

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}{2024-01-04}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2024-01-04}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2024-01-04}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2024-01-04}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2024-01-04}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2024-01-04}{ }
25   {L3 backend support: XeTeX}
26 </xetex>
```

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

```
27 \cs_if_exist:NTF \_kernel\_dependency\_version\_check:Nn
28   {
29     \_kernel\_dependency\_version\_check:Nn {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.

```

166 \cs_new_protected:Npn \_kernel_backend_scope_begin:
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 T_EX 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 \end{dvisvgm}
445 \end{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:
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 }
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.)

```

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 ~ exch ~ pop ~ } ~
677     { ~
678       2 ~ index ~ 1 ~ index ~ gt ~
679       { ~ exch ~ pop ~ exch ~ 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/XYTeX`, 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/XYTeX`.

```

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
 __color_backend_devicen_init:w work.

```

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_qTeX 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 Lua_T_E_X 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 = 1.P' '^0
1091 local digit16 = 1.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 = 1.Cs(octet * octet * octet * -1 * 1.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 % If no l3color named color with this name is known, check for defined xcolor colors
1124     local l3color_prop = token.get_macro(string.format('l__color_named_%s_prop', value))
1125     if l3color_prop == nil or l3color_prop == '' then
1126         local legacy_color_macro = token.create(string.format('\\color@%s', value))
1127         if legacy_color_macro.cmdname ~= 'undefined_cs' then
1128             token.put_next(legacy_color_macro)
1129             return token.scan_argument()
1130         end
1131     end
1132
1133     tex.runtoks(function()
1134         token.get_next()
1135         color_export[6] = value
1136         tex.sprint(-2, color_export)
1137     end)
1138     local list = token.scan_list()
1139     if not list.head or list.head.next
1140         or list.head.subtype ~= node.subtype'pdf_colorstack' then
1141         error'Unexpected backend behavior'
1142     end
1143     local cmd = list.head.data
1144     node.free(list)
1145     return cmd
1146 end, 'l3color')
1147 end
1148 </lua>
1149 <*luatex>
1150 <*package>
1151 \lua_load_module:n {l3backend-luatex}
1152 </package>
1153 </luatex>

```

4 l3backend-draw implementation

```

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

```

4.1 dvips backend

```

1156 <*dvips>

```

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

`__draw_backend_literal:e`

```

1157 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_postscript:n
1158 \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 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.

```

1159 \cs_new_protected:Npn \__draw_backend_begin:
1160 {
1161   \__kernel_backend_literal:n { ps::[begin] }
1162   \__draw_backend_literal:n { @beginspecial }
1163 }
1164 \cs_new_protected:Npn \__draw_backend_end:
1165 {
1166   \__draw_backend_literal:n { @endspecial }
1167   \__kernel_backend_literal:n { ps::[end] }
1168 }

```

(End 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.

```

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

```

(End 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.

```

1173 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1174 {
1175   \__draw_backend_literal:e
1176   {
1177     \dim_to_decimal_in_bp:n {#1} ~
1178     \dim_to_decimal_in_bp:n {#2} ~ moveto
1179   }
1180 }
1181 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1182 {
1183   \__draw_backend_literal:e
1184   {
1185     \dim_to_decimal_in_bp:n {#1} ~

```



```

1186         \dim_to_decimal_in_bp:n {#2} ~ lineto
1187     }
1188 }
1189 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1190 {
1191     \__draw_backend_literal:e
1192     {
1193         \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1194         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1195         moveto~dup-0~rlineto~exch-0~exch~rlineto~neg-0~rlineto~closepath
1196     }
1197 }
1198 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1199 {
1200     \__draw_backend_literal:e
1201     {
1202         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1203         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1204         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1205         curveto
1206     }
1207 }

```

(End 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: 1208 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
\g__draw_draw_eor_bool 1209 { \bool_gset_true:N \g__draw_draw_eor_bool }
1210 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1211 { \bool_gset_false:N \g__draw_draw_eor_bool }
1212 \bool_new:N \g__draw_draw_eor_bool

```

(End 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 also desirable to have the clip keyword after a stroke or fill. To achieve those outcomes, there is some work to do. For color, the stroke color is simple but the fill one has to be inserted by hand. For clipping, the required ordering is achieved using a T_EX switch. All of the operations end with a new path instruction as they do not terminate (again in contrast to PDF).

```

\__draw_backend_stroke: 1213 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_closestroke: 1214 { \__draw_backend_literal:n { closepath } }
\__draw_backend_fill: 1215 \cs_new_protected:Npn \__draw_backend_stroke:
\__draw_backend_fillstroke: 1216 {
\__draw_backend_clip: 1217     \__draw_backend_literal:n { gsave }
\__draw_backend_discardpath: 1218     \__draw_backend_literal:n { color.sc }
\g__draw_draw_clip_bool 1219     \__draw_backend_literal:n { stroke }
1220     \__draw_backend_literal:n { grestore }
1221     \bool_if:NT \g__draw_draw_clip_bool
1222     {
1223         \__draw_backend_literal:e
1224         {
1225             \bool_if:NT \g__draw_draw_eor_bool { eo }

```

```

1226         clip
1227     }
1228 }
1229 \__draw_backend_literal:n { newpath }
1230 \bool_gset_false:N \g__draw_draw_clip_bool
1231 }
1232 \cs_new_protected:Npn \__draw_backend_closestroke:
1233 {
1234     \__draw_backend_closepath:
1235     \__draw_backend_stroke:
1236 }
1237 \cs_new_protected:Npn \__draw_backend_fill:
1238 {
1239     \__draw_backend_literal:e
1240     {
1241         \bool_if:NT \g__draw_draw_eor_bool { eo }
1242         fill
1243     }
1244     \bool_if:NT \g__draw_draw_clip_bool
1245     {
1246         \__draw_backend_literal:e
1247         {
1248             \bool_if:NT \g__draw_draw_eor_bool { eo }
1249             clip
1250         }
1251     }
1252     \__draw_backend_literal:n { newpath }
1253     \bool_gset_false:N \g__draw_draw_clip_bool
1254 }
1255 \cs_new_protected:Npn \__draw_backend_fillstroke:
1256 {
1257     \__draw_backend_literal:e
1258     {
1259         \bool_if:NT \g__draw_draw_eor_bool { eo }
1260         fill
1261     }
1262     \__draw_backend_literal:n { gsave }
1263     \__draw_backend_literal:n { color.sc }
1264     \__draw_backend_literal:n { stroke }
1265     \__draw_backend_literal:n { grestore }
1266     \bool_if:NT \g__draw_draw_clip_bool
1267     {
1268         \__draw_backend_literal:e
1269         {
1270             \bool_if:NT \g__draw_draw_eor_bool { eo }
1271             clip
1272         }
1273     }
1274     \__draw_backend_literal:n { newpath }
1275     \bool_gset_false:N \g__draw_draw_clip_bool
1276 }
1277 \cs_new_protected:Npn \__draw_backend_clip:
1278 { \bool_gset_true:N \g__draw_draw_clip_bool }
1279 \bool_new:N \g__draw_draw_clip_bool

```

```

1280 \cs_new_protected:Npn \__draw_backend_discardpath:
1281 {
1282   \bool_if:NT \g__draw_draw_clip_bool
1283   {
1284     \__draw_backend_literal:e
1285     {
1286       \bool_if:NT \g__draw_draw_eor_bool { eo }
1287       clip
1288     }
1289   }
1290   \__draw_backend_literal:n { newpath }
1291   \bool_gset_false:N \g__draw_draw_clip_bool
1292 }

```

(End of definition for __draw_backend_closepath: and others.)

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

```

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

```

(End 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/X_YTeX). Thus we take the shortest path available and simply dump the matrix as given.

```

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

```

(End of definition for __draw_backend_cm:nmmm.)

__draw_backend_box_use:Nmmmm

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

```

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

```

(End of definition for __draw_backend_box_use:Nmmmm.)

1350 </dvips>

4.2 LuaTeX, pdfTeX, dvipdfmx and X_YTeX

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

```

1351 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

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

`_draw_backend_literal:e` 1352 `\cs_new_eq:NN _draw_backend_literal:n _kernel_backend_literal_pdf:n`
1353 `\cs_generate_variant:Nn _draw_backend_literal:n { e }`

(End of definition for _draw_backend_literal:n.)

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

`_draw_backend_end:` 1354 `\cs_new_protected:Npn _draw_backend_begin:`
1355 `{ _draw_backend_scope_begin: }`
1356 `\cs_new_protected:Npn _draw_backend_end:`
1357 `{ _draw_backend_scope_end: }`

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

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

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

(End 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

`_draw_backend_lineto:nn` to convert to bp.

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

(End 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: 1389 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
\g__draw_draw_eor_bool 1390 { \bool_gset_true:N \g__draw_draw_eor_bool }
1391 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1392 { \bool_gset_false:N \g__draw_draw_eor_bool }
1393 \bool_new:N \g__draw_draw_eor_bool

```

(End of definition for `_draw_backend_evenodd_rule:`, `_draw_backend_nonzero_rule:`, and `\g__draw_draw_eor_bool`.)

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

```

\__draw_backend_stroke: 1394 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_closestroke: 1395 { \__draw_backend_literal:n { h } }
\__draw_backend_fill: 1396 \cs_new_protected:Npn \__draw_backend_stroke:
\__draw_backend_fillstroke: 1397 { \__draw_backend_literal:n { S } }
\__draw_backend_clip: 1398 \cs_new_protected:Npn \__draw_backend_closestroke:
\__draw_backend_discardpath: 1399 { \__draw_backend_literal:n { s } }
1400 \cs_new_protected:Npn \__draw_backend_fill:
1401 {
1402   \__draw_backend_literal:e
1403   { f \bool_if:NT \g__draw_draw_eor_bool * }
1404 }
1405 \cs_new_protected:Npn \__draw_backend_fillstroke:
1406 {
1407   \__draw_backend_literal:e
1408   { B \bool_if:NT \g__draw_draw_eor_bool * }
1409 }
1410 \cs_new_protected:Npn \__draw_backend_clip:
1411 {
1412   \__draw_backend_literal:e
1413   { W \bool_if:NT \g__draw_draw_eor_bool * }
1414 }
1415 \cs_new_protected:Npn \__draw_backend_discardpath:
1416 { \__draw_backend_literal:n { n } }

```

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

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

```

\__draw_backend_dash:n 1417 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
\__draw_backend_linewidth:n 1418 {
\__draw_backend_miterlimit:n 1419   \__draw_backend_literal:e
\__draw_backend_cap_butt: 1420   {
\__draw_backend_cap_round: 1421     [
\__draw_backend_cap_rectangle: 1422       \exp_args:Nf \use:n
\__draw_backend_join_miter: 1423       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
\__draw_backend_join_round: 1424     ] ~
\__draw_backend_join_bevel: 1425     \dim_to_decimal_in_bp:n {#2} ~ d
1426   }
1427 }
1428 \cs_new:Npn \__draw_backend_dash:n #1
1429 { ~ \dim_to_decimal_in_bp:n {#1} }
1430 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1431 {
1432   \__draw_backend_literal:e

```

```

1433     { \dim_to_decimal_in_bp:n {#1} ~ w }
1434   }
1435   \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1436     { \__draw_backend_literal:e { #1 ~ M } }
1437   \cs_new_protected:Npn \__draw_backend_cap_but:
1438     { \__draw_backend_literal:n { 0 ~ J } }
1439   \cs_new_protected:Npn \__draw_backend_cap_round:
1440     { \__draw_backend_literal:n { 1 ~ J } }
1441   \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1442     { \__draw_backend_literal:n { 2 ~ J } }
1443   \cs_new_protected:Npn \__draw_backend_join_miter:
1444     { \__draw_backend_literal:n { 0 ~ j } }
1445   \cs_new_protected:Npn \__draw_backend_join_round:
1446     { \__draw_backend_literal:n { 1 ~ j } }
1447   \cs_new_protected:Npn \__draw_backend_join_bevel:
1448     { \__draw_backend_literal:n { 2 ~ j } }

```

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

```

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

```

```

1477      x:rotate~
1478      \fp_compare:nNnTF {#4} = \c_zero_fp
1479      { 0 }
1480      { \fp_eval:n { round ( -#4 , 5 ) } }
1481    }
1482  }
1483 </dvipdfmx | xetex>

```

(End 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.

```

1484 <*dvipdfmx | xetex>
1485 \cs_new_protected:Npn \_draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1486 {
1487   \use:e
1488   {
1489     \_draw_backend_cm_decompose_auxi:nnnnN
1490     { \fp_eval:n { (#1 + #4) / 2 } }
1491     { \fp_eval:n { (#1 - #4) / 2 } }
1492     { \fp_eval:n { (#3 + #2) / 2 } }
1493     { \fp_eval:n { (#3 - #2) / 2 } }
1494   }
1495   #5
1496 }
1497 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1498 {
1499   \use:e

```



```

1500 {
1501   \__draw_backend_cm_decompose_auxii:nnnnN
1502   { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1503   { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1504   { \fp_eval:n { atand ( #3 , #2 ) } }
1505   { \fp_eval:n { atand ( #4 , #1 ) } }
1506 }
1507 #5
1508 }
1509 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1510 {
1511   \use:e
1512   {
1513     \__draw_backend_cm_decompose_auxiii:nnnnN
1514     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1515     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1516     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1517     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1518   }
1519   #5
1520 }
1521 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1522 {
1523   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1524   { #5 {#1} {#2} {#3} {#4} }
1525   { #5 {#1} {#3} {#2} {#4} }
1526 }
1527 </dviPDFmx | 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.

```

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

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1544 < /dviPDFmx | luatex | pdftex | xetex >

```

4.3 dvisvgm backend

1545 $\langle *dvisvgm \rangle$

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

`_draw_backend_literal:e` 1546 `\cs_new_eq:NN _draw_backend_literal:n _kernel_backend_literal_svg:n`
 1547 `\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:` 1548 `\cs_new_eq:NN _draw_backend_scope_begin: _kernel_backend_scope_begin:`
 1549 `\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

1550 `\cs_new_protected:Npn _draw_backend_begin:`
 1551 `{`
 1552 `_kernel_backend_scope_begin:`
 1553 `_kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }`
 1554 `}`
 1555 `\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` 1556 `\cs_new_protected:Npn _draw_backend_moveto:nn #1#2`
 1557 `{`
 1558 `_draw_backend_add_to_path:n`
 1559 `{ M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }`
 1560 `}`
 1561 `\cs_new_protected:Npn _draw_backend_lineto:nn #1#2`
 1562 `{`
 1563 `_draw_backend_add_to_path:n`
 1564 `{ L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }`
 1565 `}`
 1566 `\cs_new_protected:Npn _draw_backend_rectangle:nnnn #1#2#3#4`
 1567 `{`
 1568 `_draw_backend_add_to_path:n`
 1569 `{`
 1570 `M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}`
 1571 `h ~ \dim_to_decimal:n {#3} ~`
 1572 `v ~ \dim_to_decimal:n {#4} ~`
 1573 `h ~ \dim_to_decimal:n { -#3 } ~`
 1574 `Z`
 1575 `}`
 1576 `}`
 1577 `\cs_new_protected:Npn _draw_backend_curveto:nnnnnn #1#2#3#4#5#6`
 1578 `{`

```

1579 \__draw_backend_add_to_path:n
1580 {
1581   C ~
1582   \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1583   \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1584   \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1585 }
1586 }
1587 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1588 {
1589   \tl_gset:Nx \g__draw_backend_path_tl
1590   {
1591     \g__draw_backend_path_tl
1592     \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1593     #1
1594   }
1595 }
1596 \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:
1597 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1598 { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1599 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1600 { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

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

```

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

```

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

```

1601 \cs_new_protected:Npn \__draw_backend_closepath:
1602 { \__draw_backend_add_to_path:n { Z } }
1603 \cs_new_protected:Npn \__draw_backend_path:n #1
1604 {
1605   \bool_if:NTF \g__draw_draw_clip_bool
1606   {
1607     \int_gincr:N \g__kernel_clip_path_int
1608     \__draw_backend_literal:e
1609     {
1610       < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1611       { ?nl }
1612       <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1613       < /clipPath > { ? nl }
1614       <
1615       use~xlink:href =
1616       "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1617       #1
1618     } />
1619   }
1620   \__kernel_backend_scope:e

```

```

1621         {
1622             clip-path =
1623                 "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1624         }
1625     }
1626     {
1627         \__draw_backend_literal:e
1628         { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1629     }
1630     \tl_gclear:N \g__draw_backend_path_tl
1631     \bool_gset_false:N \g__draw_draw_clip_bool
1632 }
1633 \int_new:N \g__draw_backend_path_int
1634 \cs_new_protected:Npn \__draw_backend_stroke:
1635     { \__draw_backend_path:n { style="fill:none" } }
1636 \cs_new_protected:Npn \__draw_backend_closestroke:
1637     {
1638         \__draw_backend_closepath:
1639         \__draw_backend_stroke:
1640     }
1641 \cs_new_protected:Npn \__draw_backend_fill:
1642     { \__draw_backend_path:n { style="stroke:none" } }
1643 \cs_new_protected:Npn \__draw_backend_fillstroke:
1644     { \__draw_backend_path:n { } }
1645 \cs_new_protected:Npn \__draw_backend_clip:
1646     { \bool_gset_true:N \g__draw_draw_clip_bool }
1647 \bool_new:N \g__draw_draw_clip_bool
1648 \cs_new_protected:Npn \__draw_backend_discardpath:
1649     {
1650         \bool_if:NT \g__draw_draw_clip_bool
1651         {
1652             \int_gincr:N \g__kernel_clip_path_int
1653             \__draw_backend_literal:e
1654             {
1655                 < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1656                 { ?nl }
1657                 <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1658                 < /clipPath >
1659             }
1660             \__kernel_backend_scope:e
1661             {
1662                 clip-path =
1663                     "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1664             }
1665         }
1666         \tl_gclear:N \g__draw_path_tl
1667         \bool_gset_false:N \g__draw_draw_clip_bool
1668     }

```

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_dash_aux:nn
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_but:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:

```

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

```

1669 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2

```

```

1670 {
1671   \use:e
1672   {
1673     \__draw_backend_dash_aux:nn
1674     { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1675     { \dim_to_decimal:n {#2} }
1676   }
1677 }
1678 \cs_new:Npn \__draw_backend_dash:n #1
1679 { , \dim_to_decimal_in_bp:n {#1} }
1680 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1681 {
1682   \__kernel_backend_scope:e
1683   {
1684     stroke-dasharray =
1685     "
1686     \tl_if_empty:nTF {#1}
1687     { none }
1688     { \use_none:n #1 }
1689     " ~
1690     stroke-offset=" #2 "
1691   }
1692 }
1693 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1694 { \__kernel_backend_scope:e { stroke-width=" \dim_to_decimal:n {#1} " } }
1695 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1696 { \__kernel_backend_scope:e { stroke-miterlimit=" #1 " } }
1697 \cs_new_protected:Npn \__draw_backend_cap_but:
1698 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1699 \cs_new_protected:Npn \__draw_backend_cap_round:
1700 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1701 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1702 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1703 \cs_new_protected:Npn \__draw_backend_join_miter:
1704 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1705 \cs_new_protected:Npn \__draw_backend_join_round:
1706 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1707 \cs_new_protected:Npn \__draw_backend_join_bevel:
1708 { \__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.

