auxiii:Nw ..... 2104, 2152, 2170
\__graphics_backend_getbb_svg_-
auxiv:Nw ..... 2104, 2155, 2172
\__graphics_backend_getbb_svg_-
auxv:Nw ..... 2104, 2156, 2174
\__graphics_backend_getbb_svg_-
auxvi:Nn 2104, 2171, 2173, 2175, 2176
\__graphics_backend_getbb_svg_-
auxvii:w ..... 2104, 2178, 2182
\__graphics_backend_include:nn ..
    ..... 2202, 2203, 2206, 2207
\__graphics_backend_include_-
auxi:nn .... 1938, 1951, 1957, 1959
\__graphics_backend_include_-
auxii:nnn .. 1938, 1961, 1974, 1983
\__graphics_backend_include_-
auxiii:nnn ..... 1938, 1981, 1984
\__graphics_backend_include_-
bmp:n ..... 1938, 1954
\__graphics_backend_include_-
dequote:w ..... 2218, 2229, 2237
\__graphics_backend_include_-
eps:n ..... 1748,
1748, 1759, 1853, 1886, 1900,
1938, 1938, 1949, 2202, 2202, 2204
\__graphics_backend_include_-
jpeg:n . 1845, 1850, 1952, 2218, 2235
\__graphics_backend_include_-
jpg:n ..... 1845,
1845, 1850, 1851, 1852, 1938,
1950, 1952, 1953, 1954, 2218, 2236
\__graphics_backend_include_-
jpseg:n ..... 1938
\__graphics_backend_include_-
pdf:n ..... 1845, 1851, 1890,
1938, 1956, 2080, 2080, 2202, 2205
\__graphics_backend_include_-
png:n .....
.. 1845, 1852, 1938, 1953, 2218, 2234
\__graphics_backend_include_ps:n
    ..... 1748, 1759,
    1853, 1900, 1938, 1949, 2202, 2204
\__graphics_backend_include_-
svg:n .. 2218, 2218, 2234, 2235, 2236
\__graphics_backend_loaded:n ...
    1729, 1729, 1741, 1743, 1760, 1764,
    1910, 1915, 2018, 2098, 2183, 2239
\l__graphics_backend_name_str . 1853
\__graphics_bb_restore:nTF .....
    ..... 1804, 2069, 2106
\__graphics_bb_save:n 1813, 2077, 2133
\l__graphics_decodearray_str ...
    ..... 1777, 1778,
    1790, 1821, 1827, 1828, 1932, 1967,
    1968, 2006, 2009, 2010, 2034, 2198
\__graphics_extract_bb:n .....
    ..... 1925, 1934, 2192, 2200
\l__graphics_final_name_str .. 1883
\__graphics_get_pagecount:n ....
    ..... 1761, 2019, 2240
\l__graphics_internal_box .....
.. 1839, 1841, 1842, 2074, 2075, 2076
\l__graphics_internal_dim 2179, 2180
\l__graphics_internal_ior .....
    ..... 2108, 2109, 2116, 2135
\l__graphics_interpolate_bool ...
    ..... 1779, 1791, 1820, 1829,
    1933, 1969, 2005, 2011, 2035, 2199
\l__graphics_llx_dim .....
    ..... 1753, 1943, 1998, 2112, 2212
\l__graphics_lly_dim .....
    ..... 1754, 1944, 1999, 2113, 2213
\l__graphics_page_int .....
    ..... 1773, 1795, 1796, 1834,
    1835, 1923, 1965, 1966, 1992, 1993,
    2025, 2040, 2041, 2083, 2084, 2190
\l__graphics_pagebox_tl .....
    ..... 55, 1774, 1794,
    1836, 1837, 1924, 1963, 1964, 1994,
    1996, 2026, 2049, 2050, 2085, 2191
\l__graphics_pdf_str .....
.. 1781, 1782, 1797, 1798, 1822, 1831
\__graphics_read_bb:n .....
