

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

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

```
\_kernel_backend_scope_end:
```

```
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.

__kernel_backend_literal_pdf:n 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:` `xdvipdfmx (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 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 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 of the open scopes. That is easiest done if we only need an outer “wrapper” `begin/end` pair, and within that we apply operations as a simple scoped statements. To keep down the non-productive groups, we also have a `begin` version that does take an argument.

```

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 `TEX` magnification. Thus for any operation involving absolute lengths there is a correction to make. See `normalscale` from `special.pro` for the variables, noting that here everything is saved on the stack rather than as a separate variable. Once all of that is done, the actual clipping is trivial.

```

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.

```

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     \c_space_tl

```

```

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 LuaTeX/pdfTeX: 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` `__box_backend_rotate_aux: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 </dviptfm | xetex>
```

2.4 dvisvgm backend

```
361 <*dvisvgm>
```

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

```

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

```

```

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

```

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

```

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

```

(End of definition for $__box_backend_clip:N$ and $__kernel_clip_path_int$.)

`_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 </divisvgn>
445 </package>

```

3 I3backend-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/XYTeX`.

```
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 </luatex>
481 <*pdftex>
482   \tex_pdfcolorstack:D
483 </pdftex>
484   \int_eval:n {#1} ~ pop \scan_stop:
485 }

```

(End of definition for __kernel_color_backend_stack_push:nm 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 <*dvips>
500     \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
501 </dvips>
502   }
503 \cs_new_protected:Npn \__color_backend_reset:
504   { \__kernel_backend_literal:n { color~pop } }

```

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

```
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 Store the values then pass to the stack.
\__color_backend_select_gray:n 511 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
\__color_backend_select_rgb:n 512 { \__color_backend_select:nn { #1 ~ k } { #1 ~ K } }
\__color_backend_select:nn 513 \cs_new_protected:Npn \__color_backend_select_gray:n #1
\__color_backend_reset: 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 dvipdfx/X_qTeX

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 Using the single stack is relatively easy as there is only one route.
\__color_backend_select_cmyk:n 527 \cs_new_protected:Npn \__color_backend_select:n #1
\__color_backend_select_gray:n 528 { \__kernel_backend_literal:n { pdf : bc ~ [ #1 ] } }
\__color_backend_select_rgb:n 529 \cs_new_eq:NN \__color_backend_select_cmyk:n \__color_backend_select:n
\__color_backend_reset: 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 <*/dviptfm | 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 </dviptfm | 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:nmnn
\_color_backend_separation_init:neenn
\_color_backend_separation_init_aux:nmnnn
\_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:nmnn
  \_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:nmnnn #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:nmnnn
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:nmnnn { nee }
598 \cs_new_protected:Npn \_color_backend_separation_init_aux:nmnnn #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:nnnnn` and others.)

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

```

734 \cs_new_protected:Npn \__color_backend_devicen_init:nm #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:nm`.)

`__color_backend_iccbased_init:nm` No support at present.

```

753 \cs_new_protected:Npn \__color_backend_iccbased_init:nm #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:nnnnn #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 `<*dviptfm | luatex | pdftex | xetex>`

`_color_backend_select_separation:nn`

`_color_backend_select_devicen:nn` 778 `<*dviptfm | xetex>`

`_color_backend_select_iccbased:nn` 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 `</dviptfm | 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:n` Initialising the PDF structures needs two parts: creating an object containing the “real” name of the Separation, then adding a reference to that to each page. We use a separate object for the tint transformation following the model in the PDF reference. 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:n #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     /CO ~ [#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 _t1 } ]
841         /Range ~ [ \c__color_model_range_CIELAB_t1 ]
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_t1 }
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 </dviptfm | luatex | pdftex | xetex>

```

3.4 Fill and stroke color

Here, `dvipdfmx/XqTeX` 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 <*dviptfm | 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 /dvi pdfmx | xetex \rangle

956 \langle * luatex | pdftex \rangle

`_color_backend_fill_cmyk:n` Drawing (fill/stroke) color is handled in $\text{dvi}\text{pdfmx}/\text{X}\text{g}\text{T}\text{E}\text{X}$ in the same way as $\text{L}\text{u}\text{a}\text{T}\text{E}\text{X}/\text{p}\text{d}\text{f}\text{T}\text{E}\text{X}$.
`_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 Fill color here is the same as general color except we skip the stroke part.
\__color_backend_fill_gray:n 991 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
\__color_backend_fill_rgb:n 992 { \__color_backend_fill:n { cmyk ~ #1 } }
\__color_backend_fill:n 993 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
\__color_backend_stroke_cmyk:n 994 { \__color_backend_fill:n { gray ~ #1 } }
\__color_backend_stroke_gray:n 995 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
\__color_backend_stroke_rgb:n 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 1007 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
\__color_backend_fill_devicen:nn 1008 { \__color_backend_fill:n { separation ~ #1 ~ #2 } }
\__color_backend_stroke_devicen:nn 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>

\__color_backend_fill_cmyk:n Fill color here is the same as general color except we skip the stroke part.
\__color_backend_fill_gray:n 1017 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
\__color_backend_fill_rgb:n 1018 { \__color_backend_fill:n { cmyk ~ #1 } }
\__color_backend_fill:n 1019 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
\__color_backend_stroke_cmyk:n 1020 { \__color_backend_fill:n { gray ~ #1 } }
\__color_backend_stroke_gray:n 1021 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
\__color_backend_stroke_rgb:n 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.)