```

1709 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1710 {
1711   \__kernel_backend_scope:n
1712   {
1713     transform =
1714     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1715   }
1716 }

```

(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.

```
1717 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
1718 {
1719   \_kernel_backend_scope_begin:
1720   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1721   \_kernel_backend_literal_svg:n
1722   {
1723     < g~
1724     stroke="none"~
1725     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1726     >
1727   }
1728   \box_set_wd:Nn #1 { Opt }
1729   \box_set_ht:Nn #1 { Opt }
1730   \box_set_dp:Nn #1 { Opt }
1731   \box_use:N #1
1732   \_kernel_backend_literal_svg:n { </g> }
1733   \_kernel_backend_scope_end:
1734 }

(End of definition for \_draw_backend_box_use:Nnnnn.)

1735 </dvisvgm>
1736 </package>
```

5 l3backend-graphics implementation

```
1737 <*package>
1738 <@@=graphics>
```

`_graphics_backend_loaded:n`

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

```
1739 \cs_new_protected:Npn \_graphics_backend_loaded:n #1
1740 {
1741   \cs_if_exist:NTF \hook_gput_code:nnn
1742   {
1743     \hook_gput_code:nnn
1744     { package / l3graphics / after }
1745     { backend }
1746     {#1}
1747   }
1748   {#1}
1749 }
```

(End of definition for _graphics_backend_loaded:n.)

5.1 dvips backend

```
1750 <*dvips>
```

`\l_graphics_search_ext_seq`

```
1751 \_graphics_backend_loaded:n
1752 { \seq_set_from_clist:Nn \l_graphics_search_ext_seq { .eps , .ps } }
```

(End of definition for `\l_graphics_search_ext_seq`.)

`_graphics_backend_getbb_eps:n`
`_graphics_backend_getbb_ps:n`

Simply use the generic function.

```
1753 \_graphics_backend_loaded:n
1754 {
1755   \cs_new_eq:NN \_graphics_backend_getbb_eps:n \_graphics_read_bb:n
1756   \cs_new_eq:NN \_graphics_backend_getbb_ps:n \_graphics_read_bb:n
1757 }
```

(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`

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

```
1758 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1759 {
1760   \_kernel_backend_literal:e
1761   {
1762     PSfile = #1 \c_space_tl
1763     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1764     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1765     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1766     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1767   }
1768 }
1769 \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`

```
1770 \_graphics_backend_loaded:n
1771 { \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n }
```

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

```
1772 </dvips>
```

5.2 LuaTeX and pdfTeX backends

```
1773 <*luatex | pdftex>
```

`\l_graphics_search_ext_seq`

```
1774 \_graphics_backend_loaded:n
1775 {
1776   \seq_set_from_clist:Nn
1777   \l_graphics_search_ext_seq
1778   { .pdf , .eps , .ps , .png , .jpg , .jpeg }
1779 }
```

(End of definition for `\l_graphics_search_ext_seq`.)

`\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.

```
1780 \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
1781 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1782 {
1783   \int_zero:N \l__graphics_page_int
1784   \tl_clear:N \l__graphics_pagebox_tl
1785   \tl_set:Ne \l__graphics_attr_tl
1786   {
1787     \tl_if_empty:NF \l__graphics_decodearray_str
1788     { :D \l__graphics_decodearray_str }
1789     \bool_if:NT \l__graphics_interpolate_bool
1790     { :I }
1791     \str_if_empty:NF \l__graphics_pdf_str
1792     { :X \l__graphics_pdf_str }
1793   }
1794   \__graphics_backend_getbb_auxi:n {#1}
1795 }
1796 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1797 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1798 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1799 {
1800   \tl_clear:N \l__graphics_decodearray_str
1801   \bool_set_false:N \l__graphics_interpolate_bool
1802   \tl_set:Ne \l__graphics_attr_tl
1803   {
1804     : \l__graphics_pagebox_tl
1805     \int_compare:nNnT \l__graphics_page_int > 1
1806     { :P \int_use:N \l__graphics_page_int }
1807     \str_if_empty:NF \l__graphics_pdf_str
1808     { :X \l__graphics_pdf_str }
1809   }
1810   \__graphics_backend_getbb_auxi:n {#1}
1811 }
1812 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1813 {
1814   \__graphics_bb_restore:eF { #1 \l__graphics_attr_tl }
1815   { \__graphics_backend_getbb_auxii:n {#1} }
1816 }

```

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.

```

1817 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1818 {
1819   \exp_args:Ne \__graphics_backend_getbb_auxiii:n
1820   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1821   \int_const:cn { c__graphics_ #1 \l__graphics_attr_tl _int }
1822   { \tex_the:D \tex_pdflastximage:D }

```



```

1823 \__graphics_bb_save:e { #1 \l__graphics_attr_tl }
1824 }
1825 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1826 {
1827 \tex_immediate:D \tex_pdfximage:D
1828 \bool_lazy_any:nT
1829 {
1830 { \l__graphics_interpolate_bool }
1831 { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1832 { ! \str_if_empty_p:N \l__graphics_pdf_str }
1833 }
1834 {
1835 attr ~
1836 {
1837 \tl_if_empty:NF \l__graphics_decodearray_str
1838 { /Decode~[ \l__graphics_decodearray_str ] }
1839 \bool_if:NT \l__graphics_interpolate_bool
1840 { /Interpolate~true }
1841 \l__graphics_pdf_str
1842 }
1843 }
1844 \int_compare:nNnT \l__graphics_page_int > 0
1845 { page ~ \int_use:N \l__graphics_page_int }
1846 \tl_if_empty:NF \l__graphics_pagebox_tl
1847 { \l__graphics_pagebox_tl }
1848 {#1}
1849 \hbox_set:Nn \l__graphics_internal_box
1850 { \tex_pdfrefximage:D \tex_pdflastximage:D }
1851 \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1852 \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1853 }
1854 \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.

```

1855 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1856 {
1857 \tex_pdfrefximage:D
1858 \int_use:c { c__graphics_ #1 \l__graphics_attr_tl _int }
1859 }
1860 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1861 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1862 \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

```

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.

```

1863 \sys_if_shell:T
1864 {

```

```

\l__graphics_backend_dir_str
\l__graphics_backend_name_str
\l__graphics_backend_ext_str

```

```

1865 \str_new:N \l__graphics_backend_dir_str
1866 \str_new:N \l__graphics_backend_name_str
1867 \str_new:N \l__graphics_backend_ext_str
1868 \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1869 {
1870   \file_parse_full_name:nNNN {#1}
1871   \l__graphics_backend_dir_str
1872   \l__graphics_backend_name_str
1873   \l__graphics_backend_ext_str
1874   \exp_args:Ne \__graphics_backend_getbb_eps:nn
1875   {
1876     \exp_args:Ne \__kernel_file_name_quote:n
1877     {
1878       \l__graphics_backend_name_str
1879       - \str_tail:N \l__graphics_backend_ext_str
1880       -converted-to.pdf
1881     }
1882   }
1883   {#1}
1884 }
1885 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_backend_getbb_eps:n
1886 \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1887 {
1888   \file_compare_timestamp:nNnT {#2} > {#1}
1889   {
1890     \sys_shell_now:n
1891     { repstopdf ~ #2 ~ #1 }
1892   }
1893   \tl_set:Nn \l__graphics_final_name_str {#1}
1894   \__graphics_backend_getbb_pdf:n {#1}
1895 }
1896 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1897 {
1898   \file_parse_full_name:nNNN {#1}
1899   \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1900   \exp_args:Ne \__graphics_backend_include_pdf:n
1901   {
1902     \exp_args:Ne \__kernel_file_name_quote:n
1903     {
1904       \l__graphics_backend_name_str
1905       - \str_tail:N \l__graphics_backend_ext_str
1906       -converted-to.pdf
1907     }
1908   }
1909 }
1910 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1911 }

```

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

__graphics_backend_get_pagecount:n Simply load and store.

```

1912 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1913 {
1914   \tex_pdfximage:D {#1}

```

```

1915 \int_const:cn { c__graphics_ #1 _pages_int }
1916 { \int_use:N \tex_pdflastximagepages:D }
1917 }

```

(End of definition for __graphics_backend_get_pagecount:n.)

```

1918 </luatex | pdftex>

```

5.3 dvipdfmx backend

```

1919 <*dvipdfmx | xetex>

```

\l_graphics_search_ext_seq

```

1920 \__graphics_backend_loaded:n
1921 {
1922   \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1923   { .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }
1924 }

```

(End of definition for \l_graphics_search_ext_seq.)

__graphics_backend_getbb_eps:n
__graphics_backend_getbb_ps: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

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

```

1925 \__graphics_backend_loaded:n
1926 {
1927   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1928   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
1929 }
1930 <*dvipdfmx>
1931 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1932 {
1933   \int_zero:N \l__graphics_page_int
1934   \tl_clear:N \l__graphics_pagebox_tl
1935   \__graphics_extract_bb:n {#1}
1936 }
1937 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1938 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1939 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
1940 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1941 {
1942   \tl_clear:N \l__graphics_decodearray_str
1943   \bool_set_false:N \l__graphics_interpolate_bool
1944   \__graphics_extract_bb:n {#1}
1945 }
1946 </dvipdfmx>

```

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

\g__graphics_track_int

Used to track the object number associated with each graphic.

```

1947 \int_new:N \g__graphics_track_int

```

(End of definition for \g__graphics_track_int.)

_graphics_backend_include_eps:n
 _graphics_backend_include_ps:n
 _graphics_backend_include_jpg:n
 _graphics_backend_include_jpseg:n
 _graphics_backend_include_pdf:n
 _graphics_backend_include_png:n
 _graphics_backend_include_bmp:n
 _graphics_backend_include_auxi:nn
 _graphics_backend_include_auxii:nnn
 _graphics_backend_include_auxiii:nnn

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.

```

1948 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1949 {
1950   \__kernel_backend_literal:e
1951   {
1952     PSfile = #1 \c_space_tl
1953     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1954     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1955     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1956     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1957   }
1958 }
1959 \cs_new_eq:NN \_graphics_backend_include_ps:n \_graphics_backend_include_eps:n
1960 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1961 { \_graphics_backend_include_auxi:nn {#1} { image } }
1962 \cs_new_eq:NN \_graphics_backend_include_jpeg:n \_graphics_backend_include_jpg:n
1963 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1964 \cs_new_eq:NN \_graphics_backend_include_bmp:n \_graphics_backend_include_jpg:n
1965 {*dvipdfmx}
1966 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1967 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1968 </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.

```

1969 \cs_new_protected:Npn \_graphics_backend_include_auxi:nn #1#2
1970 {
1971   \_graphics_backend_include_auxii:enn
1972   {
1973     \tl_if_empty:NF \l__graphics_pagebox_tl
1974     { : \l__graphics_pagebox_tl }
1975     \int_compare:nNnT \l__graphics_page_int > 1
1976     { :P \int_use:N \l__graphics_page_int }
1977     \tl_if_empty:NF \l__graphics_decodearray_str
1978     { :D \l__graphics_decodearray_str }
1979     \bool_if:NT \l__graphics_interpolate_bool
1980     { :I }
1981   }
1982   {#1} {#2}
1983 }
1984 \cs_new_protected:Npn \_graphics_backend_include_auxii:nnn #1#2#3
1985 {
1986   \int_if_exist:cTF { c__graphics_ #2#1 _int }
1987   {
1988     \__kernel_backend_literal:e
1989     { pdf:usexobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
1990   }
1991   { \_graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1992 }
1993 \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!

```

1994 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1995 {
1996   \int_gincr:N \g__graphics_track_int
1997   \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
1998   \__kernel_backend_literal:e
1999   {
2000     pdf:#3~
2001     @graphic \int_use:c { c__graphics_ #1#2 _int } ~
2002     \int_compare:nNnT \l__graphics_page_int > 1
2003     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2004     \tl_if_empty:NF \l__graphics_pagebox_tl
2005     {
2006       pagebox ~ \l__graphics_pagebox_tl \c_space_tl
2007       bbox ~
2008         \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2009         \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2010         \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2011         \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
2012     }
2013     (#1)
2014     \bool_lazy_or:nnT
2015     { \l__graphics_interpolate_bool }
2016     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
2017     {
2018       <<
2019       \tl_if_empty:NF \l__graphics_decodearray_str
2020       { /Decode~[ \l__graphics_decodearray_str ] }
2021       \bool_if:NT \l__graphics_interpolate_bool
2022       { /Interpolate~true }
2023       >>
2024     }
2025   }
2026 }

```

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

`__graphics_backend_get_pagecount:n`

```

2027 <*dvipdfmx>
2028 \__graphics_backend_loaded:n
2029 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }
2030 </dvipdfmx>

```

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

```

2031 </dvipdfmx | xetex>

```

5.4 X_YTeX backend

```

2032 <*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

```

\__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_auxii:VnN
\__graphics_backend_getbb_auxiii:nNnn
\__graphics_backend_getbb_auxiv:nnNnn
\__graphics_backend_getbb_auxiv:VnNnn
\__graphics_backend_getbb_auxv:nNnn

```

a common core process. The XeTeX primitive omits the text box from the page box specification, so there is also some “trimming” to do here.

```

2033 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2034 {
2035   \int_zero:N \l__graphics_page_int
2036   \tl_clear:N \l__graphics_pagebox_tl
2037   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2038 }
2039 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2040 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
2041 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2042 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2043 {
2044   \tl_clear:N \l__graphics_decodearray_str
2045   \bool_set_false:N \l__graphics_interpolate_bool
2046   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2047 }
2048 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2049 {
2050   \int_compare:nNnTF \l__graphics_page_int > 1
2051     { \__graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2052     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2053 }
2054 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2055 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2056 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2057 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2058 {
2059   \tl_if_empty:NTF \l__graphics_pagebox_tl
2060     { \__graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2061     { \__graphics_backend_getbb_auxv:nNnn {#1} #2 {#3} {#4} }
2062 }
2063 }
2064 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2065 {
2066   \use:e
2067   {
2068     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2069     {
2070       #5
2071       \tl_if_blank:nF {#1}
2072       { \c_space_tl \__graphics_backend_getbb_pagebox:w #1 }
2073     }
2074   }
2075 }
2076 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2077 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2078 {
2079   \__graphics_bb_restore:nF {#1#3}
2080   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2081 }
2082 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2083 {
2084   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }

```

```

2085 \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2086 \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2087 \__graphics_bb_save:n {#1#3}
2088 }
2089 \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`.

```

2090 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2091 {
2092   \tex_XeTeXpdffile:D #1 ~
2093   \int_compare:nNnT \l__graphics_page_int > 0
2094     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2095   \exp_after:wN \__graphics_backend_getbb_pagebox:w \l__graphics_pagebox_tl
2096 }

```

(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.

```

2097 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
2098 {
2099   \int_const:cn { c__graphics_ #1 _pages_int }
2100   {
2101     \int_max:nn
2102       { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2103       { 1 }
2104   }
2105 }

```

(End of definition for __graphics_backend_get_pagecount:n.)

2106 \langle /xetex \rangle

5.5 dvisvgm backend

2107 \langle *dvisvgm \rangle

\l_graphics_search_ext_seq

```

2108 \__graphics_backend_loaded:n
2109 {
2110   \seq_set_from_clist:Nn
2111     \l_graphics_search_ext_seq
2112     { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }
2113 }

```

(End of definition for \l_graphics_search_ext_seq.)

\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.

```

2114 \cs_new_protected:Npn \__graphics_backend_getbb_svg:n #1
2115 {
2116   \__graphics_bb_restore:nF {#1}
2117   {
2118     \ior_open:Nn \l__graphics_internal_ior {#1}
2119     \ior_if_eof:NTF \l__graphics_internal_ior
2120     { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2121     {
2122       \dim_zero:N \l__graphics_llx_dim
2123       \dim_zero:N \l__graphics_lly_dim
2124       \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2125       \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2126       \ior_str_map_inline:Nn \l__graphics_internal_ior
2127       {
2128         \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2129         {
2130           \__graphics_backend_getbb_svg_auxi:nNn
2131             { width } \l__graphics_urx_dim {##1}
2132         }
2133         \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2134         {
2135           \__graphics_backend_getbb_svg_auxi:nNn
2136             { height } \l__graphics_ury_dim {##1}
2137         }
2138         \bool_lazy_and:nnF
2139           { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2140           { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2141           { \ior_map_break: }
2142       }
2143       \__graphics_bb_save:n {#1}
2144     }
2145     \ior_close:N \l__graphics_internal_ior
2146   }
2147 }
2148 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2149 {
2150   \use:e
2151   {
2152     \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxiii:w
2153       ##1 \tl_to_str:n {#1} = ##2 \tl_to_str:n {#1} = ##3
2154     \s__graphics_stop
2155   }
2156   {
2157     \tl_if_blank:nF {##2}
2158     {
2159       \peek_remove_spaces:n
2160       {
2161         \peek_meaning:NTF ' % '
2162         { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2163         {
2164           \peek_meaning:NTF " % "
2165           { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2166           { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2167         }
2168       }
2169     }
2170   }

```



```

2168         }
2169         ##2 \s__graphics_stop
2170     }
2171 }
2172 \use:e
2173 {
2174     \__graphics_backend_getbb_svg_auxii:w #3
2175     \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2176     \s__graphics_stop
2177 }
2178 }
2179 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2180 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop
2181 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2182 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2183 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2184 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2185 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2186 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2187 {
2188     \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2189     \l__graphics_internal_dim #2 bp \scan_stop:
2190     \dim_set_eq:NN #1 \l__graphics_internal_dim
2191 }
2192 \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.

```

2193 \__graphics_backend_loaded:n
2194 {
2195     \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2196     \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
2197 }

```

(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.

```

2198 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2199 {
2200     \int_zero:N \l__graphics_page_int
2201     \tl_clear:N \l__graphics_pagebox_tl
2202     \__graphics_extract_bb:n {#1}
2203 }
2204 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2205 \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

```

2206 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2207 {
2208     \tl_clear:N \l__graphics_decodearray_str
2209     \bool_set_false:N \l__graphics_interpolate_bool

```

```

2210     \__graphics_extract_bb:n {#1}
2211 }

(End of definition for \__graphics_backend_getbb_pdf:n.)

```

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

```

\__graphics_backend_include_eps:n
\__graphics_backend_include_ps:n
\__graphics_backend_include_pdf:n
\__graphics_backend_include:n
2212 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2213 { \__graphics_backend_include:nn { PSfile } {#1} }
2214 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
2215 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2216 { \__graphics_backend_include:nn { pdffile } {#1} }
2217 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
2218 {
2219     \__kernel_backend_literal:e
2220     {
2221         #1 = #2 \c_space_tl
2222         llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2223         lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2224         urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2225         ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2226     }
2227 }

```

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

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.