.. 1745, 1746, 1917, 1918, 2185, 2186
\g__graphics_track_int .....
    ..... 1937, 1986, 1987
\l__graphics_urx_dim .....
... 1755, 1841, 1945, 2000, 2075,
2114, 2118, 2121, 2129, 2214, 2227
\l__graphics_ury_dim .....
    1756, 1842, 1946, 2001, 2076, 2115,
    2123, 2126, 2130, 2215, 2220, 2228
group commands:
\group_begin: ..... 172, 191
\group_end: ..... 180

```

\group_insert_after:N ... 3275, 3325

\ior_str_map_inline:Nn ... 2116

H

hbox commands:

\hbox:n ... 222, 2351, 2354,
2429, 2435, 2588, 2595, 3082, 3093
\hbox_overlap_right:n ... 224,
256, 272, 313, 329, 357, 441, 1333, 1528
\hbox_set:Nn .. 1839, 2074, 2421, 2453
\hbox_set:Nw ... 2404
\hbox_set_end: ... 2419
\hbox_unpack:N ... 2540

hook commands:

\hook_gput_code:nnn .. 54, 1731, 1733

I

int commands:

\int_compare:nNnTF ...
... 1795, 1834, 1965, 1992,
2040, 2083, 2512, 2613, 2896, 2924
\int_const:Nn ... 454, 1811,
1905, 1987, 2089, 2260, 2787, 2962
\int_eval:n 474, 484, 630, 639, 652,
654, 658, 671, 2637, 2641, 2874,
2899, 2906, 2919, 3111, 3119, 3124
\int_gincr:N ... 198,
364, 1597, 1642, 1986, 2259, 2330,
2361, 2438, 2961, 2998, 3011, 3031
\int_gset:Nn ... 173, 192, 2501
\int_gset_eq:NN 181, 2362, 2439, 3012
\int_if_exist:NTF ... 1976
\int_if_odd:nTF ... 2424
\int_max:nn ... 2091
\int_new:N ... 164,
165, 411, 449, 1623, 1937, 2256,
2342, 2373, 2375, 2957, 3008, 3024
\int_set_eq:NN ... 169, 188, 2513
\int_step_function:nnnN ... 656
\int_use:N ...
. 366, 397, 583, 592, 740, 768, 817,
823, 824, 878, 879, 888, 912, 1600,
1606, 1613, 1645, 1653, 1796, 1835,
1848, 1906, 1966, 1979, 1991, 1993,
2084, 2092, 2265, 2332, 2337, 2365,
2372, 2443, 2544, 2797, 2807, 2967,
3000, 3005, 3015, 3023, 3036, 3047
\int_value:w ...
... 2666, 2701, 2845, 2856, 2874
\int_zero:N ... 1773, 1923, 2025, 2190

ior commands:

\ior_close:N ... 2135
\ior_if_eof:NTF ... 2109
\ior_map_break: ... 2131
\ior_open:Nn ... 2108

K

kernel internal commands:

__kernel_backend_align_begin: ..
... 71, 71, 209, 233, 248
__kernel_backend_align_end: ...
... 71, 77, 223, 241, 255
__kernel_backend_first_shipout:n
... 49, 53, 56, 58, 68, 580, 3167
\g_kernel_backend_header_bool ..