```

2228 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2229 {
2230     \box_move_up:nn { \l__graphics_ury_dim }
2231     {
2232         \hbox:n
2233         {
2234             \__kernel_backend_literal:e
2235             {
2236                 dvisvgm:img~
2237                 \dim_to_decimal:n { \l__graphics_urx_dim } ~
2238                 \dim_to_decimal:n { \l__graphics_ury_dim } ~
2239                 \__graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2240             }
2241         }
2242     }
2243 }
2244 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_svg:n
2245 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_svg:n
2246 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_svg:n
2247 \cs_new:Npn \__graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2248 {#2}

```

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

```

\__graphics_backend_get_pagecount:n
2249 \__graphics_backend_loaded:n
2250 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }

(End of definition for \__graphics_backend_get_pagecount:n.)

2251 \</dvisvgm>
2252 \</package>

```

6 l3backend-pdf implementation

```

2253 \*package>
2254 \<@=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.

```

2255 \*!dvisvgm>

\l__pdf_internal_box
2256 \box_new:N \l__pdf_internal_box

(End of definition for \l__pdf_internal_box.)

2257 \</!dvisvgm>

```

6.2 dvips backend

```

2258 \*dvips>

\__pdf_backend_pdfmark:n
\__pdf_backend_pdfmark:e
2259 \cs_new_protected:Npn \__pdf_backend_pdfmark:n #1
2260 { \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2261 \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
2262 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2263 { \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2264 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2265 { \__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.

2266 `\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`

2267 `\cs_new_protected:Npn __pdf_backend_object_new:n #1`

2268 `{`

2269 `\int_gincr:N \g__pdf_backend_object_int`

2270 `\int_const:cn`

2271 `{ c__pdf_object_ \tl_to_str:n {#1} _int }`

2272 `{ \g__pdf_backend_object_int }`

2273 `}`

2274 `\cs_new:Npn __pdf_backend_object_ref:n #1`

2275 `{ { 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`.)

`__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`

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.

2276 `\cs_new_protected:Npn __pdf_backend_object_write:nnn #1#2#3`

2277 `{`

2278 `__pdf_backend_object_write_aux:nnn`

2279 `{ __pdf_backend_object_ref:n {#1} }`

2280 `{#2} {#3}`

2281 `}`

2282 `\cs_generate_variant:Nn __pdf_backend_object_write:nnn { nne }`

2283 `\cs_new_protected:Npn __pdf_backend_object_write_aux:nnn #1#2#3`

2284 `{`

2285 `__pdf_backend_pdfmark:e`

2286 `{`

2287 `/objdef ~ #1`

2288 `/type`

2289 `\str_case:nn {#2}`

2290 `{`

2291 `{ array } { /array }`

2292 `{ dict } { /dict }`

2293 `{ fstream } { /stream }`

2294 `{ stream } { /stream }`

2295 `}`

2296 `/OBJ`

2297 `}`

2298 `\use:c { __pdf_backend_object_write_ #2 :nn } {#1} {#3}`

2299 `}`

2300 `\cs_new_protected:Npn __pdf_backend_object_write_array:nn #1#2`

2301 `{`

2302 `__pdf_backend_pdfmark:e`

2303 `{ #1 ~0~ [~ \exp_not:n {#2} ~] ~ /PUTINTERVAL }`

2304 `}`

2305 `\cs_new_protected:Npn __pdf_backend_object_write_dict:nn #1#2`

2306 `{`

2307 `__pdf_backend_pdfmark:e`

```

2308     { #1 << \exp_not:n {#2} >> /PUT }
2309   }
2310 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2311 {
2312   \exp_args:Ne
2313     \__pdf_backend_object_write_fstream:nnn {#1} #2
2314 }
2315 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2316 {
2317   \__kernel_backend_postscript:n
2318   {
2319     SDict ~ begin ~
2320     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2321     mark ~ #1 ~ ( #3 ) ~ ( r ) ~ file ~ /PUT ~ pdfmark ~
2322     end
2323   }
2324 }
2325 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2326 {
2327   \exp_args:Ne
2328     \__pdf_backend_object_write_stream:nnn {#1} #2
2329 }
2330 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2331 {
2332   \__kernel_backend_postscript:n
2333   {
2334     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2335     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2336   }
2337 }

```

(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
2338 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2339 {
2340   \int_gincr:N \g__pdf_backend_object_int
2341   \__pdf_backend_object_write_aux:nnn
2342     { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2343     {#1} {#2}
2344 }
2345 \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.

```

2346 \cs_new:Npn \__pdf_backend_object_last:
2347 { { 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.

```

2348 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2349 { { 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.
2350 \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.
2351 \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.
2352 \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.
2353 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2354 {
2355   \exp_args:Nf \__pdf_backend_annotation_aux:nnnn
2356   { \dim_eval:n {#1} } {#2} {#3} {#4}
2357 }
2358 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2359 {
2360   \box_move_down:nn {#3}
2361   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2362   \box_move_up:nn {#2}
2363   {
2364     \hbox:n
2365     {
2366       \__kernel_kern:n {#1}
2367       \__kernel_backend_postscript:n { pdf.save.ur }
2368       \__kernel_kern:n { -#1 }
2369     }
2370   }
2371   \int_gincr:N \g__pdf_backend_object_int
2372   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2373   \__pdf_backend_pdfmark:e
2374   {
2375     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2376     pdf.rect
2377     #4 ~
2378     /ANN
2379   }
2380 }
(End of definition for \__pdf_backend_annotation:nnnn.)

```

<code>_pdf_backend_annotation_last:</code>	Provide the last annotation we created: could get tricky of course if other packages are loaded. <pre> 2381 \cs_new:Npn _pdf_backend_annotation_last: 2382 { { pdf.obj \int_use:N \g__pdf_backend_annotation_int } } </pre> <i>(End of definition for _pdf_backend_annotation_last:.)</i>
<code>\g__pdf_backend_link_int</code>	To track annotations which are links. <pre> 2383 \int_new:N \g__pdf_backend_link_int </pre> <i>(End of definition for \g__pdf_backend_link_int.)</i>
<code>\g__pdf_backend_link_dict_tl</code>	To pass information to the end-of-link function. <pre> 2384 \tl_new:N \g__pdf_backend_link_dict_tl </pre> <i>(End of definition for \g__pdf_backend_link_dict_tl.)</i>
<code>\g__pdf_backend_link_sf_int</code>	Needed to save/restore space factor, which is needed to deal with the face we need a box. <pre> 2385 \int_new:N \g__pdf_backend_link_sf_int </pre> <i>(End of definition for \g__pdf_backend_link_sf_int.)</i>
<code>\g__pdf_backend_link_math_bool</code>	Needed to save/restore math mode. <pre> 2386 \bool_new:N \g__pdf_backend_link_math_bool </pre> <i>(End of definition for \g__pdf_backend_link_math_bool.)</i>
<code>\g__pdf_backend_link_bool</code>	Track link formation: we cannot nest at all. <pre> 2387 \bool_new:N \g__pdf_backend_link_bool </pre> <i>(End of definition for \g__pdf_backend_link_bool.)</i>
<code>\l__pdf_breaklink_pdfmark_tl</code>	Swappable content for link breaking. <pre> 2388 \tl_new:N \l__pdf_breaklink_pdfmark_tl 2389 \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdfmark } </pre> <i>(End of definition for \l__pdf_breaklink_pdfmark_tl.)</i>
<code>_pdf_breaklink_postscript:n</code>	To allow dropping material unless link breaking is active. <pre> 2390 \cs_new_protected:Npn _pdf_breaklink_postscript:n #1 { } </pre> <i>(End of definition for _pdf_breaklink_postscript:n.)</i>
<code>__pdf_breaklink_usebox:N</code>	Swappable box unpacking or use. <pre> 2391 \cs_new_eq:NN __pdf_breaklink_usebox:N \box_use:N </pre> <i>(End of definition for __pdf_breaklink_usebox:N.)</i>

```

\__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:

```

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.

```

2392 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2393 {
2394   \__pdf_backend_link_begin:nw
2395   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2396 }
2397 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
2398 { \__pdf_backend_link_begin:nw {#1#2} }
2399 \cs_new_protected:Npn \__pdf_backend_link_begin:nw #1
2400 {
2401   \bool_if:NF \g__pdf_backend_link_bool
2402   { \__pdf_backend_link_begin_aux:nw {#1} }
2403 }

```

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.

```

2404 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2405 {
2406   \bool_gset_true:N \g__pdf_backend_link_bool
2407   \__kernel_backend_postscript:n
2408   { /pdf.link.dict ( #1 ) def }
2409   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2410   \__pdf_backend_link_sf_save:
2411   \mode_if_math:TF
2412   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2413   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2414   \hbox_set:Nw \l__pdf_backend_content_box
2415   \__pdf_backend_link_sf_restore:
2416   \bool_if:NT \g__pdf_backend_link_math_bool
2417   { \c_math_toggle_token }
2418 }
2419 \cs_new_protected:Npn \__pdf_backend_link_end:
2420 {
2421   \bool_if:NT \g__pdf_backend_link_bool
2422   { \__pdf_backend_link_end_aux: }
2423 }
2424 \cs_new_protected:Npn \__pdf_backend_link_end_aux:

```



```

2425 {
2426   \bool_if:NT \g__pdf_backend_link_math_bool
2427   { \c_math_toggle_token }
2428   \__pdf_backend_link_sf_save:
2429   \hbox_set_end:
2430   \__pdf_backend_link_minima:
2431   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2432   \exp_args:Ne \__pdf_backend_link_outerbox:n
2433   {
2434     \int_if_odd:nTF { \value { page } }
2435     { \oddsidemargin }
2436     { \evensidemargin }
2437   }
2438   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2439   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2440   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2441   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2442   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2443   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2444   {
2445     \hbox:n
2446     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2447   }
2448   \int_gincr:N \g__pdf_backend_object_int
2449   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2450   \__kernel_backend_postscript:e
2451   {
2452     mark
2453     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2454     \g__pdf_backend_link_dict_tl \c_space_tl
2455     pdf.rect
2456     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2457   }
2458   \__pdf_backend_link_sf_restore:
2459   \bool_gset_false:N \g__pdf_backend_link_bool
2460 }
2461 \cs_new_protected:Npn \__pdf_backend_link_minima:
2462 {
2463   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2464   \__kernel_backend_postscript:e
2465   {
2466     /pdf.linkdp.pad ~
2467     \dim_to_decimal:n
2468     {
2469       \dim_max:nn
2470       {
2471         \box_dp:N \l__pdf_backend_model_box
2472         - \box_dp:N \l__pdf_backend_content_box
2473       }
2474       { Opt }
2475     } ~
2476     pdf.pt.dvi ~ def
2477     /pdf.linkht.pad ~
2478     \dim_to_decimal:n

```

```

2479         {
2480             \dim_max:nn
2481             {
2482                 \box_ht:N \l__pdf_backend_model_box
2483                 - \box_ht:N \l__pdf_backend_content_box
2484             }
2485             { 0pt }
2486         } ~
2487         pdf.pt.dvi ~ def
2488     }
2489 }
2490 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2491 {
2492     \__kernel_backend_postscript:e
2493     {
2494         /pdf.outerbox
2495         [
2496             \dim_to_decimal:n {#1} ~
2497             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2498             \dim_to_decimal:n { #1 + \textwidth } ~
2499             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2500         ]
2501         [ exch { pdf.pt.dvi } forall ] def
2502         /pdf.baselineskip ~
2503         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2504         { pdf.pt.dvi ~ def }
2505         { pop ~ pop }
2506         ifelse
2507     }
2508 }
2509 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2510 {
2511     \int_gset:Nn \g__pdf_backend_link_sf_int
2512     {
2513         \mode_if_horizontal:TF
2514         { \tex_spacefactor:D }
2515         { 0 }
2516     }
2517 }
2518 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2519 {
2520     \mode_if_horizontal:T
2521     {
2522         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2523         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2524     }
2525 }

```

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

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.

```

2526 \use_none:n
2527 {

```

```

2528 \cs_if_exist:NT \@makecol@hook
2529 {
2530   \tl_put_right:Nn \@makecol@hook
2531   {
2532     \box_if_empty:NF \l_shipout_box
2533     {
2534       \vbox_set:Nn \l_shipout_box
2535       {
2536         \__kernel_backend_postscript:n
2537         {
2538           pdf.globaldict /pdf.brokenlink.rect ~ known
2539           { pdf.bordertracking.continue }
2540           if
2541           }
2542           \vbox_unpack_drop:N \l_shipout_box
2543           \__kernel_backend_postscript:n
2544           { pdf.bordertracking.endpage }
2545         }
2546       }
2547     }
2548     \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2549     \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2550     \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2551   }
2552 }

```

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

```

2553 \cs_new:Npn \__pdf_backend_link_last:
2554 { { 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.

```

2555 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2556 {
2557   \__kernel_backend_postscript:e
2558   {
2559     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2560   }
2561 }

```

(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.

`__pdf_backend_destination:nnnn`
`__pdf_backend_destination_aux:nnnn`

```

2562 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2563 {
2564   \__kernel_backend_postscript:n { pdf.dest.anchor }
2565   \__pdf_backend_pdfmark:e
2566   {
2567     /View
2568     [

```

```

2569     \str_case:nnF {#2}
2570     {
2571         { xyz }    { /XYZ ~ pdf.dest.point ~ null }
2572         { fit }    { /Fit }
2573         { fitb }   { /FitB }
2574         { fitbh }  { /FitBH ~ pdf.dest.y }
2575         { fitbv }  { /FitBV ~ pdf.dest.x }
2576         { fith }   { /FitH ~ pdf.dest.y }
2577         { fitv }   { /FitV ~ pdf.dest.x }
2578         { fitr }   { /Fit }
2579     }
2580     {
2581         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2582     }
2583 ]
2584 /Dest ( \exp_not:n {#1} ) cvn
2585 /DEST
2586 }
2587 }
2588 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2589 {
2590     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2591     { \dim_eval:n {#2} } {#1} {#3} {#4}
2592 }
2593 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2594 {
2595     \vbox_to_zero:n
2596     {
2597         \__kernel_kern:n {#4}
2598         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2599         \tex_vss:D
2600     }
2601     \__kernel_kern:n {#1}
2602     \vbox_to_zero:n
2603     {
2604         \__kernel_kern:n { -#3 }
2605         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2606         \tex_vss:D
2607     }
2608     \__kernel_kern:n { -#1 }
2609     \__pdf_backend_pdfmark:n
2610     {
2611         /View
2612         [
2613             /FitR ~
2614             pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2615             pdf.urx ~ pdf.ury ~ pdf.dest2device
2616         ]
2617         /Dest ( #2 ) cvn
2618         /DEST
2619     }
2620 }

```

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

6.2.4 Structure

Doable for the usual ps2pdf method.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2621 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2622 {
2623   \int_compare:nNnT {#1} = 0
2624   {
2625     \__kernel_backend_literal_postscript:n
2626     {
2627       /setdistillerparams ~ where
2628       { pop << /CompressPages ~ false >> setdistillerparams }
2629       if
2630     }
2631   }
2632 }
2633 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2634 {
2635   \bool_if:nF {#1}
2636   {
2637     \__kernel_backend_literal_postscript:n
2638     {
2639       /setdistillerparams ~ where
2640       { pop << /CompressStreams ~ false >> setdistillerparams }
2641       if
2642     }
2643   }
2644 }

```

(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
2645 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2646 {
2647   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2648 }
2649 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2650 {
2651   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2652 }

```

(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:
2653 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2654 \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:
2655 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2656 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2657 \cs_new_protected:Npn \__pdf_backend_emc:
2658 { \__pdf_backend_pdfmark:n { /EMC } }

```

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

2659 `</dvips>`

6.3 LuaTeX and pdfTeX backend

2660 `<*luatex | pdftex>`

6.3.1 Annotations

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

```
2661 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2662 {
2663   <*luatex>
2664     \tex_pdfextension:D annot ~
2665   </luatex>
2666   <*pdftex>
2667     \tex_pdfannot:D
2668   </pdftex>
2669     width ~ \dim_eval:n {#1} ~
2670     height ~ \dim_eval:n {#2} ~
2671     depth ~ \dim_eval:n {#3} ~
2672     {#4}
2673 }
```

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

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

```
2674 \cs_new:Npe \_pdf_backend_annotation_last:
2675 {
2676   \exp_not:N \int_value:w
2677   <*luatex>
2678     \exp_not:N \tex_pdffeedback:D lastannot ~
2679   </luatex>
2680   <*pdftex>
2681     \exp_not:N \tex_pdflastannot:D
2682   </pdftex>
2683     \c_space_tl 0 ~ R
2684 }
```

(End of definition for `_pdf_backend_annotation_last:.`)

`_pdf_backend_link_begin_goto:nnw` Links are all created using the same internals.

```
\_pdf_backend_link_begin_user:nnw
\_pdf_backend_link_begin:nnnw
\_pdf_backend_link_end:
2685 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2686 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2687 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2688 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2689 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2690 {
2691   <*luatex>
2692     \tex_pdfextension:D startlink ~
2693   </luatex>
2694   <*pdftex>
```

```

2695 \tex_pdfstartlink:D
2696 </pdfTeX>
2697   attr {#1}
2698   #2 {#3}
2699 }
2700 \cs_new_protected:Npn \__pdf_backend_link_end:
2701 {
2702 <*luatex>
2703   \tex_pdfextension:D endlink \scan_stop:
2704 </luatex>
2705 <*pdfTeX>
2706   \tex_pdfendlink:D
2707 </pdfTeX>
2708 }

```

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

__pdf_backend_link_last: Formatted for direct use.

```

2709 \cs_new:Npe \__pdf_backend_link_last:
2710 {
2711   \exp_not:N \int_value:w
2712 <*luatex>
2713   \exp_not:N \tex_pdffeedback:D lastlink ~
2714 </luatex>
2715 <*pdfTeX>
2716   \exp_not:N \tex_pdflastlink:D
2717 </pdfTeX>
2718   \c_space_tl 0 ~ R
2719 }

```

(End of definition for __pdf_backend_link_last:.)