... 66, 578
__kernel_backend_literal:n ...
... 46, 46, 47, 48,
61, 64, 69, 73, 80, 83, 85, 151, 154,
156, 158, 162, 338, 351, 498, 504,
528, 533, 600, 736, 780, 932, 937,
943, 948, 999, 1025, 1151, 1157,
1452, 1459, 1465, 1525, 1530, 1750,
1940, 1978, 1988, 2209, 2224, 2951,
3049, 3111, 3115, 3120, 3125, 3169
__kernel_backend_literal_page:n
... 99, 99,
109, 153, 153, 2945, 2947, 3130, 3132
__kernel_backend_literal_pdf:n .
... 88, 88, 98, 150, 150,
152, 264, 321, 1342, 3269, 3286, 3319
__kernel_backend_literal_-
postscript:n ... 60,
60, 62, 74, 75, 79, 210, 211, 213,
214, 222, 234, 249, 1147, 2615, 2627
__kernel_backend_literal_svg:n .
. 161, 161, 163, 168, 179, 187, 197,
365, 367, 384, 762, 1536, 1711, 1722
__kernel_backend_matrix:n
... 137, 137, 147, 286, 307, 1442
__kernel_backend_postscript:n ..
... 63, 63, 65,
500, 1002, 1004, 1006, 1010, 2250,
2307, 2322, 2351, 2357, 2397, 2429,
2436, 2440, 2454, 2482, 2526, 2533,
2539, 2547, 2554, 2588, 2595, 3223
__kernel_backend_scope:n
... 166, 195, 200, 394,
399, 1065, 1543, 1588, 1590, 1610,
1650, 1672, 1684, 1686, 1688, 1690,
1692, 1694, 1696, 1698, 1701, 3338
__kernel_backend_scope_begin: ..
82, 82, 119, 119, 155, 155, 166, 166,
208, 232, 247, 263, 280, 306, 320,
337, 350, 1348, 1520, 1538, 1542, 1709
__kernel_backend_scope_begin:n .
... 166, 185, 194, 386, 414, 427

<code>__kernel_backend_scope_end:</code> 82 , 84 , 119 , 128 , 155 , 157 , 166 , 175 , 225 , 243 , 257 , 273 , 300 , 314 , 330 , 346 , 358 , 409 , 423 , 442 , 1349 , 1532 , 1539 , 1545 , 1723	<code>__opacity_backend_fillstroke:nn</code> 3293
<code>\g__kernel_backend_scope_int</code> . . . 164 , 171 , 173 , 178 , 182 , 190 , 192 , 198	<code>__opacity_backend_reset:</code> 3256 , 3275 , 3283 , 3325
<code>\l__kernel_backend_scope_int</code> 164 , 170 , 183 , 189	<code>__opacity_backend_select:n</code> 3197 , 3197 , 3256 , 3256 , 3331 , 3331
<code>\g__kernel_clip_path_int</code> 362 , 1597 , 1600 , 1613 , 1642 , 1645 , 1653	<code>__opacity_backend_select_aux:n</code> 3197 , 3199 , 3202 , 3256 , 3258 , 3261 , 3281 , 3308
<code>__kernel_color_backend_stack_- init:Nnn</code> 452 , 452 , 3248	<code>\c__opacity_backend_stack_int</code> 3243 , 3272 , 3290 , 3322
<code>__kernel_color_backend_stack_- pop:n</code> 466 , 476 , 524 , 3290	<code>__opacity_backend_stroke:n</code> 3197 , 3214 , 3293 , 3299 , 3331 , 3335
<code>__kernel_color_backend_stack_- push:nn</code> 466 , 466 , 521 , 966 , 978 , 3272 , 3322	<code>\l__opacity_backend_stroke_tl</code> 3254 , 3264 , 3297 , 3311
<code>__kernel_dependency_version_- check:Nn</code> 1	
<code>__kernel_dependency_version_- check:nn</code> 27 , 29	