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

```

2720 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2721 {
2722 <*luatex>
2723   \tex_pdfvariable:D linkmargin
2724 </luatex>
2725 <*pdfTeX>
2726   \tex_pdflinkmargin:D
2727 </pdfTeX>
2728   \dim_eval:n {#1} \scan_stop:
2729 }

```

(End of definition for __pdf_backend_link_margin:n.)

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

_pdf_backend_destination:nnnn

```

2730 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2731 {
2732 <*luatex>
2733   \tex_pdfextension:D dest ~
2734 </luatex>
2735 <*pdfTeX>

```

```

2736 \tex_pdfdest:D
2737 </pdftex>
2738 name {#1}
2739 \str_case:nnF {#2}
2740 {
2741 { xyz } { xyz }
2742 { fit } { fit }
2743 { fitb } { fitb }
2744 { fitbh } { fitbh }
2745 { fitbv } { fitbv }
2746 { fith } { fith }
2747 { fitv } { fitv }
2748 { fitr } { fitr }
2749 }
2750 { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2751 \scan_stop:
2752 }
2753 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2754 {
2755 <*luatex>
2756 \tex_pdfextension:D dest ~
2757 </luatex>
2758 <*pdftex>
2759 \tex_pdfdest:D
2760 </pdftex>
2761 name {#1}
2762 fitr ~
2763 width \dim_eval:n {#2} ~
2764 height \dim_eval:n {#3} ~
2765 depth \dim_eval:n {#4} \scan_stop:
2766 }

```

(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
2767 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2768 {
2769 <*luatex>
2770 \tex_pdfextension:D catalog
2771 </luatex>
2772 <*pdftex>
2773 \tex_pdfcatalog:D
2774 </pdftex>
2775 { / #1 ~ #2 }
2776 }
2777 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2778 {
2779 <*luatex>
2780 \tex_pdfextension:D info
2781 </luatex>
2782 <*pdftex>
2783 \tex_pdfinfo:D

```



```

2784 </pdftex>
2785 { / #1 ~ #2 }
2786 }

```

(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.

```

2787 \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
2788 \cs_new_protected:Npn \_pdf_backend_object_new:n #1
2789 {
2790 <*luatex>
2791 \tex_pdfextension:D obj ~
2792 </luatex>
2793 <*pdftex>
2794 \tex_pdfobj:D
2795 </pdftex>
2796 reserveobjnum ~
2797 \int_const:cn
2798 { c_pdf_object_ \tl_to_str:n {#1} _int }
2799 <*luatex>
2800 { \tex_pdffeedback:D lastobj }
2801 </luatex>
2802 <*pdftex>
2803 { \tex_pdflastobj:D }
2804 </pdftex>
2805 }
2806 \cs_new:Npn \_pdf_backend_object_ref:n #1
2807 { \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
2808 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
2809 {
2810 <*luatex>
2811 \tex_immediate:D \tex_pdfextension:D obj ~
2812 </luatex>
2813 <*pdftex>
2814 \tex_immediate:D \tex_pdfobj:D
2815 </pdftex>
2816 useobjnum ~
2817 \int_use:c
2818 { c_pdf_object_ \tl_to_str:n {#1} _int }
2819 \_pdf_backend_object_write:nn {#2} {#3}
2820 }
2821 \cs_new:Npn \_pdf_backend_object_write:nn #1#2
2822 {
2823 \str_case:nn {#1}
2824 {

```

```

2825 { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2826 { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2827 { fstream }
2828 {
2829   stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2830   file ~ { \__pdf_exp_not_ii:nn #2 }
2831 }
2832 { stream }
2833 {
2834   stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2835   { \__pdf_exp_not_ii:nn #2 }
2836 }
2837 }
2838 }
2839 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2840 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2841 \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 Much like writing, but direct creation.

```

\__pdf_backend_object_now:ne
2842 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2843 {
2844   <*luatex>
2845   \tex_immediate:D \tex_pdfextension:D obj ~
2846   </luatex>
2847   <*pdftex>
2848   \tex_immediate:D \tex_pdfobj:D
2849   </pdftex>
2850   \__pdf_backend_object_write:nn {#1} {#2}
2851 }
2852 \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.

```

2853 \cs_new:Npe \__pdf_backend_object_last:
2854 {
2855   \exp_not:N \int_value:w
2856   <*luatex>
2857   \exp_not:N \tex_pdffeedback:D lastobj ~
2858   </luatex>
2859   <*pdftex>
2860   \exp_not:N \tex_pdflastobj:D
2861   </pdftex>
2862   \c_space_tl 0 ~ R
2863 }

```

(End of definition for __pdf_backend_object_last:.)

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

```

2864 \cs_new:Npe \__pdf_backend_pageobject_ref:n #1
2865 {
2866   \exp_not:N \int_value:w
2867   <*luatex>

```

```

2868 \exp_not:N \tex_pdffeedback:D pageref
2869 </luatex>
2870 <*pdfTeX>
2871 \exp_not:N \tex_pdfpageref:D
2872 </pdfTeX>
2873 \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2874 }

```

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

6.3.4 Structure

Simply pass data to the engine.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
\__pdf_backend_objcompresslevel:n
2875 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2876 {
2877   \tex_global:D
2878   <*luatex>
2879   \tex_pdfvariable:D compresslevel
2880 </luatex>
2881 <*pdfTeX>
2882   \tex_pdfcompresslevel:D
2883 </pdfTeX>
2884   \int_value:w \int_eval:n {#1} \scan_stop:
2885 }
2886 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2887 {
2888   \bool_if:nTF {#1}
2889     { \__pdf_backend_objcompresslevel:n { 2 } }
2890     { \__pdf_backend_objcompresslevel:n { 0 } }
2891 }
2892 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2893 {
2894   \tex_global:D
2895   <*luatex>
2896   \tex_pdfvariable:D objcompresslevel
2897 </luatex>
2898 <*pdfTeX>
2899   \tex_pdfobjcompresslevel:D
2900 </pdfTeX>
2901   #1 \scan_stop:
2902 }

```

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

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

```

2903 \cs_new_protected:Npe \__pdf_backend_version_major_gset:n #1
2904 {
2905   <*luatex>
2906   \int_compare:nNnT \tex luatexversion:D > { 106 }
2907   {
2908     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2909     \exp_not:N \int_eval:n {#1} \scan_stop:
2910   }
2911 </luatex>

```

```

2912 <*pdftex>
2913   \cs_if_exist:NT \tex_pdfmajorversion:D
2914   {
2915     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2916     \exp_not:N \int_eval:n {#1} \scan_stop:
2917   }
2918 </pdftex>
2919 }
2920 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2921 {
2922   \tex_global:D
2923   <*luatex>
2924     \tex_pdfvariable:D minorversion
2925   </luatex>
2926   <*pdftex>
2927     \tex_pdfminorversion:D
2928   </pdftex>
2929     \int_eval:n {#1} \scan_stop:
2930 }

```

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

__pdf_backend_version_major: As above.

```

\__pdf_backend_version_minor:
2931 \cs_new:Npe \__pdf_backend_version_major:
2932 {
2933   <*luatex>
2934     \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2935     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2936     { 1 }
2937   </luatex>
2938   <*pdftex>
2939     \cs_if_exist:NTF \tex_pdfmajorversion:D
2940     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2941     { 1 }
2942   </pdftex>
2943 }
2944 \cs_new:Npn \__pdf_backend_version_minor:
2945 {
2946   \tex_the:D
2947   <*luatex>
2948     \tex_pdfvariable:D minorversion
2949   </luatex>
2950   <*pdftex>
2951     \tex_pdfminorversion:D
2952   </pdftex>
2953 }

```

(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>.
 __pdf_backend_emc:

```

2954 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2955 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2956 \cs_new_protected:Npn \__pdf_backend_emc:
2957 { \__kernel_backend_literal_page:n { EMC } }

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

2958 </luatex | pdftex>

```

6.4 dvipdfmx backend

```

2959 <*dvipdfmx | xetex>

```

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

```

\__pdf_backend:e
2960 \cs_new_protected:Npe \__pdf_backend:n #1
2961 { \__kernel_backend_literal:n { pdf: #1 } }
2962 \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
2963 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2964 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2965 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2966 { \__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
2967 \int_new:N \g__pdf_backend_object_int
2968 \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
2969 \cs_new_protected:Npn \__pdf_backend_object_new:n #1
2970 {
2971   \int_gincr:N \g__pdf_backend_object_int
2972   \int_const:cn
2973     { c__pdf_object_ \tl_to_str:n {#1} _int }
2974   { \g__pdf_backend_object_int }
2975 }
2976 \cs_new:Npn \__pdf_backend_object_ref:n #1
2977 { @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.)

```

\__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

```

This is where we choose the actual type.

```

2978 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2979 {
2980   \use:c { __pdf_backend_object_write_ #2 :nn }
2981   { \__pdf_backend_object_ref:n {#1} } {#3}
2982 }
2983 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2984 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2985 {
2986   \__pdf_backend:e
2987   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2988 }
2989 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2990 {
2991   \__pdf_backend:e
2992   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2993 }
2994 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2995 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2996 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2997 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2998 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2999 {
3000   \__pdf_backend:e
3001   {
3002     #1 stream ~ #2 ~
3003     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
3004   }
3005 }

```

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

```

\__pdf_backend_object_now:nn
\__pdf_backend_object_now:ne

```

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

```

3006 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
3007 {
3008   \int_gincr:N \g__pdf_backend_object_int
3009   \exp_args:Nne \use:c { __pdf_backend_object_write_ #1 :nn }
3010   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
3011   {#2}
3012 }
3013 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

```

\__pdf_backend_object_last:

```

```

3014 \cs_new:Npn \__pdf_backend_object_last:
3015 { @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.

```

3016 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
3017 { @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.

```
3018 \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.

```
3019 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
3020 {
3021   \int_gincr:N \g_pdf_backend_object_int
3022   \int_gset_eq:NN \g_pdf_backend_annotation_int \g_pdf_backend_object_int
3023   \_pdf_backend:e
3024   {
3025     ann ~ @pdf.obj \int_use:N \g_pdf_backend_object_int \c_space_tl
3026     width ~ \dim_eval:n {#1} ~
3027     height ~ \dim_eval:n {#2} ~
3028     depth ~ \dim_eval:n {#3} ~
3029     << /Type /Annot #4 >>
3030   }
3031 }
```

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

`_pdf_backend_annotation_last:`

```
3032 \cs_new:Npn \_pdf_backend_annotation_last:
3033 { @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.

```
3034 \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:`

```
3035 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
3036 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
3037 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
3038 { \_pdf_backend_link_begin:n {#1#2} }
3039 \cs_new_protected:Npe \_pdf_backend_link_begin:n #1
3040 {
3041   \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int
3042   \_pdf_backend:e
3043   {
3044     bann ~
3045     @pdf.lnk
3046     \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int
3047     \c_space_tl
3048     <<
3049     /Type /Annot
3050     #1
3051     >>
3052   }
3053 }
3054 \cs_new_protected:Npn \_pdf_backend_link_end:
3055 { \_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.

```
3056 \cs_new:Npn \__pdf_backend_link_last:
3057 { @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.

```
3058 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
3059 { \__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
`__pdf_backend_destination:nnnn` Grahn: the idea is to avoid needing to do any calculations in `TeX` by using the backend
`__pdf_backend_destination_aux:nnnn` data for `@xpos` and `@ypos`. `/FitR` without rule spec doesn't work, so it falls back to `/Fit`
here.

```
3060 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
3061 {
3062   \__pdf_backend:e
3063   {
3064     dest ~ ( \exp_not:n {#1} )
3065     [
3066       @thispage
3067       \str_case:nnF {#2}
3068       {
3069         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
3070         { fit } { /Fit }
3071         { fitb } { /FitB }
3072         { fitbh } { /FitBH }
3073         { fitbv } { /FitBV ~ @xpos }
3074         { fith } { /FitH ~ @ypos }
3075         { fitv } { /FitV ~ @xpos }
3076         { fitr } { /Fit }
3077       }
3078       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
3079     ]
3080   }
3081 }
3082 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
3083 {
3084   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
3085   { \dim_eval:n {#2} } {#1} {#3} {#4}
3086 }
3087 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
3088 {
3089   \vbox_to_zero:n
3090   {
3091     \__kernel_kern:n {#4}
3092     \hbox:n
3093     {
3094       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
3095       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
```



```

3096     }
3097     \tex_vss:D
3098   }
3099   \__kernel_kern:n {#1}
3100   \vbox_to_zero:n
3101   {
3102     \__kernel_kern:n { -#3 }
3103     \hbox:n
3104     {
3105       \__pdf_backend:n
3106       {
3107         dest ~ (#2)
3108         [
3109           @thispage
3110           /FitR ~
3111           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
3112           @xpos ~ @ypos
3113         ]
3114       }
3115     }
3116     \tex_vss:D
3117   }
3118   \__kernel_kern:n { -#1 }
3119 }

```

(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
_pdf_backend_compress_objects:n

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

```

3120 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
3121 { \__kernel_backend_literal:e { dvipdfmx:config~z~ \int_eval:n {#1} } }
3122 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
3123 {
3124   \bool_if:nF {#1}
3125   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
3126 }

```

(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

We start with the assumption that the default is active.

```

3127 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
3128 {
3129   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
3130   \__kernel_backend_literal:e { pdf:majorversion~ \__pdf_backend_version_major: }
3131 }
3132 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
3133 {
3134   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
3135   \__kernel_backend_literal:e { pdf:minorversion~ \__pdf_backend_version_minor: }
3136 }

```

(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:` 3137 `\cs_new:Npn _pdf_backend_version_major: { 1 }`
3138 `\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>.
`_pdf_backend_emc:` 3139 `\cs_new_protected:Npn _pdf_backend_bdc:nn #1#2`
3140 `{ _kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }`
3141 `\cs_new_protected:Npn _pdf_backend_emc:`
3142 `{ _kernel_backend_literal_page:n { EMC } }`
(End of definition for _pdf_backend_bdc:nn and _pdf_backend_emc:.)
3143 `\</dvipdfmx | xetex>`

6.5 dvisvgm backend

3144 `*dvisvgm>`

6.5.1 Annotations

`_pdf_backend_annotation:nnnn` 3145 `\cs_new_protected:Npn _pdf_backend_annotation:nnnn #1#2#3#4 { }`
(End of definition for _pdf_backend_annotation:nnnn.)

`_pdf_backend_annotation_last:` 3146 `\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` 3147 `\cs_new_protected:Npn _pdf_backend_link_begin_goto:nnw #1#2 { }`
`_pdf_backend_link_begin:nnnw` 3148 `\cs_new_protected:Npn _pdf_backend_link_begin_user:nnw #1#2 { }`
`_pdf_backend_link_end:` 3149 `\cs_new_protected:Npn _pdf_backend_link_begin:nnnw #1#2#3 { }`
3150 `\cs_new_protected:Npn _pdf_backend_link_end: { }`
(End of definition for _pdf_backend_link_begin_goto:nnw and others.)

`_pdf_backend_link_last:` 3151 `\cs_new:Npn _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.
3152 `\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:nnnn` 3153 `\cs_new_protected:Npn _pdf_backend_destination:nn #1#2 { }`
3154 `\cs_new_protected:Npn _pdf_backend_destination:nnnn #1#2#3#4 { }`
(End of definition for _pdf_backend_destination:nn and _pdf_backend_destination:nnnn.)

6.5.2 Catalogue entries

No-op.

```

    \_pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
3155 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2 { }
3156 \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

All no-ops here.

```

\__pdf_backend_object_new:n
\__pdf_backend_object_ref:n
    \_pdf_backend_object_write:nnn
    \_pdf_backend_object_write:ne
\__pdf_backend_object_now:nn
\__pdf_backend_object_now:ne
\__pdf_backend_object_last:
    \_pdf_backend_pageobject_ref:n
3157 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1 { }
3158 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
3159 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3 { }
3160 \cs_new_protected:Npn \_pdf_backend_object_write:nne #1#2#3 { }
3161 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
3162 \cs_new_protected:Npn \_pdf_backend_object_now:ne #1#2 { }
3163 \cs_new:Npn \_pdf_backend_object_last: { }
3164 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1 { }

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

```

6.5.4 Structure

These are all no-ops.

```

    \_pdf_backend_compresslevel:n
    \_pdf_backend_compress_objects:n
3165 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1 { }
3166 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1 { }

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

```

Data not available!

```

\__pdf_backend_version_major_gset:n
\__pdf_backend_version_minor_gset:n
3167 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
3168 \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.)

```

Data not available!

```

    \_pdf_backend_version_major:
    \_pdf_backend_version_minor:
3169 \cs_new:Npn \_pdf_backend_version_major: { -1 }
3170 \cs_new:Npn \_pdf_backend_version_minor: { -1 }

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

```

More no-ops.