<code>__kernel_file_name_quote:n</code> 1866 , 1892	
<code>__kernel_kern:n</code> 2356 , 2358 , 2587 , 2591 , 2594 , 2598 , 3081 , 3089 , 3092 , 3108	
	P
	pdf commands:
	<code>\pdf_object_if_exist:nTF</code> 832 , 898 , 916
	<code>\pdf_object_new:n</code> 823 , 834 , 878 , 900 , 918
	<code>\pdf_object_ref:n</code> 780 , 847 , 911 , 926 , 944 , 949
	<code>\pdf_object_ref_last:</code> 800 , 825 , 828 , 884
	<code>\pdf_object_unnamed_write:nn</code> 807 , 854 , 910 , 925
	<code>\pdf_object_write:nnn</code> 824 , 835 , 879 , 901 , 919
	pdf internal commands:
	<code>__pdf_backend:n</code> 2950 , 2950 , 2952 , 2954 , 2956 , 2976 , 2981 , 2990 , 3013 , 3032 , 3045 , 3052 , 3084 , 3085 , 3095
	<code>__pdf_backend_annotation:nnnn</code> 2343 , 2343 , 2651 , 2651 , 3009 , 3009 , 3135 , 3135
	<code>__pdf_backend_annotation_- aux:nnnn</code> 2345 , 2348
	<code>\g__pdf_backend_annotation_int</code> 2342 , 2362 , 2372 , 3008 , 3012 , 3023
	<code>__pdf_backend_annotation_last:</code> 2371 , 2371 , 2664 , 2664 , 3022 , 3022 , 3136 , 3136
	<code>__pdf_backend_bdc:nn</code> 2645 , 2645 , 2944 , 2944 , 3129 , 3129 , 3161 , 3161
	<code>__pdf_backend_catalog_gput:nn</code> 2252 , 2252 , 2757 , 2757 , 2953 , 2953 , 3145 , 3145
	<code>__pdf_backend_compress_objects:n</code> 2611 , 2623 , 2865 , 2876 , 3110 , 3112 , 3155 , 3156
	<code>__pdf_backend_compresslevel:n</code> 2611 , 2611 , 2865 , 2865 , 3110 , 3110 , 3155 , 3155

\l__pdf_backend_content_box	2340, 2404, 2428, 2431, 2433, 2462, 2473	__pdf_backend_link_outerbox:n	2382, 2422, 2480
__pdf_backend_destination:nn	2552, 2552, 2720, 2720, 3050, 3050, 3143, 3143	\g__pdf_backend_link_sf_int	2375, 2501, 2512, 2513
__pdf_backend_destination:nnnn	2552, 2578, 2720, 2743, 3050, 3072, 3143, 3144	__pdf_backend_link_sf_restore:	2382, 2405, 2448, 2508
__pdf_backend_destination_-aux:nnnn	2552, 2580, 2583, 3050, 3074, 3077	__pdf_backend_link_sf_save:	2382, 2400, 2418, 2499
__pdf_backend_emc:	2645, 2647, 2944, 2946, 3129, 3131, 3161, 3162	\l__pdf_backend_model_box	2341, 2421, 2453, 2461, 2472, 2487, 2489
__pdf_backend_info_gput:nn	2252, 2254, 2757, 2767, 2953, 2955, 3145, 3146	__pdf_backend_objcompresslevel:n	2865, 2879, 2880, 2882
__pdf_backend_link:nw	2382	\g__pdf_backend_object_int	2256, 2259, 2262, 2330, 2332, 2337, 2361, 2362, 2365, 2438, 2439, 2957, 2961, 2964, 2998, 3000, 3005, 3011, 3012, 3015
__pdf_backend_link_aux:nw	2382	__pdf_backend_object_last:	2336, 2336, 2843, 2843, 3004, 3004, 3147, 3153
__pdf_backend_link_begin:n	3025, 3026, 3028, 3029	__pdf_backend_object_new:n	2257, 2257, 2778, 2778, 2959, 2959, 3147