```

\__pdf_backend_bdc:nn
    \__pdf_backend_emc:
3171 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2 { }
3172 \cs_new_protected:Npn \_pdf_backend_emc: { }

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

3173 </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.

```
3174 <*dvipdfmx | dvips>
```

```
\_pdf_backend_pagesize_gset:nn
```

This is done as a backend literal, so we deal with it using the shipout hook.

```
3175 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
3176 {
3177   \__kernel_backend_first_shipout:n
3178   {
3179     \__kernel_backend_literal:e
3180     {
3181       <*dvipdfmx>
3182         pdf:pagesize ~
3183         width ~ \dim_eval:n {#1} ~
3184         height ~ \dim_eval:n {#2}
3185       </dvipdfmx>
3186       <*dvips>
3187         papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
3188       </dvips>
3189     }
3190   }
3191 }
```

(End of definition for _pdf_backend_pagesize_gset:nn.)

```
3192 </dvipdfmx | dvips>
```

```
3193 <*luatex | pdftex | xetex>
```

```
\_pdf_backend_pagesize_gset:nn
```

Pass to the primitives.

```
3194 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
3195 {
3196   \dim_gset:Nn \tex_pagewidth:D {#1}
3197   \dim_gset:Nn \tex_pageheight:D {#2}
3198 }
```

(End of definition for _pdf_backend_pagesize_gset:nn.)

```
3199 </luatex | pdftex | xetex>
```

```
3200 <*dvisvgm>
```

```
\_pdf_backend_pagesize_gset:nn
```

A no-op.

```
3201 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2 { }
```

(End of definition for _pdf_backend_pagesize_gset:nn.)

```
3202 </dvisvgm>
```

```
3203 </package>
```

7 l3backend-opacity implementation

```
3204 <*package>
3205 <@@=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.

```
3206 <*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.

```
3207 \cs_new_protected:Npn \__opacity_backend_select:n #1
3208 {
3209   \exp_args:Ne \__opacity_backend_select_aux:n
3210   { \fp_eval:n { min(max(0,#1),1) } }
3211 }
3212 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3213 {
3214   \__opacity_backend:nnn {#1} { fill } { ca }
3215   \__opacity_backend:nnn {#1} { stroke } { CA }
3216 }
3217 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3218 {
3219   \__opacity_backend:enn
3220   { \fp_eval:n { min(max(0,#1),1) } }
3221   { fill }
3222   { ca }
3223 }
3224 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3225 {
3226   \__opacity_backend:enn
3227   { \fp_eval:n { min(max(0,#1),1) } }
3228   { stroke }
3229   { CA }
3230 }
3231 \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3232 {
3233   \__kernel_backend_postscript:n
3234   {
3235     product ~ (Ghostscript) ~ search
3236     {
3237       pop ~ pop ~ pop ~
3238       #1 ~ .set #2 constantalpha
3239     }
3240     {
3241       pop ~
3242       mark ~
3243       /#3 ~ #1
```

```

3244         /SetTransparency ~
3245         pdfmark
3246     }
3247     ifelse
3248 }
3249 }
3250 \cs_generate_variant:Nn \_opacity_backend:nnn { e }

```

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

```

3251 </dvips>
3252 <*dvipdfmx | luatex | pdftex | xetex>

```

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

```

3253 \bool_lazy_and:nnT
3254 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3255 { \pdfmanagement_if_active_p:}
3256 {
3257 <*luatex | pdftex>
3258     \_kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3259     { page ~ direct } { /opacity 1 ~ gs }
3260 </luatex | pdftex>
3261     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3262     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3263 }

```

(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
3264 \tl_new:N \l__opacity_backend_fill_tl
3265 \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:
3266 \cs_new_protected:Npn \_opacity_backend_select:n #1
3267 {
3268     \exp_args:Ne \_opacity_backend_select_aux:n
3269     { \fp_eval:n { min(max(0,#1),1) } }
3270 }
3271 \cs_new_protected:Npn \_opacity_backend_select_aux:n #1
3272 {
3273     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3274     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3275     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3276     { opacity #1 }
3277     { << /ca ~ #1 /CA ~ #1 >> }
3278 <*dvipdfmx | xetex>
3279     \_kernel_backend_literal_pdf:n
3280 </dvipdfmx | xetex>
3281 <*luatex | pdftex>
3282     \_kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3283 </luatex | pdftex>
3284     { /opacity #1 ~ gs }
3285     \group_insert_after:N \_opacity_backend_reset:

```

```

3286 }
3287 \bool_lazy_and:nnF
3288 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3289 { \pdfmanagement_if_active_p:}
3290 {
3291   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1 { }
3292 }
3293 \cs_new_protected:Npn \__opacity_backend_reset:
3294 {
3295   <*dvipdfmx | xetex>
3296   \__kernel_backend_literal_pdf:n
3297   { /opacity1 ~ gs }
3298   </dvipdfmx | xetex>
3299   <*luatex | pdftex>
3300   \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int
3301   </luatex | pdftex>
3302 }

```

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

```

\__opacity_backend_fill:n
\__opacity_backend_stroke:n
  \__opacity_backend_fillstroke:nn
  \__opacity_backend_fillstroke:ee

```

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

```

3303 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3304 {
3305   \__opacity_backend_fill_stroke:ee
3306   { \fp_eval:n { min(max(0,#1),1) } }
3307   \l__opacity_backend_stroke_tl
3308 }
3309 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3310 {
3311   \__opacity_backend_fill_stroke:ee
3312   \l__opacity_backend_fill_tl
3313   { \fp_eval:n { min(max(0,#1),1) } }
3314 }
3315 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3316 {
3317   \str_if_eq:nnTF {#1} {#2}
3318   { \__opacity_backend_select_aux:n {#1} }
3319   {
3320     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3321     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3322     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3323     { opacity.fill #1 }
3324     { << /ca ~ #1 >> }
3325     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3326     { opacity.stroke #1 }
3327     { << /CA ~ #2 >> }
3328   <*dvipdfmx | xetex>
3329   \__kernel_backend_literal_pdf:n
3330   </dvipdfmx | xetex>
3331   <*luatex | pdftex>
3332   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3333   </luatex | pdftex>

```

```

3334         { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3335         \group_insert_after:N \__opacity_backend_reset:
3336     }
3337 }
3338 \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.)

3339 </dviptfm | luatex | pdftex | xetex>
3340 <*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
3341 \cs_new_protected:Npn \__opacity_backend_select:n #1
3342 { \__opacity_backend:nn {#1} { } }
3343 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3344 { \__opacity_backend:nn {#1} { fill- } }
3345 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3346 { \__opacity_backend:nn { {#1} } { stroke- } }
3347 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3348 { \__kernel_backend_scope:e { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

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

3349 </dvisvgm>
3350 </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.

```

3351 <*lua>

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

3352 local pdfmanagement_active do
3353     local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3354     local cmd = pdfmanagement_if_active_p.cmdname
3355     if cmd == 'undefined_cs' then
3356         pdfmanagement_active = false
3357     else
3358         token.put_next(pdfmanagement_if_active_p)
3359         pdfmanagement_active = token.scan_int() ~= 0
3360     end
3361 end
3362
3363 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3364     luaotfload.set_transparent_colorstack(function() return token.create'c__opacity_backend_st
3365
3366 local transparent_register = {
3367     token.create'pdfmanagement_add:nnn',
3368     token.new(0, 1),
3369     'Page/Resources/ExtGState',
3370     token.new(0, 2),

```



```

3371     token.new(0, 1),
3372     '',
3373     token.new(0, 2),
3374     token.new(0, 1),
3375     '<</ca ',
3376     '',
3377     '/CA ',
3378     '',
3379     '>>',
3380     token.new(0, 2),
3381 }
3382 luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)
3383     value = (octet * -1):match(value)
3384     if not value then
3385         tex.error'Invalid transparency value'
3386         return
3387     end
3388     value = value:sub(1, -2)
3389     local result = 'opacity' .. value
3390     tex.runtoks(function()
3391         transparent_register[6], transparent_register[10], transparent_register[12] = result,
3392         tex.sprint(-2, transparent_register)
3393     end)
3394     return '/' .. result .. ' gs'
3395 end, 'l3opacity')
3396 end
3397 </lua>

```

8 l3backend-header implementation

```

3398 <*dvips & header>

```

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

```

3399 /color.sc { } def

```

(End of definition for color.sc.)

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

```

3400 TeXDict begin
3401 /TeXcolorseparation { setcolor } def
3402 end

```

(End of definition for TeXcolorseparation and separation.)

pdf.globaldict A small global dictionary for backend use.

```

3403 true setglobal
3404 /pdf.globaldict 4 dict def
3405 false setglobal

```

(End of definition for pdf.globaldict.)

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	

```

3406 /pdf.cvs { 65534 string cvs } def
3407 /pdf.dvi.pt { 72.27 mul Resolution div } def
3408 /pdf.pt.dvi { 72.27 div Resolution mul } def
3409 /pdf.rect.ht { dup 1 get neg exch 3 get add } def

```

(End of definition for pdf.cvs and others.)

pdf.linkmargin	Settings which are defined up-front in SDict.
pdf.linkdp.pad	
pdf.linkht.pad	

```

3410 /pdf.linkmargin { 1 pdf.pt.dvi } def
3411 /pdf.linkdp.pad { 0 } def
3412 /pdf.linkht.pad { 0 } def

```

(End of definition for pdf.linkmargin, pdf.linkdp.pad, and pdf.linkht.pad.)

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	
pdf.save.linkur	
pdf.llx	
pdf.lly	
pdf.urx	
pdf.ury	

```

3413 /pdf.rect
3414 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
3415 /pdf.save.ll
3416 {
3417     currentpoint
3418     /pdf.lly exch def
3419     /pdf.llx exch def
3420 }
3421 def
3422 /pdf.save.ur
3423 {
3424     currentpoint
3425     /pdf.ury exch def
3426     /pdf.urx exch def
3427 }
3428 def
3429 /pdf.save.linkll
3430 {
3431     currentpoint
3432     pdf.linkmargin add
3433     pdf.linkdp.pad add
3434     /pdf.lly exch def
3435     pdf.linkmargin sub
3436     /pdf.llx exch def
3437 }
3438 def
3439 /pdf.save.linkur
3440 {
3441     currentpoint
3442     pdf.linkmargin sub
3443     pdf.linkht.pad sub
3444     /pdf.ury exch def
3445     pdf.linkmargin add
3446     /pdf.urx exch def

```

```

3447 }
3448 def

```

(End of definition for pdf.rect and others.)

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 3449 /pdf.dest.anchor
pdf.dev.y 3450 {
pdf.tmpa 3451 currentpoint exch
pdf.tmpb 3452 pdf.dvi.pt 72 add
pdf.tmpc 3453 /pdf.dest.x exch def
pdf.tmpd 3454 pdf.dvi.pt
3455 vsize 72 sub exch sub
3456 /pdf.dest.y exch def
3457 }
3458 def
3459 /pdf.dest.point
3460 { pdf.dest.x pdf.dest.y } def
3461 /pdf.dest2device
3462 {
3463 /pdf.dest.y exch def
3464 /pdf.dest.x exch def
3465 matrix currentmatrix
3466 matrix defaultmatrix
3467 matrix invertmatrix
3468 matrix concatmatrix
3469 cvx exec
3470 /pdf.dev.y exch def
3471 /pdf.dev.x exch def
3472 /pdf.tmpd exch def
3473 /pdf.tmpc exch def
3474 /pdf.tmpb exch def
3475 /pdf.tmpa exch def
3476 pdf.dest.x pdf.tmpa mul
3477 pdf.dest.y pdf.tmpc mul add
3478 pdf.dev.x add
3479 pdf.dest.x pdf.tmpb mul
3480 pdf.dest.y pdf.tmpd mul add
3481 pdf.dev.y add
3482 }
3483 def

```

(End of definition for pdf.dest.anchor and others.)

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 3484 /pdf.bordertracking false def
pdf.bordertracking.end 3485 /pdf.bordertracking.begin
pdf.leftboundary
pdf.rightboundary
pdf.brokenlink.rect
pdf.brokenlink.skip
pdf.brokenlink.dict
pdf.bordertracking.endpage
pdf.bordertracking.continue
pdf.originx
pdf.originy

```

```

3486 {
3487   SDict /pdf.bordertracking true put
3488   SDict /pdf.leftboundary undef
3489   SDict /pdf.rightboundary undef
3490   /a where
3491   {
3492     /a
3493     {
3494       currentpoint pop
3495       SDict /pdf.rightboundary known dup
3496       {
3497         SDict /pdf.rightboundary get 2 index lt
3498         { not }
3499         if
3500       }
3501       if
3502       { pop }
3503       { SDict exch /pdf.rightboundary exch put }
3504       ifelse
3505       moveto
3506       currentpoint pop
3507       SDict /pdf.leftboundary known dup
3508       {
3509         SDict /pdf.leftboundary get 2 index gt
3510         { not }
3511         if
3512       }
3513       if
3514       { pop }
3515       { SDict exch /pdf.leftboundary exch put }
3516       ifelse
3517     }
3518     put
3519   }
3520   if
3521 }
3522 def
3523 /pdf.bordertracking.end
3524 {
3525   /a where { /a { moveto } put } if
3526   /x where { /x { 0 exch rmoveto } put } if
3527   SDict /pdf.leftboundary known
3528   { pdf.outerbox 0 pdf.leftboundary put }
3529   if
3530   SDict /pdf.rightboundary known
3531   { pdf.outerbox 2 pdf.rightboundary put }
3532   if
3533   SDict /pdf.bordertracking false put
3534 }
3535 def
3536 /pdf.bordertracking.endpage
3537 {
3538   pdf.bordertracking
3539   {

```

```

3540 pdf.bordertracking.end
3541 true setglobal
3542 pdf.globaldict
3543 /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3544 pdf.globaldict
3545 /pdf.brokenlink.skip pdf.baselineskip put
3546 pdf.globaldict
3547 /pdf.brokenlink.dict
3548 pdf.link.dict pdf.cvs put
3549 false setglobal
3550 mark pdf.link.dict cvx exec /Rect
3551 [
3552 pdf.llx
3553 pdf.lly
3554 pdf.outerbox 2 get pdf.linkmargin add
3555 currentpoint exch pop
3556 pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3557 ]
3558 /ANN pdf.pdfmark
3559 }
3560 if
3561 }
3562 def
3563 /pdf.bordertracking.continue
3564 {
3565 /pdf.link.dict pdf.globaldict
3566 /pdf.brokenlink.dict get def
3567 /pdf.outerbox pdf.globaldict
3568 /pdf.brokenlink.rect get def
3569 /pdf.baselineskip pdf.globaldict
3570 /pdf.brokenlink.skip get def
3571 pdf.globaldict dup dup
3572 /pdf.brokenlink.dict undef
3573 /pdf.brokenlink.skip undef
3574 /pdf.brokenlink.rect undef
3575 currentpoint
3576 /pdf.originy exch def
3577 /pdf.originx exch def
3578 /a where
3579 {
3580 /a
3581 {
3582 moveto
3583 SDict
3584 begin
3585 currentpoint pdf.originy ne exch
3586 pdf.originx ne or
3587 {
3588 pdf.save.link11
3589 /pdf.lly
3590 pdf.lly pdf.outerbox 1 get sub def
3591 pdf.bordertracking.begin
3592 }
3593 if

```

```

3594         end
3595     }
3596     put
3597 }
3598 if
3599 /x where
3600 {
3601     /x
3602     {
3603         0 exch rmoveto
3604         SDict
3605         begin
3606             currentpoint
3607             pdf.originy ne exch pdf.originx ne or
3608             {
3609                 pdf.save.linkll
3610                 /pdf.lly
3611                 pdf.lly pdf.outerbox 1 get sub def
3612                 pdf.bordertracking.begin
3613             }
3614             if
3615             end
3616         }
3617         put
3618     }
3619     if
3620 }
3621 def

```

(End of definition for pdf.bordertracking and others.)

```

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

```

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

```

3622 /pdf.breaklink
3623 {
3624     pop
3625     counttomark 2 mod 0 eq
3626     {
3627         counttomark /pdf.count exch def
3628         {
3629             pdf.count 0 eq { exit } if
3630             counttomark 2 roll
3631             1 index /Rect eq
3632             {
3633                 dup 4 array copy
3634                 dup dup
3635                 1 get
3636                 pdf.outerbox pdf.rect.ht
3637                 pdf.linkmargin 2 mul add sub
3638                 3 exch put
3639                 dup

```

```

3640 pdf.outerbox 2 get
3641 pdf.linkmargin add
3642 2 exch put
3643 dup dup
3644 3 get
3645 pdf.outerbox pdf.rect.ht
3646 pdf.linkmargin 2 mul add add
3647 1 exch put
3648 /pdf.currentrect exch def
3649 pdf.breaklink.write
3650 {
3651 pdf.currentrect
3652 dup
3653 pdf.outerbox 0 get
3654 pdf.linkmargin sub
3655 0 exch put
3656 dup
3657 pdf.outerbox 2 get
3658 pdf.linkmargin add
3659 2 exch put
3660 dup dup
3661 1 get
3662 pdf.baselineskip add
3663 1 exch put
3664 dup dup
3665 3 get
3666 pdf.baselineskip add
3667 3 exch put
3668 /pdf.currentrect exch def
3669 pdf.breaklink.write
3670 }
3671 1 index 3 get
3672 pdf.linkmargin 2 mul add
3673 pdf.outerbox pdf.rect.ht add
3674 2 index 1 get sub
3675 pdf.baselineskip div round cvi 1 sub
3676 exch
3677 repeat
3678 pdf.currentrect
3679 dup
3680 pdf.outerbox 0 get
3681 pdf.linkmargin sub
3682 0 exch put
3683 dup dup
3684 1 get
3685 pdf.baselineskip add
3686 1 exch put
3687 dup dup
3688 3 get
3689 pdf.baselineskip add
3690 3 exch put
3691 dup 2 index 2 get 2 exch put
3692 /pdf.currentrect exch def
3693 pdf.breaklink.write

```

```

3694             SDict /pdf.pdfmark.good false put
3695             exit
3696         }
3697         { pdf.count 2 sub /pdf.count exch def }
3698     ifelse
3699 }
3700 loop
3701 }
3702 if
3703 /ANN
3704 }
3705 def
3706 /pdf.breaklink.write
3707 {
3708     counttomark 1 sub
3709     index /_objdef eq
3710     {
3711         counttomark -2 roll
3712         dup wcheck
3713         {
3714             readonly
3715             counttomark 2 roll
3716         }
3717         { pop pop }
3718     ifelse
3719 }
3720 if
3721 counttomark 1 add copy
3722 pop pdf.currentrect
3723 /ANN pdfmark
3724 }
3725 def

```

(End of definition for pdf.breaklink and others.)

```

pdf.pdfmark
pdf.pdfmark.good
pdf.outerbox
pdf.baselineskip
pdf.pdfmark.dict

```

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.

```

3726 /pdf.pdfmark
3727 {
3728     SDict /pdf.pdfmark.good true put
3729     dup /ANN eq
3730     {
3731         pdf.pdfmark.store
3732         pdf.pdfmark.dict
3733         begin
3734             Subtype /Link eq
3735             currentdict /Rect known and
3736             SDict /pdf.outerbox known and
3737             SDict /pdf.baselineskip known and
3738             {
3739                 Rect 3 get

```



```

3740         pdf.linkmargin 2 mul add
3741         pdf.outerbox pdf.rect.ht add
3742         Rect 1 get sub
3743         pdf.baselineskip div round cvi 0 gt
3744         { pdf.breaklink }
3745         if
3746     }
3747     if
3748     end
3749     SDict /pdf.outerbox undef
3750     SDict /pdf.baselineskip undef
3751     currentdict /pdf.pdfmark.dict undef
3752 }
3753 if
3754 pdf.pdfmark.good
3755 { pdfmark }
3756 { cleartomark }
3757 ifelse
3758 }
3759 def
3760 /pdf.pdfmark.store
3761 {
3762     /pdf.pdfmark.dict 65534 dict def
3763     counttomark 1 add copy
3764     pop
3765     {
3766         dup mark eq
3767         {
3768             pop
3769             exit
3770         }
3771         {
3772             pdf.pdfmark.dict
3773             begin def end
3774         }
3775     } ifelse
3776 }
3777 loop
3778 }
3779 def