__pdf_backend_link_begin:nnnw	2675, 2676, 2678, 2679, 3137, 3139	__pdf_backend_object_new:nn	3147
__pdf_backend_link_begin:nw	2384, 2388, 2389	__pdf_backend_object_now:nn	2328, 2328, 2335, 2832, 2832, 2842, 2996, 2996, 3003, 3147, 3151, 3152
__pdf_backend_link_begin_aux:nw	2392, 2394	\g__pdf_backend_object_prop	2777, 2957
__pdf_backend_link_begin_-goto:nnw	2382, 2382, 2675, 2675, 3025, 3025, 3137, 3137	__pdf_backend_object_ref:n	2257, 2264, 2269, 2778, 2796, 2959, 2966, 2971, 3147, 3148
__pdf_backend_link_begin_-user:nnw	2382, 2387, 2675, 2677, 3025, 3027, 3137, 3138	__pdf_backend_object_write:nn	2798, 2809, 2811, 2840, 3147
\g__pdf_backend_link_bool	2377, 2391, 2396, 2411, 2449	__pdf_backend_object_write:nnn	2266, 2266, 2272, 2798, 2798, 2829, 2968, 2968, 2973, 3147, 3149, 3150
\g__pdf_backend_link_dict_tl	2374, 2399, 2444	__pdf_backend_object_write_-array:nn	2266, 2290, 2968, 2974
__pdf_backend_link_end:	2382, 2409, 2675, 2690, 3025, 3044, 3137, 3140	__pdf_backend_object_write_-aux:nnn	2266, 2268, 2273, 2331
__pdf_backend_link_end_aux:	2382, 2412, 2414	__pdf_backend_object_write_-dict:nn	2266, 2295, 2968, 2979
\g__pdf_backend_link_int	2373, 2439, 2443, 2544, 3024, 3031, 3036, 3047	__pdf_backend_object_write_-fstream:nn	2266, 2300, 2968, 2984
__pdf_backend_link_last:	2543, 2543, 2699, 2699, 3046, 3046, 3141, 3141	__pdf_backend_object_write_-fstream:nnn	2303, 2305
__pdf_backend_link_margin:n	2545, 2545, 2710, 2710, 3048, 3048, 3142, 3142	__pdf_backend_object_write_-stream:nn	2266, 2315, 2968, 2986
\g__pdf_backend_link_math_bool	2376, 2402, 2403, 2406, 2416	__pdf_backend_object_write_-stream:nnn	2266, 2318, 2320
__pdf_backend_link_minima:	2382, 2420, 2451	__pdf_backend_object_write_-stream:nnnn	2968, 2985, 2987, 2988
		__pdf_backend_pageobject_ref:n	2338, 2338,

2854 , 2854 , 3006 , 3006 , 3147 , 3154	pdf.linkdp.pad	2382 , 3400
_pdf_backend_pagesize_gset:nn	pdf.linkht.pad	2382 , 3400
.. 3165 , 3165 , 3184 , 3184 , 3191 , 3191	pdf.linkmargin	3400
_pdf_backend_pdfmark:n	pdf.llx	2382 , 3403
2249 , 2251 , 2253 , 2255 , 2275 , 2292 , 2297 , 2363 , 2555 , 2599 , 2646 , 2648	pdf.lly	2382 , 3403
_pdf_backend_version_major:	pdf.originx	3474
... 2637 , 2643 , 2643 , 2921 , 2921 , 3119 , 3120 , 3127 , 3127 , 3159 , 3159	pdf.originy	3474
_pdf_backend_version_major_-gset:n	pdf.outerbox	2382 , 3716
2893 , 2893 , 3117 , 3117 , 3157 , 3157	pdf.pdfmark	3716
_pdf_backend_version_minor:	pdf.pdfmark.dict	3716
... 2641 , 2643 , 2644 , 2921 , 2934 , 3124 , 3125 , 3127 , 3128 , 3159 , 3160	pdf.pdfmark.good	3716
_pdf_backend_version_minor_-gset:n	pdf.pt.dvi	3396