```

(End of definition for pdf.pdfmark and others.)

```

3780 </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.

Symbols	
<code>\</code>	1126
A	
<code>\AtBeginDvi</code>	56
B	
bool commands:	
<code>\bool_gset_false:N</code>	1211, 1230, 1253, 1275, 1291, 1392, 1631, 1667, 2413, 2459
<code>\bool_gset_true:N</code>	1209, 1278, 1390, 1646, 2406, 2412
<code>\bool_if:NTF</code>	66, 578, 1221, 1225, 1241, 1244, 1248, 1259, 1266, 1270, 1282, 1286, 1403, 1408, 1413, 1605, 1650, 1789, 1839, 1979, 2021, 2401, 2416, 2421, 2426
<code>\bool_if:nTF</code>	2635, 2888, 3124
<code>\bool_lazy_and:nnTF</code>	791, 2138, 3253, 3287
<code>\bool_lazy_any:nTF</code>	1828
<code>\bool_lazy_or:nnTF</code>	2014
<code>\bool_new:N</code>	1212, 1279, 1393, 1647, 2386, 2387
<code>\bool_set_false:N</code>	1801, 1943, 2045, 2209
box commands:	
<code>\box_dp:N</code>	217, 219, 267, 269, 324, 326, 373, 375, 377, 379, 2438, 2471, 2472, 2497
<code>\box_ht:N</code>	219, 269, 326, 377, 379, 1852, 2086, 2443, 2482, 2483, 2499
<code>\box_if_empty:N</code>	2532
<code>\box_move_down:nn</code>	2360, 2438
<code>\box_move_up:nn</code>	2230, 2362, 2443
<code>\box_new:N</code>	2256, 2350, 2351
<code>\box_set_dp:Nn</code>	1730
<code>\box_set_ht:Nn</code>	1729
<code>\box_set_wd:Nn</code>	281, 1728
<code>\box_use:N</code>	224, 242, 256, 272, 299, 313, 329, 345, 357, 408, 422, 441, 1343, 1538, 1731, 2391
<code>\box_wd:N</code>	218, 226, 268, 274, 325, 331, 374, 376, 1851, 2085
box internal commands:	
<code>__box_backend_clip:N</code>	206, 206, 261, 261, 318, 318, 362, 362
<code>\l__box_backend_cos_fp</code>	276
<code>__box_backend_rotate:Nn</code>	228, 228, 276, 276, 333, 333, 412, 412
<code>__box_backend_rotate_aux:Nn</code>	228, 229, 230, 276, 277, 278, 333, 334, 335
<code>__box_backend_scale:Nnn</code>	245, 245, 304, 304, 348, 348, 425, 425
<code>\l__box_backend_sin_fp</code>	276
C	
clist commands:	
<code>\clist_map_function:nN</code>	1299, 1423, 1674
color internal commands:	
<code>__color_backend:nnn</code>	1027, 1034, 1049, 1057, 1063
<code>__color_backend_cmyk:w</code>	1028
<code>\g__color_backend_colorant_prop</code>	544, 563, 566, 586, 827
<code>__color_backend_devicen_colorants:n</code>	545, 545, 747, 885
<code>__color_backend_devicen_colorants:w</code>	545, 553, 560, 568
<code>__color_backend_devicen_init:nnn</code>	734, 734, 852, 852, 1084, 1084
<code>__color_backend_devicen_init:w</code>	852, 861, 890, 894
<code>__color_backend_fill:n</code>	931, 931, 933, 934, 935, 957, 958, 960, 962, 963, 982, 991, 992, 994, 996, 997, 1008, 1017, 1018, 1020, 1022, 1023
<code>__color_backend_fill_cmyk:n</code>	931, 933, 957, 957, 991, 991, 1017, 1017
<code>__color_backend_fill_devicen:nn</code>	941, 951, 981, 985, 1007, 1011, 1078, 1080
<code>__color_backend_fill_gray:n</code>	931, 934, 957, 959, 991, 993, 1017, 1019
<code>__color_backend_fill_reset:</code>	953, 953, 987, 987, 1013, 1013, 1082, 1082
<code>__color_backend_fill_rgb:n</code>	931, 935, 957, 961, 991, 995, 1017, 1021
<code>__color_backend_fill_separation:nn</code>	941, 941, 951, 981, 981, 985, 1007, 1007, 1011, 1078, 1078, 1080
<code>\l__color_backend_fill_tl</code>	507, 519, 965, 979

_color_backend_iccbased_-	_color_backend_separation_-
device:nnn 914 , 914	init_CIELAB:nnn 576 , 688 , 758 , 805 , 830
_color_backend_iccbased_-	_color_backend_separation_-
init:nnn 753 , 753 , 896 , 896 , 1084 , 1085	init_CIELAB:nnnnnn 759
_color_backend_init_resource:n	_color_backend_separation_-
..... 788 , 788 , 817 , 888 , 912 , 927	init_count:n 576 , 635 , 638
_color_backend_reset:	_color_backend_separation_-
..... 488 , 503 , 511 , 523 ,	init_count:w ... 576 , 639 , 640 , 644
527 , 532 , 953 , 954 , 987 , 988 , 1013 , 1082	_color_backend_separation_-
_color_backend_rgb:w 1051	init_Device:Nn 576 , 620 , 622 , 624 , 625
_color_backend_select:n 488 , 489 , 491 , 493 ,	\l_color_backend_stack_int 449 , 521 , 524 , 966 , 978
495 , 496 , 527 , 527 , 529 , 530 , 531 , 573	_color_backend_stroke:n 931 , 936 , 938 ,
_color_backend_select:nn 511 , 512 , 514 , 516 , 517 , 784 939 , 940 , 957 , 970 , 972 , 974 , 975 , 984
_color_backend_select_cmyk:n .. 488 , 488 , 511 , 511 , 527 , 529	_color_backend_stroke_cmyk:n .. 931 ,
_color_backend_select_devicen:nn 572 , 574 , 756 , 757 , 778 , 786 938 , 957 , 969 , 991 , 1001 , 1027 , 1027
_color_backend_select_gray:n .. 488 , 490 , 511 , 513 , 527 , 530 , 537	_color_backend_stroke_cmyk:w .. 1027 , 1029
_color_backend_select_iccbased:nn 575 , 575 , 760 , 760 , 778 , 787	_color_backend_stroke_devicen:nn 941 ,
_color_backend_select_named:n . 488 , 492 , 534 , 534 952 , 981 , 986 , 1007 , 1012 , 1078 , 1081
_color_backend_select_rgb:n ... 488 , 494 , 511 , 515 , 527 , 531	_color_backend_stroke_gray:n .. 931 ,
_color_backend_select_separation:nn 572 , 572 , 574 , 939 , 957 , 971 , 991 , 1003 , 1027 , 1040
756 , 756 , 757 , 778 , 779 , 783 , 786 , 787	_color_backend_stroke_gray_-
_color_backend_separation_-	aux:n 1027 , 1044 , 1048
init:n 576 , 657 , 670	_color_backend_stroke_reset: .. 953 ,
_color_backend_separation_- 954 , 987 , 988 , 1013 , 1014 , 1082 , 1083
init:nn 805 , 815 , 819	_color_backend_stroke_rgb:n ... 931 ,
_color_backend_separation_- 940 , 957 , 973 , 991 , 1005 , 1027 , 1050
init:nnn 576 , 611 , 632	_color_backend_stroke_rgb:w ... 1027 , 1052
_color_backend_separation_-	_color_backend_stroke_separation:nn 941 , 946 , 952 , 981 , 983 ,
init:nnnn 576 , 634 , 646 986 , 1007 , 1009 , 1012 , 1078 , 1079 , 1081
_color_backend_separation_-	\l_color_backend_stroke_tl 507 , 520 , 967 , 977
init:nnnnn 576 ,	\g_color_model_int 583 , 592 , 740 ,
576 , 597 , 690 , 758 , 758 , 805 , 805 , 845 768 , 817 , 823 , 824 , 878 , 879 , 888 , 912
_color_backend_separation_-	\c_color_model_range_CIELAB_tl . 695 , 730 , 841 , 848
init:nw 576 , 661 , 672 , 686	color.sc 3399
_color_backend_separation_-	cs commands:
init:w 576 , 648 , 663 , 668	\cs_generate_variant:Nn 62 , 65 , 98 , 147 , 152 , 163 , 194 ,
_color_backend_separation_- 200 , 597 , 1158 , 1353 , 1547 , 1993 ,
init_/DeviceCMYK:nnn 576 2056 , 2076 , 2261 , 2282 , 2345 , 2839 ,
_color_backend_separation_- 2852 , 2962 , 2983 , 3013 , 3250 , 3338
init_/DeviceGray:nnn 576	
_color_backend_separation_-	
init_/DeviceRGB:nnn 576	
_color_backend_separation_-	
init_aux:nnnnnn 576 , 582 , 598	

\cs_gset:Npe ..	2647, 2651, 3129, 3134	1400, 1405, 1410, 1415, 1417, 1430,
\cs_gset_protected:Npn	3291	1435, 1437, 1439, 1441, 1443, 1445,
\cs_if_exist:Ntf		1447, 1449, 1460, 1485, 1497, 1509,
.....	27, 49, 1741, 2528, 2913, 2939	1521, 1528, 1550, 1556, 1561, 1566,
\cs_if_exist_p:N	792, 3254, 3288	1577, 1587, 1597, 1599, 1601, 1603,
\cs_if_exist_use:Ntf	38, 610	1634, 1636, 1641, 1643, 1645, 1648,
\cs_new:Npe		1669, 1680, 1693, 1695, 1697, 1699,
.....	545, 2674, 2709, 2853, 2864, 2931, 3151	1701, 1703, 1705, 1707, 1709, 1717,
\cs_new:Npn		1739, 1758, 1781, 1798, 1812, 1817,
.....	560, 619, 621, 623, 625, 632, 638,	1825, 1855, 1868, 1886, 1896, 1912,
.....	640, 646, 663, 670, 672, 890, 1304,	1931, 1940, 1948, 1960, 1966, 1969,
.....	1428, 1678, 1854, 2089, 2247, 2274,	1984, 1994, 2033, 2042, 2048, 2054,
.....	2346, 2348, 2381, 2553, 2653, 2654,	2057, 2064, 2077, 2082, 2090, 2097,
.....	2806, 2821, 2840, 2841, 2944, 2976,	2114, 2148, 2179, 2180, 2182, 2184,
.....	3014, 3016, 3032, 3056, 3137, 3138,	2186, 2192, 2198, 2206, 2212, 2215,
.....	3146, 3158, 3163, 3164, 3169, 3170	2217, 2228, 2259, 2262, 2264, 2267,
\cs_new_eq:NN	46, 56, 58,	2276, 2283, 2300, 2305, 2310, 2315,
.....	529, 530, 531, 574, 757, 786, 787,	2325, 2330, 2338, 2353, 2358, 2390,
.....	933, 934, 935, 938, 939, 940, 951,	2392, 2397, 2399, 2404, 2419, 2424,
.....	952, 953, 954, 985, 986, 987, 988,	2461, 2490, 2509, 2518, 2555, 2562,
.....	1011, 1012, 1013, 1080, 1081, 1082,	2588, 2593, 2621, 2633, 2645, 2649,
.....	1157, 1352, 1358, 1359, 1546, 1548,	2655, 2657, 2661, 2685, 2687, 2689,
.....	1549, 1555, 1755, 1756, 1769, 1771,	2700, 2720, 2730, 2753, 2767, 2777,
.....	1796, 1797, 1860, 1861, 1862, 1885,	2788, 2808, 2842, 2875, 2886, 2892,
.....	1910, 1927, 1928, 1937, 1938, 1939,	2920, 2954, 2956, 2963, 2965, 2969,
.....	1959, 1962, 1963, 1964, 2029, 2039,	2978, 2984, 2989, 2994, 2996, 2998,
.....	2040, 2041, 2195, 2196, 2204, 2205,	3006, 3019, 3035, 3037, 3054, 3058,
.....	2214, 2244, 2245, 2246, 2250, 2391	3060, 3082, 3087, 3120, 3122, 3127,
\cs_new_protected:Npe		3132, 3139, 3141, 3145, 3147, 3148,
.....	576, 1063, 2903, 2960, 3039	3149, 3150, 3152, 3153, 3154, 3155,
\cs_new_protected:Npn		3156, 3157, 3159, 3160, 3161, 3162,
.....	47, 53, 60, 63, 71, 77, 82,	3165, 3166, 3167, 3168, 3171, 3172,
.....	84, 88, 99, 109, 119, 128, 137, 150,	3175, 3194, 3201, 3207, 3212, 3217,
.....	153, 155, 157, 161, 166, 175, 185,	3224, 3231, 3266, 3271, 3293, 3303,
.....	195, 206, 228, 230, 245, 261, 276,	3309, 3315, 3341, 3343, 3345, 3347
.....	278, 304, 318, 333, 335, 348, 362,	\cs_set_eq:NN
.....	412, 425, 452, 466, 476, 488, 490,	2549, 2550
.....	492, 494, 496, 503, 511, 513, 515,	\cs_set_protected:Npn
.....	517, 523, 527, 532, 534, 572, 575,	2152
.....	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, 1159, 1164, 1169,	
.....	1171, 1173, 1181, 1189, 1198, 1208,	
.....	1210, 1213, 1215, 1232, 1237, 1255,	
.....	1277, 1280, 1293, 1306, 1311, 1313,	
.....	1315, 1317, 1319, 1321, 1323, 1325,	
.....	1330, 1354, 1356, 1360, 1365, 1370,	
.....	1380, 1389, 1391, 1394, 1396, 1398,	

D

dim commands:

\dim_compare:nNnTF	2128, 2133
\dim_compare_p:nNn	2139, 2140
\dim_eval:n	
.....	2356, 2591, 2669, 2670, 2671,
.....	2728, 2763, 2764, 2765, 3026, 3027,
.....	3028, 3059, 3085, 3183, 3184, 3187
\dim_gset:Nn	3196, 3197
\dim_max:nn	2469, 2480
\dim_set:Nn	
.....	1851, 1852, 2085, 2086, 2124, 2125
\dim_set_eq:NN	2190
\dim_to_decimal:n ..	373, 374, 375,
.....	376, 377, 379, 1559, 1564, 1570,
.....	1571, 1572, 1573, 1582, 1583, 1584,

1675, 1694, 2237, 2238, 2467, 2478,
 2496, 2497, 2498, 2499, 2503, 2559
 \dim_to_decimal_in_bp:n
 217, 218, 219, 267, 268, 269,
 324, 325, 326, 1177, 1178, 1185,
 1186, 1193, 1194, 1202, 1203, 1204,
 1301, 1305, 1309, 1363, 1368, 1374,
 1375, 1376, 1384, 1385, 1425, 1429,
 1433, 1679, 1763, 1764, 1765, 1766,
 1953, 1954, 1955, 1956, 2008, 2009,
 2010, 2011, 2222, 2223, 2224, 2225
 \dim_zero:N 2122, 2123
 \c_max_dim
 .. 2124, 2125, 2128, 2133, 2139, 2140
 draw internal commands:
 __draw_align_currentpoint... .. 36
 __draw_backend_add_to_path:n ...
 1556,
 1558, 1563, 1568, 1579, 1587, 1602
 __draw_backend_begin:
 .. 1159, 1159, 1354, 1354, 1550, 1550
 __draw_backend_box_use:Nnnnn ...
 .. 32, 1330, 1330, 1528, 1528, 1717, 1717
 __draw_backend_cap_but:
 .. 1293, 1313, 1417, 1437, 1669, 1697
 __draw_backend_cap_rectangle: ..
 .. 1293, 1317, 1417, 1441, 1669, 1701
 __draw_backend_cap_round:
 .. 1293, 1315, 1417, 1439, 1669, 1699
 __draw_backend_clip:
 .. 1213, 1277, 1394, 1410, 1601, 1645
 __draw_backend_closepath:
 1213, 1213,
 1234, 1394, 1394, 1601, 1601, 1638
 __draw_backend_closestroke: ...
 .. 1213, 1232, 1394, 1398, 1601, 1636
 __draw_backend_cm:nnnn
 1325, 1325, 1338, 1339, 1340,
 1449, 1449, 1532, 1709, 1709, 1720
 __draw_backend_cm_aux:nnnn
 1449, 1456, 1460
 __draw_backend_cm_decompose:nnnnN
 1455, 1484, 1485
 __draw_backend_cm_decompose_-
 auxi:nnnnN 1484, 1489, 1497
 __draw_backend_cm_decompose_-
 auxii:nnnnN 1484, 1501, 1509
 __draw_backend_cm_decompose_-
 auxiii:nnnnN 1484, 1513, 1521
 __draw_backend_curveto:nnnnnn ..
 .. 1173, 1198, 1360, 1370, 1556, 1577
 __draw_backend_dash:n
 1293, 1299, 1304,
 1417, 1423, 1428, 1669, 1674, 1678
 __draw_backend_dash_aux:nn
 1669, 1673, 1680
 __draw_backend_dash_pattern:nn ..
 .. 1293, 1293, 1417, 1417, 1669, 1669
 __draw_backend_discardpath: ...
 .. 1213, 1280, 1394, 1415, 1601, 1648
 __draw_backend_end:
 .. 1159, 1164, 1354, 1356, 1550, 1555
 __draw_backend_evenodd_rule: ...
 .. 1208, 1208, 1389, 1389, 1597, 1597
 __draw_backend_fill:
 .. 1213, 1237, 1394, 1400, 1601, 1641
 __draw_backend_fillstroke:
 .. 1213, 1255, 1394, 1405, 1601, 1643
 __draw_backend_join_bevel:
 .. 1293, 1323, 1417, 1447, 1669, 1707
 __draw_backend_join_miter:
 .. 1293, 1319, 1417, 1443, 1669, 1703
 __draw_backend_join_round:
 .. 1293, 1321, 1417, 1445, 1669, 1705
 __draw_backend_lineto:nn
 .. 1173, 1181, 1360, 1365, 1556, 1561
 __draw_backend_linewidth:n
 .. 1293, 1306, 1417, 1430, 1669, 1693
 __draw_backend_literal:n
 1157, 1157, 1158, 1162,
 1166, 1170, 1172, 1175, 1183, 1191,
 1200, 1214, 1217, 1218, 1219, 1220,
 1223, 1229, 1239, 1246, 1252, 1257,
 1262, 1263, 1264, 1265, 1268, 1274,
 1284, 1290, 1295, 1308, 1312, 1314,
 1316, 1318, 1320, 1322, 1324, 1327,
 1332, 1333, 1334, 1335, 1336, 1337,
 1341, 1342, 1344, 1345, 1346, 1347,
 1348, 1352, 1352, 1353, 1362, 1367,
 1372, 1382, 1395, 1397, 1399, 1402,
 1407, 1412, 1416, 1419, 1432, 1436,
 1438, 1440, 1442, 1444, 1446, 1448,
 1546, 1546, 1547, 1608, 1627, 1653
 __draw_backend_miterlimit:n ...
 .. 1293, 1311, 1417, 1435, 1669, 1695
 __draw_backend_moveto:nn
 .. 1173, 1173, 1360, 1360, 1556, 1556
 __draw_backend_nonzero_rule: ...
 .. 1208, 1210, 1389, 1391, 1597, 1599
 __draw_backend_path:n
 1601, 1603, 1635, 1642, 1644
 \g__draw_backend_path_int 1616, 1633
 \g__draw_backend_path_tl
 1556, 1612, 1628, 1630, 1657
 __draw_backend_rectangle:nnnn ..
 .. 1173, 1189, 1360, 1380, 1556, 1566
 __draw_backend_scope_begin: 1169,
 1169, 1355, 1358, 1358, 1548, 1548

`__draw_backend_scope_end:` [1169](#),
[1171](#), [1357](#), [1358](#), [1359](#), [1548](#), [1549](#)
`__draw_backend_stroke:` [1213](#), [1215](#),
[1235](#), [1394](#), [1396](#), [1601](#), [1634](#), [1639](#)
`\g__draw_draw_clip_bool` .. [1213](#), [1601](#)
`\g__draw_draw_eor_bool`
... [1208](#), [1225](#), [1241](#), [1248](#), [1259](#),
[1270](#), [1286](#), [1389](#), [1403](#), [1408](#), [1413](#)
`\g__draw_draw_path_int` [1601](#)
`\g__draw_path_tl` [1666](#)

E

`\errmessage` [38](#)
`\evensidemargin` [2436](#)
exp commands:
`\exp_after:wN` [2095](#)
`\exp_args:Ne` [580](#), [634](#), [815](#),
[1819](#), [1874](#), [1876](#), [1900](#), [1902](#), [2312](#),
[2327](#), [2432](#), [2590](#), [3084](#), [3209](#), [3268](#)
`\exp_args:Nf` [1298](#), [1422](#), [2355](#)
`\exp_args:Nne` [3009](#)
`\exp_args:NNf` [229](#), [277](#), [334](#)
`\exp_not:N` . [547](#), [553](#), [554](#), [555](#), [580](#),
[582](#), [583](#), [586](#), [587](#), [592](#), [2676](#), [2678](#),
[2681](#), [2711](#), [2713](#), [2716](#), [2855](#), [2857](#),
[2860](#), [2866](#), [2868](#), [2871](#), [2908](#), [2909](#),
[2915](#), [2916](#), [2935](#), [2940](#), [3041](#), [3046](#)
`\exp_not:n` [48](#), [96](#), [107](#), [145](#),
[904](#), [2303](#), [2308](#), [2584](#), [2825](#), [2826](#),
[2840](#), [2841](#), [2987](#), [2992](#), [3003](#), [3064](#)
`\ExplBackendFileDate` [1](#)

F

file commands:
`\file_compare_timestamp:nNnTF` . [1888](#)
`\file_parse_full_name:nNNN` [1870](#), [1898](#)
`\fmtversion` [51](#)
fp commands:
`\fp_compare:nNnTF`
. [236](#), [283](#), [289](#), [341](#), [1465](#), [1478](#), [1523](#)
`\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](#), [1467](#), [1472](#),
[1473](#), [1480](#), [1490](#), [1491](#), [1492](#), [1493](#),
[1502](#), [1503](#), [1504](#), [1505](#), [1514](#), [1515](#),
[1516](#), [1517](#), [2581](#), [2750](#), [3078](#), [3210](#),
[3220](#), [3227](#), [3269](#), [3306](#), [3313](#), [3348](#)
`\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](#), [1465](#), [1478](#)

G

graphics commands:
`\l_graphics_search_ext_seq`
..... [1751](#), [1774](#), [1920](#), [2108](#)
graphics internal commands:
`\l__graphics_attr_tl` [1780](#),
[1785](#), [1802](#), [1814](#), [1821](#), [1823](#), [1858](#)
`__graphics_backend_dequote:w` ...
..... [1781](#), [1820](#), [1854](#)
`\l__graphics_backend_dir_str` . [1863](#)
`\l__graphics_backend_ext_str` . [1863](#)
`__graphics_backend_get_pagecount:n`
..... [1770](#), [1771](#), [1912](#), [1912](#),
[2027](#), [2029](#), [2097](#), [2097](#), [2249](#), [2250](#)
`__graphics_backend_getbb_auxi:n`
..... [1781](#), [1794](#), [1810](#), [1812](#)
`__graphics_backend_getbb_-`
`auxi:nN` [2033](#), [2037](#), [2046](#), [2048](#)
`__graphics_backend_getbb_-`
`auxii:n` [1781](#), [1815](#), [1817](#)
`__graphics_backend_getbb_-`
`auxiii:nnN` .. [2033](#), [2051](#), [2054](#), [2056](#)
`__graphics_backend_getbb_-`
`auxiiii:n` [1781](#), [1819](#), [1825](#)
`__graphics_backend_getbb_-`
`auxiiii:nnNnn` . [2033](#), [2052](#), [2055](#), [2057](#)
`__graphics_backend_getbb_-`
`auxiv:nnNnn` . [2033](#), [2060](#), [2064](#), [2076](#)
`__graphics_backend_getbb_-`
`auxv:nNnn` .. [2033](#), [2061](#), [2068](#), [2077](#)
`__graphics_backend_getbb_-`
`auxvi:nNnn` [2080](#), [2082](#)
`__graphics_backend_getbb_bmp:n` .
..... [1925](#), [1939](#), [2033](#), [2041](#)
`__graphics_backend_getbb_eps:n` .
..... [1753](#), [1755](#), [1863](#),
[1868](#), [1885](#), [1925](#), [1927](#), [2193](#), [2195](#)
`__graphics_backend_getbb_eps:nm`
..... [1863](#)
`__graphics_backend_getbb_eps:nn`
..... [1874](#), [1886](#)
`__graphics_backend_getbb_jpeg:n`
..... [1781](#), [1796](#),
[1925](#), [1937](#), [2033](#), [2039](#), [2198](#), [2204](#)
`__graphics_backend_getbb_jpg:n` .
[1781](#), [1781](#), [1796](#), [1797](#), [1925](#), [1931](#),
[1937](#), [1938](#), [1939](#), [2033](#), [2033](#), [2039](#),
[2040](#), [2041](#), [2198](#), [2198](#), [2204](#), [2205](#)
`__graphics_backend_getbb_-`
`pagebox:w` .. [2033](#), [2072](#), [2089](#), [2095](#)
`__graphics_backend_getbb_pdf:n` .
..... [1781](#), [1798](#), [1894](#),
[1925](#), [1940](#), [2033](#), [2042](#), [2206](#), [2206](#)

__graphics_backend_getbb_png:n .	1863, 1910, 1948, 1959, 2212, 2214
..... 1781, 1797,	
1925, 1938, 2033, 2040, 2198, 2205	
__graphics_backend_getbb_ps:n . .	__graphics_backend_include_-
..... 1753, 1756,	svg:n . . 2228, 2228, 2244, 2245, 2246
1863, 1885, 1925, 1928, 2193, 2196	__graphics_backend_loaded:n . . .