2893 , 2910 , 3117 , 3122 , 3157 , 3158	pdf.rect	3403
_pdf_breaklink_pdfmark_tl	pdf.rect.ht	3396
... 2378 , 2446 , 2538	pdf.rightboundary	3474
_pdf_breaklink_postscript:n	pdf.save.linkll	3403
... 2380 , 2380 , 2430 , 2432 , 2539	pdf.save.linkur	3403
_pdf_breaklink_usebox:N	pdf.save.ll	3403
... 2381 , 2381 , 2431 , 2540	pdf.save.ur	3403
_pdf_exp_not_i:nn	pdf.tmpa	3439
... 2798 , 2819 , 2824 , 2830	pdf.tmpb	3439
_pdf_exp_not_ii:nn	pdf.tmpc	3439
... 2798 , 2820 , 2825 , 2831	pdf.tmpd	3439
_pdf_internal_box	pdf.urx	3403
2246	pdf.ury	2382 , 3403
pdf.baselineskip	pdfmanagement commands:	
2382 , 3716	_pdfmanagement_add:nnn	
pdf.bordertracking 797 , 3251 , 3265 , 3312 , 3315	
3474	_pdfmanagement_if_active_p:	
pdf.bordertracking.begin 792 , 793 , 3244 , 3245 , 3278 , 3279	
3474	peek commands:	
pdf.bordertracking.continue	_peek_meaning:NtF	2151 , 2154
3474	_peek_remove_spaces:n	2149
pdf.bordertracking.end	prg commands:	
3474	_prg_replicate:nn	
pdf.bordertracking.endpage 177 , 628 , 649 , 659 , 860	
3474	prop commands:	
pdf.breaklink	_prop_gput:Nnn	586 , 827
3612	_prop_if_in:NnTF	563
pdf.breaklink.write	_prop_item:Nn	566
3612	_prop_new:N	544 , 2777 , 2958
pdf.brokenlink.dict	\ProvidesExplFile	2
3474		
pdf.brokenlink.rect		
3474		
pdf.brokenlink.skip		
3474		
pdf.count		
3612		
pdf.currentrect		
3612		
pdf.cvs		
3396		
pdf.dest.anchor		
3439		
pdf.dest.point		
3439		
pdf.dest.x		
3439		
pdf.dest.y		
3439		
pdf.dest2device		
3439		
pdf.dev.x		
3439		
pdf.dev.y		
3439		
pdf.dvi.pt		
3396		
pdf.globaldict		
3393		
pdf.leftboundary		
3474		
pdf.link.dict		
2382		

Q

quark commands:	
_quark_if_recursion_tail_stop:n	562
_q_recursion_stop	555
_q_recursion_tail	554

S

scan commands:	
_scan_stop:	122 , 131 , 484 , 2179 , 2182 , 2693 , 2718 , 2741 , 2755 , 2874 , 2891 , 2899 , 2906 , 2919

<code>\tl_if_empty_p:N</code>	1821, 2006				U
<code>\tl_new:N</code>	507,			use commands:	
	508, 1586, 1770, 2374, 2378, 3254, 3255			<code>\use:N</code>	43, 2288, 2970, 2999
<code>\tl_put_right:Nn</code>	2520			<code>\use:n</code>	58, 795, 821, 876,
<code>\tl_set:Nn</code>	509, 510, 519,				1032, 1042, 1055, 1288, 1412, 1477,
	520, 965, 977, 1775, 1792, 1883,				1489, 1501, 1661, 2056, 2140, 2162
	2379, 2538, 3263, 3264, 3310, 3311			<code>\use_none:n</code>	1678, 2516
<code>\tl_to_str:n</code>	2143, 2165, 2261,				V
	2265, 2788, 2797, 2808, 2963, 2967			<code>\value</code>	2424
<code>\tl_use:N</code>	727, 840			vbox commands:	
token commands:				<code>\vbox_set:Nn</code>	2524
<code>\c_math_toggle_token</code>	2407, 2417			<code>\vbox_to_zero:n</code>	2585, 2592, 3079, 3090
				<code>\vbox_unpack_drop:N</code>	2532