__graphics_backend_getbb_svg:n .	1739, 1739, 1751, 1753, 1770, 1774,
..... 2114, 2114	1920, 1925, 2028, 2108, 2193, 2249
__graphics_backend_getbb_svg_-	\l__graphics_backend_name_str . 1863
auxi:nNn . . . 2114, 2130, 2135, 2148	__graphics_bb_restore:nTF
__graphics_backend_getbb_svg_- 1814, 2079, 2116
auxii:w 2114, 2152, 2174, 2179	__graphics_bb_save:n 1823, 2087, 2143
__graphics_backend_getbb_svg_-	\l__graphics_decodearray_str . . .
auxiii:Nw 2114, 2162, 2180 1787, 1788,
__graphics_backend_getbb_svg_-	1800, 1831, 1837, 1838, 1942, 1977,
auxiv:Nw 2114, 2165, 2182	1978, 2016, 2019, 2020, 2044, 2208
__graphics_backend_getbb_svg_-	__graphics_extract_bb:n
auxv:Nw 2114, 2166, 2184 1935, 1944, 2202, 2210
__graphics_backend_getbb_svg_-	\l__graphics_final_name_str . . 1893
auxvi:Nn 2114, 2181, 2183, 2185, 2186	__graphics_get_pagecount:n
__graphics_backend_getbb_svg_- 1771, 2029, 2250
auxvii:w 2114, 2188, 2192	\l__graphics_internal_box
__graphics_backend_include:nn 1849, 1851, 1852, 2084, 2085, 2086
..... 2212, 2213, 2216, 2217	\l__graphics_internal_dim 2189, 2190
__graphics_backend_include_-	\l__graphics_internal_ior
auxi:nn 1948, 1961, 1967, 1969 2118, 2119, 2126, 2145
__graphics_backend_include_-	\l__graphics_interpolate_bool . . .
auxii:nnn . . 1948, 1971, 1984, 1993 1789, 1801, 1830, 1839,
__graphics_backend_include_-	1943, 1979, 2015, 2021, 2045, 2209
auxiii:nnn 1948, 1991, 1994	\l__graphics_llx_dim
__graphics_backend_include_- 1763, 1953, 2008, 2122, 2222
bmp:n 1948, 1964	\l__graphics_lly_dim
__graphics_backend_include_- 1764, 1954, 2009, 2123, 2223
dequote:w 2228, 2239, 2247	\l__graphics_page_int
__graphics_backend_include_- 1783, 1805, 1806, 1844,
eps:n 1758,	1845, 1933, 1975, 1976, 2002, 2003,
1758, 1769, 1863, 1896, 1910,	2035, 2050, 2051, 2093, 2094, 2200
1948, 1948, 1959, 2212, 2212, 2214	\l__graphics_pagebox_tl
__graphics_backend_include_- 55, 1784, 1804,
jpeg:n . 1855, 1860, 1962, 2228, 2245	1846, 1847, 1934, 1973, 1974, 2004,
__graphics_backend_include_-	2006, 2036, 2059, 2060, 2095, 2201
jpg:n 1855,	\l__graphics_pdf_str
1855, 1860, 1861, 1862, 1948,	.. 1791, 1792, 1807, 1808, 1832, 1841
1960, 1962, 1963, 1964, 2228, 2246	__graphics_read_bb:n
__graphics_backend_include_-	.. 1755, 1756, 1927, 1928, 2195, 2196
jps:n 1948	\g__graphics_track_int
__graphics_backend_include_- 1947, 1996, 1997
pdf:n 1855, 1861, 1900,	\l__graphics_urx_dim
1948, 1966, 2090, 2090, 2212, 2215	.. 1765, 1851, 1955, 2010, 2085,
__graphics_backend_include_-	2124, 2128, 2131, 2139, 2224, 2237
png:n	\l__graphics_ury_dim
.. 1855, 1862, 1948, 1963, 2228, 2244	1766, 1852, 1956, 2011, 2086, 2125,
__graphics_backend_include_ps:n	2133, 2136, 2140, 2225, 2230, 2238
..... 1758, 1769,	group commands:
	\group_begin: 172, 191
	\group_end: 180

\group_insert_after:N ... 3285, 3335

\ior_str_map_inline:Nn ... 2126

H

hbox commands:

\hbox:n ... 2232, 2361, 2364,
2439, 2445, 2598, 2605, 3092, 3103
\hbox_overlap_right:n ... 224,
256, 272, 313, 329, 357, 441, 1343, 1538
\hbox_set:Nn .. 1849, 2084, 2431, 2463
\hbox_set:Nw ... 2414
\hbox_set_end: ... 2429
\hbox_unpack:N ... 2550

hook commands:

\hook_gput_code:nnn .. 54, 1741, 1743

I

int commands:

\int_compare:nNnTF ...
... 1805, 1844, 1975, 2002,
2050, 2093, 2522, 2623, 2906, 2934
\int_const:Nn ... 454, 1821,
1915, 1997, 2099, 2270, 2797, 2972
\int_eval:n 474, 484, 630, 639, 652,
654, 658, 671, 2647, 2651, 2884,
2909, 2916, 2929, 3121, 3129, 3134
\int_gincr:N ... 198,
364, 1607, 1652, 1996, 2269, 2340,
2371, 2448, 2971, 3008, 3021, 3041
\int_gset:Nn ... 173, 192, 2511
\int_gset_eq:NN 181, 2372, 2449, 3022
\int_if_exist:NTF ... 1986
\int_if_odd:nTF ... 2434
\int_max:nn ... 2101
\int_new:N ... 164,
165, 411, 449, 1633, 1947, 2266,
2352, 2383, 2385, 2967, 3018, 3034
\int_set_eq:NN ... 169, 188, 2523
\int_step_function:nnnN ... 656
\int_use:N ...
. 366, 397, 583, 592, 740, 768, 817,
823, 824, 878, 879, 888, 912, 1610,
1616, 1623, 1655, 1663, 1806, 1845,
1858, 1916, 1976, 1989, 2001, 2003,
2094, 2102, 2275, 2342, 2347, 2375,
2382, 2453, 2554, 2807, 2817, 2977,
3010, 3015, 3025, 3033, 3046, 3057
\int_value:w ...
... 2676, 2711, 2855, 2866, 2884
\int_zero:N ... 1783, 1933, 2035, 2200

ior commands:

\ior_close:N ... 2145
\ior_if_eof:NTF ... 2119
\ior_map_break: ... 2141
\ior_open:Nn ... 2118

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, 3177
\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, 1161, 1167,
1462, 1469, 1475, 1535, 1540, 1760,
1950, 1988, 1998, 2219, 2234, 2961,
3059, 3121, 3125, 3130, 3135, 3179
__kernel_backend_literal_page:n
... 99, 99,
109, 153, 153, 2955, 2957, 3140, 3142
__kernel_backend_literal_pdf:n .
... 88, 88, 98, 150, 150,
152, 264, 321, 1352, 3279, 3296, 3329
__kernel_backend_literal_-
postscript:n ... 60,
60, 62, 74, 75, 79, 210, 211, 213,
214, 222, 234, 249, 1157, 2625, 2637
__kernel_backend_literal_svg:n .
. 161, 161, 163, 168, 179, 187, 197,
365, 367, 384, 762, 1546, 1721, 1732
__kernel_backend_matrix:n
... 137, 137, 147, 286, 307, 1452
__kernel_backend_postscript:n ..
... 63, 63, 65,
500, 1002, 1004, 1006, 1010, 2260,
2317, 2332, 2361, 2367, 2407, 2439,
2446, 2450, 2464, 2492, 2536, 2543,
2549, 2557, 2564, 2598, 2605, 3233
__kernel_backend_scope:n
... 166, 195, 200, 394,
399, 1065, 1553, 1598, 1600, 1620,
1660, 1682, 1694, 1696, 1698, 1700,
1702, 1704, 1706, 1708, 1711, 3348
__kernel_backend_scope_begin: ..
82, 82, 119, 119, 155, 155, 166, 166,
208, 232, 247, 263, 280, 306, 320,
337, 350, 1358, 1530, 1548, 1552, 1719
__kernel_backend_scope_begin:n .
... 166, 185, 194, 386, 414, 427

_kernel_backend_scope_end:	...
.....	82 , 84 , 119 , 128 ,
	155 , 157 , 166 , 175 , 225 , 243 , 257 ,
	273 , 300 , 314 , 330 , 346 , 358 , 409 ,
	423 , 442 , 1359 , 1542 , 1549 , 1555 , 1733
\g_kernel_backend_scope_int	...
.....	164 , 171 , 173 , 178 , 182 , 190 , 192 , 198
\l_kernel_backend_scope_int	...
.....	164 , 170 , 183 , 189
\g_kernel_clip_path_int
	362 , 1607 , 1610 , 1623 , 1652 , 1655 , 1663
_kernel_color_backend_stack-	
init:Nnn 452 , 452 , 3258
_kernel_color_backend_stack-	
pop:n 466 , 476 , 524 , 3300
_kernel_color_backend_stack-	
push:nn
..	466 , 466 , 521 , 966 , 978 , 3282 , 3332
_kernel_dependency_version-	
check:Nn 1
_kernel_dependency_version-	
check:nn 27 , 29
_kernel_file_name_quote:n
.....	1876 , 1902
_kernel_kern:n
.....	2366 , 2368 , 2597 , 2601 ,
	2604 , 2608 , 3091 , 3099 , 3102 , 3118
L	
lua commands:	
\lua_load_module:n 1151
M	
\MessageBreak 40
mode commands:	
\mode_if_horizontal:TF	... 2513 , 2520
\mode_if_math:TF 2411
msg commands:	
\msg_error:nnn 538 , 2120
\msg_new:nnn 540
O	
\oddsidemargin 2435
opacity internal commands:	
_opacity_backend:nn
.....	3341 , 3342 , 3344 , 3346 , 3347
_opacity_backend:nnn	... 3207 ,
.....	3214 , 3215 , 3219 , 3226 , 3231 , 3250
_opacity_backend_fill:n
..	3207 , 3217 , 3303 , 3303 , 3341 , 3343
_opacity_backend_fill_stroke:nn
.....	3305 , 3311 , 3315 , 3338
\l_opacity_backend_fill_tl
.....	3264 , 3273 , 3312 , 3320
P	
pdf commands:	
\pdf_object_if_exist:nTF	832 , 898 , 916
\pdf_object_new:n
.....	823 , 834 , 878 , 900 , 918
\pdf_object_ref:n
.....	780 , 847 , 911 , 926 , 944 , 949
\pdf_object_ref_last:
.....	800 , 825 , 828 , 884
\pdf_object_unnamed_write:nn	...
.....	807 , 854 , 910 , 925
\pdf_object_write:nnn
.....	824 , 835 , 879 , 901 , 919
pdf internal commands:	
_pdf_backend:n	. 2960 , 2960 , 2962 ,
.....	2964 , 2966 , 2986 , 2991 , 3000 , 3023 ,
.....	3042 , 3055 , 3062 , 3094 , 3095 , 3105
_pdf_backend_annotation:nnnn	..
.....	2353 , 2353 ,
.....	2661 , 2661 , 3019 , 3019 , 3145 , 3145
_pdf_backend_annotation-	
aux:nnnn 2355 , 2358
\g_pdf_backend_annotation_int	..
..	2352 , 2372 , 2382 , 3018 , 3022 , 3033
_pdf_backend_annotation_last:	.
.....	2381 , 2381 ,
.....	2674 , 2674 , 3032 , 3032 , 3146 , 3146
_pdf_backend_bdc:nn	2655 , 2655 ,
.....	2954 , 2954 , 3139 , 3139 , 3171 , 3171
_pdf_backend_catalog_gput:nn	..
.....	2262 , 2262 ,
.....	2767 , 2767 , 2963 , 2963 , 3155 , 3155
_pdf_backend_compress_objects:n
.....	2621 , 2633 ,
.....	2875 , 2886 , 3120 , 3122 , 3165 , 3166
_pdf_backend_compresslevel:n	..
.....	2621 , 2621 ,
.....	2875 , 2875 , 3120 , 3120 , 3165 , 3165

\l__pdf_backend_content_box	2350, 2414, 2438, 2441, 2443, 2472, 2483	__pdf_backend_link_outerbox:n	2392, 2432, 2490
__pdf_backend_destination:nn	2562, 2562, 2730, 2730, 3060, 3060, 3153, 3153	\g__pdf_backend_link_sf_int	2385, 2511, 2522, 2523
__pdf_backend_destination:nnnn	2562, 2588, 2730, 2753, 3060, 3082, 3153, 3154	__pdf_backend_link_sf_restore:	2392, 2415, 2458, 2518
__pdf_backend_destination_-aux:nnnn	2562, 2590, 2593, 3060, 3084, 3087	__pdf_backend_link_sf_save:	2392, 2410, 2428, 2509
__pdf_backend_emc:	2655, 2657, 2954, 2956, 3139, 3141, 3171, 3172	\l__pdf_backend_model_box	2351, 2431, 2463, 2471, 2482, 2497, 2499
__pdf_backend_info_gput:nn	2262, 2264, 2767, 2777, 2963, 2965, 3155, 3156	__pdf_backend_objcompresslevel:n	2875, 2889, 2890, 2892
__pdf_backend_link:nw	2392	\g__pdf_backend_object_int	2266, 2269, 2272, 2340, 2342, 2347, 2371, 2372, 2375, 2448, 2449, 2967, 2971, 2974, 3008, 3010, 3015, 3021, 3022, 3025
__pdf_backend_link_aux:nw	2392	__pdf_backend_object_last:	2346, 2346, 2853, 2853, 3014, 3014, 3157, 3163
__pdf_backend_link_begin:n	3035, 3036, 3038, 3039	__pdf_backend_object_new:n	2267, 2267, 2788, 2788, 2969, 2969, 3157
__pdf_backend_link_begin:nnnw	2685, 2686, 2688, 2689, 3147, 3149	__pdf_backend_object_new:nn	3157
__pdf_backend_link_begin:nw	2394, 2398, 2399	__pdf_backend_object_now:nn	2338, 2338, 2345, 2842, 2842, 2852, 3006, 3006, 3013, 3157, 3161, 3162
__pdf_backend_link_begin_aux:nw	2402, 2404	\g__pdf_backend_object_prop	2787, 2967
__pdf_backend_link_begin_-goto:nnw	2392, 2392, 2685, 2685, 3035, 3035, 3147, 3147	__pdf_backend_object_ref:n	2267, 2274, 2279, 2788, 2806, 2969, 2976, 2981, 3157, 3158
__pdf_backend_link_begin_-user:nnw	2392, 2397, 2685, 2687, 3035, 3037, 3147, 3148	__pdf_backend_object_write:nn	2808, 2819, 2821, 2850, 3157
\g__pdf_backend_link_bool	2387, 2401, 2406, 2421, 2459	__pdf_backend_object_write:nnn	2276, 2276, 2282, 2808, 2808, 2839, 2978, 2978, 2983, 3157, 3159, 3160
\g__pdf_backend_link_dict_tl	2384, 2409, 2454	__pdf_backend_object_write_-array:nn	2276, 2300, 2978, 2984
__pdf_backend_link_end:	2392, 2419, 2685, 2700, 3035, 3054, 3147, 3150	__pdf_backend_object_write_-aux:nnn	2276, 2278, 2283, 2341
__pdf_backend_link_end_aux:	2392, 2422, 2424	__pdf_backend_object_write_-dict:nn	2276, 2305, 2978, 2989
\g__pdf_backend_link_int	2383, 2449, 2453, 2554, 3034, 3041, 3046, 3057	__pdf_backend_object_write_-fstream:nn	2276, 2310, 2978, 2994
__pdf_backend_link_last:	2553, 2553, 2709, 2709, 3056, 3056, 3151, 3151	__pdf_backend_object_write_-fstream:nnn	2313, 2315
__pdf_backend_link_margin:n	2555, 2555, 2720, 2720, 3058, 3058, 3152, 3152	__pdf_backend_object_write_-stream:nn	2276, 2325, 2978, 2996
\g__pdf_backend_link_math_bool	2386, 2412, 2413, 2416, 2426	__pdf_backend_object_write_-stream:nnn	2276, 2328, 2330
__pdf_backend_link_minima:	2392, 2430, 2461	__pdf_backend_object_write_-stream:nnnn	2978, 2995, 2997, 2998
		__pdf_backend_pageobject_ref:n	2348, 2348,

2864 , 2864 , 3016 , 3016 , 3157 , 3164	pdf.linkht.pad	3410
_pdf_backend_pagesize_gset:nn	pdf.linkmargin	3410
.. 3175 , 3175 , 3194 , 3194 , 3201 , 3201	pdf.llx	3413
_pdf_backend_pdfmark:n	pdf.lly	3413
.. 2259 , 2259 , 2261 , 2263 , 2265 , 2285 , 2302 , 2307 , 2373 , 2565 , 2609 , 2656 , 2658	pdf.originx	3484
_pdf_backend_version_major:...	pdf.originy	3484
... 2647 , 2653 , 2653 , 2931 , 2931 , 3129 , 3130 , 3137 , 3137 , 3169 , 3169	pdf.outerbox	3726
_pdf_backend_version_major_gset:n	pdf.pdfmark	3726
... 2645 , 2645 , 2903 , 2903 , 3127 , 3127 , 3167 , 3167	pdf.pdfmark.dict	3726
_pdf_backend_version_minor:...	pdf.pdfmark.good	3726
... 2651 , 2653 , 2654 , 2931 , 2944 , 3134 , 3135 , 3137 , 3138 , 3169 , 3170	pdf.pt.dvi	3406
_pdf_backend_version_minor_gset:n	pdf.rect	3413
... 2645 , 2649 , 2903 , 2920 , 3127 , 3132 , 3167 , 3168	pdf.rect.ht	3406
_pdf_breaklink_pdfmark_tl	pdf.rightboundary	3484
... 2388 , 2456 , 2548	pdf.save.linkll	3413
_pdf_breaklink_postscript:n	pdf.save.linkur	3413
... 2390 , 2390 , 2440 , 2442 , 2549	pdf.save.ll	3413
_pdf_breaklink_usebox:N	pdf.save.ur	3413
... 2391 , 2391 , 2441 , 2550	pdf.tmpa	3449
_pdf_exp_not_i:nn	pdf.tmpb	3449
... 2808 , 2829 , 2834 , 2840	pdf.tmpc	3449
_pdf_exp_not_ii:nn	pdf.tmpd	3449
... 2808 , 2830 , 2835 , 2841	pdf.urx	3413
_pdf_internal_box	pdf.ury	3413
... 2256	pdfmanagement commands:	
pdf.baselineskip	\pdfmanagement_add:nnn	...
pdf.bordertracking	... 797 , 3261 , 3275 , 3322 , 3325	
pdf.bordertracking.begin	\pdfmanagement_if_active_p:	...
pdf.bordertracking.continue	... 792 , 793 , 3254 , 3255 , 3288 , 3289	
pdf.bordertracking.end	peek commands:	
pdf.bordertracking.endpage	\peek_meaning:NtF	... 2161 , 2164
pdf.breaklink	\peek_remove_spaces:n	... 2159
pdf.breaklink.write	prg commands:	
pdf.brokenlink.dict	\prg_replicate:nn	...
pdf.brokenlink.rect	... 177 , 628 , 649 , 659 , 860	
pdf.brokenlink.skip	prop commands:	
pdf.count	\prop_gput:Nnn	... 586 , 827
pdf.currentrect	\prop_if_in:NnTF	... 563
pdf.cvs	\prop_item:Nn	... 566
pdf.dest.anchor	\prop_new:N	... 544 , 2787 , 2968
pdf.dest.point	\ProvidesExplFile	... 2
pdf.dest.x		
pdf.dest.y		
pdf.dest2device		
pdf.dev.x		
pdf.dev.y		
pdf.dvi.pt		
pdf.globaldict		
pdf.leftboundary		
pdf.linkdp.pad		

scan internal commands:		<code>\tex_pdfextension:D</code>	91, 102, 112, 122, 131, 140, 469, 479, 2664, 2692, 2703, 2733, 2756, 2770, 2780, 2791, 2811, 2845
<code>\s_color_stop</code>	639, 640, 644, 648, 661, 664, 668, 672, 686, 861, 890, 894, 1028, 1030, 1051, 1053	<code>\tex_pdffeedback:D</code>	457, 2678, 2713, 2800, 2857, 2868
<code>\s_graphics_stop</code>	1820, 1854, 2154, 2169, 2176, 2180, 2182, 2184, 2239, 2247	<code>\tex_pdfinfo:D</code>	2783
separation	<u>3400</u>	<code>\tex_pdflastannot:D</code>	2681
seq commands:		<code>\tex_pdflastlink:D</code>	2716
<code>\seq_set_from_clist:Nn</code>	1752, 1776, 1922, 2110	<code>\tex_pdflastobj:D</code>	2803, 2860
shipout commands:		<code>\tex_pdflastximage:D</code>	1822, 1850
<code>\l_shipout_box</code>	2532, 2534, 2542	<code>\tex_pdflastximagepages:D</code>	1916
skip commands:		<code>\tex_pdflinkmargin:D</code>	2726
<code>\skip_horizontal:n</code>	226, 274, 331	<code>\tex_pdfliterate:D</code>	94, 105, 115
str commands:		<code>\tex_pdfmajorversion:D</code>	2913, 2915, 2939, 2940
<code>\c_hash_str</code>	397, 1616, 1623, 1663	<code>\tex_pdfminorversion:D</code>	2927, 2951
<code>\c_percent_str</code>	1071, 1072, 1073	<code>\tex_pdfobj:D</code>	2794, 2814, 2848
<code>\str_case:nn</code>	866, 2289, 2823	<code>\tex_pdfobjcompresslevel:D</code>	2899
<code>\str_case:nnTF</code>	2569, 2739, 3067	<code>\tex_pdfpageref:D</code>	2871
<code>\str_convert_pdfname:n</code>	587, 607, 816	<code>\tex_pdfrefximage:D</code>	1850, 1857
<code>\str_if_empty:NTF</code>	1791, 1807	<code>\tex_pdfrestore:D</code>	134
<code>\str_if_empty_p:N</code>	1832	<code>\tex_pdfsave:D</code>	125
<code>\str_if_eq:nnTF</code>	536, 766, 3317	<code>\tex_pdfsetmatrix:D</code>	143
<code>\str_new:N</code>	1865, 1866, 1867	<code>\tex_pdfstartlink:D</code>	2695
<code>\str_tail:N</code>	1879, 1905	<code>\tex_pdfvariable:D</code>	2723, 2879, 2896, 2908, 2924, 2935, 2948
sys commands:		<code>\tex_pdfximage:D</code>	1827, 1914
<code>\sys_if_shell:TF</code>	1863	<code>\tex_spacefactor:D</code>	2514, 2523
<code>\sys_shell_now:n</code>	1890	<code>\tex_special:D</code>	46
T		<code>\tex_the:D</code>	1822, 2935, 2940, 2946
TeX and L ^A T _E X 2 _ε commands:		<code>\tex_vss:D</code>	2599, 2606, 3097, 3116
<code>\@ifl@t@r</code>	49, 51	<code>\tex_XeTeXpdffile:D</code>	2046, 2092
<code>\@makecol@hook</code>	2528, 2530	<code>\tex_XeTeXpdfpagecount:D</code>	2102
<code>\special</code>	2	<code>\tex_XeTeXpicfile:D</code>	2037
tex commands:		TeXcolorseparation	<u>3400</u>
<code>\tex_afterassignment:D</code>	2188	<code>\textwidth</code>	2498
<code>\tex_baselineskip:D</code>	2503	tl commands:	
<code>\tex_endinput:D</code>	44	<code>\c_space_tl</code>	288, 293, 296, 549, 554, 592, 695, 769, 979, 1592, 1762, 1763, 1764, 1765, 1952, 1953, 1954, 1955, 2003, 2006, 2008, 2009, 2010, 2011, 2072, 2094, 2221, 2222, 2223, 2224, 2454, 2683, 2718, 2862, 2873, 3025, 3047
<code>\tex_global:D</code>	2877, 2894, 2908, 2915, 2922	<code>\tl_clear:N</code>	1784, 1800, 1934, 1942, 2036, 2044, 2201, 2208
<code>\tex_immediate:D</code>	1827, 2811, 2814, 2845, 2848	<code>\tl_gclear:N</code>	1630, 1666
<code>\tex luatexversion:D</code>	2906, 2934	<code>\tl_gset:Nn</code>	1589, 2409
<code>\tex_pageheight:D</code>	3197	<code>\tl_if_blank:nTF</code>	462, 547, 643, 660, 667, 685, 811, 893, 2071, 2157
<code>\tex_pagewidth:D</code>	3196	<code>\tl_if_empty:NTF</code>	1592, 1787, 1837, 1846, 1973, 1977, 2004, 2019, 2059
<code>\tex_pdfannot:D</code>	2667	<code>\tl_if_emptyv:nTF</code>	905, 1688
<code>\tex_pdfcatalog:D</code>	2773		
<code>\tex_pdfcolorstack:D</code>	472, 482		
<code>\tex_pdfcolorstackinit:D</code>	460		
<code>\tex_pdfcompresslevel:D</code>	2882		
<code>\tex_pdfdest:D</code>	2736, 2759		
<code>\tex_pdfendlink:D</code>	2706		

<code>\tl_if_empty_p:N</code>	1831, 2016				U
<code>\tl_new:N</code>	507,			use commands:	
	508, 1596, 1780, 2384, 2388, 3264, 3265			<code>\use:N</code>	43, 2298, 2980, 3009
<code>\tl_put_right:Nn</code>	2530			<code>\use:n</code>	58, 795, 821, 876,
<code>\tl_set:Nn</code>	509, 510, 519,				1032, 1042, 1055, 1298, 1422, 1487,
	520, 965, 977, 1785, 1802, 1893,				1499, 1511, 1671, 2066, 2150, 2172
	2389, 2548, 3273, 3274, 3320, 3321			<code>\use_none:n</code>	1688, 2526
<code>\tl_to_str:n</code>	2153, 2175, 2271,				V
	2275, 2798, 2807, 2818, 2973, 2977			<code>\value</code>	2434
<code>\tl_use:N</code>	727, 840			vbox commands:	
token commands:				<code>\vbox_set:Nn</code>	2534
<code>\c_math_toggle_token</code>	2417, 2427			<code>\vbox_to_zero:n</code>	2595, 2602, 3089, 3100
				<code>\vbox_unpack_drop:N</code>	2542