```

`_color_backend_stroke_cmyk:n` For drawings in SVG, we use scopes for all stroke colors. That requires using RGB values,

`_color_backend_stroke_cmyk:w` which luckily are easy to convert here (cmyk to RGB is a fixed function).

`_color_backend_stroke_gray:n`

`_color_backend_stroke_gray_aux:n`

`_color_backend_stroke_rgb:n`

`_color_backend_stroke_rgb:w`

`_color_backend:nnn`

```

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.

```
\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
1078 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1079 \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`

No support at present.

```
\_color_backend_iccbased_init:nnn
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 </dvisvgm>
```

```
1087 </package>
```

3.5 Font handling integration

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

```
1088 <*lua>
1089 local l = lpeg
1090 local spaces = l.P' '^0
1091 local digit16 = l.R('09', 'af', 'AF')
1092
1093 local octet = digit16 * digit16 / function(s)
1094   return string.format('%.3g ', tonumber(s, 16) / 255)
1095 end
1096
1097 if luaotfload and luaotfload.set_transparent_colorstack then
1098   local htmlcolor = l.Cs(octet * octet * octet * -1 * l.Cc'rg')
1099   local color_export = {
1100     token.create'tex_endlocalcontrol:D',
1101     token.create'tex_hpack:D',
1102     token.new(0, 1),
1103     token.create'color_export:nnN',
1104     token.new(0, 1),
1105     '',
1106     token.new(0, 2),
1107     token.new(0, 1),
1108     'backend',
1109     token.new(0, 2),
1110     token.create'l_tmpa_tl',
1111     token.create'exp_after:wN',
1112     token.create'__color_select:nn',
```

```

1113     token.create'l_tmpa_tl',
1114     token.new(0, 2),
1115 }
1116 local group_end = token.create'group_end:'
1117 local value = (1 - l.P}')^0
1118 luatexbase.add_to_callback('luaotfload.parse_color', function (value)
1119 % Also allow HTML colors to preserve compatibility
1120     local html = htmlcolor:match(value)
1121     if html then return html end
1122
1123 % 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 here.
`__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. As for pgf, we need to save the current point as this is required for box placement. (Note that @beginspecial/@endspecial forms a backend scope.)

```

1159 \cs_new_protected:Npn \_draw_backend_begin:
1160 {
1161   \_draw_backend_literal:n { [begin] }
1162   \_draw_backend_literal:n { /draw.x~currentpoint~/draw.y~exch~def~def }
1163   \_draw_backend_literal:n { @beginspecial }
1164 }
1165 \cs_new_protected:Npn \_draw_backend_end:
1166 {
1167   \_draw_backend_literal:n { @endspecial }
1168   \_draw_backend_literal:n { [end] }
1169 }

```

(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.

_draw_backend_scope_end:

```

1170 \cs_new_protected:Npn \_draw_backend_scope_begin:
1171 { \_draw_backend_literal:n { save } }
1172 \cs_new_protected:Npn \_draw_backend_scope_end:
1173 { \_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.

_draw_backend_lineto:nn
_draw_backend_rectangle:nmmn
_draw_backend_curveto:nmmmmn

```

1174 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1175 {
1176   \_draw_backend_literal:e
1177   {
1178     \dim_to_decimal_in_bp:n {#1} ~
1179     \dim_to_decimal_in_bp:n {#2} ~ moveto
1180   }
1181 }
1182 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1183 {
1184   \_draw_backend_literal:e
1185   {
1186     \dim_to_decimal_in_bp:n {#1} ~
1187     \dim_to_decimal_in_bp:n {#2} ~ lineto
1188   }

```



```

1189 }
1190 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1191 {
1192   \__draw_backend_literal:e
1193   {
1194     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1195     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1196     moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1197   }
1198 }
1199 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1200 {
1201   \__draw_backend_literal:e
1202   {
1203     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1204     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1205     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1206     curveto
1207   }
1208 }

```

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

```

\__draw_backend_evenodd_rule: The even-odd rule here can be implemented as a simply switch.
\__draw_backend_nonzero_rule:
\g__draw_draw_eor_bool
1209 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1210 { \bool_gset_true:N \g__draw_draw_eor_bool }
1211 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1212 { \bool_gset_false:N \g__draw_draw_eor_bool }
1213 \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. `__draw_backend_fillstroke:` All of the operations end with a new path instruction as they do not terminate (again in contrast to PDF).

```

\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
1214 \cs_new_protected:Npn \__draw_backend_closepath:
1215 { \__draw_backend_literal:n { closepath } }
1216 \cs_new_protected:Npn \__draw_backend_stroke:
1217 {
1218   \__draw_backend_literal:n { gsave }
1219   \__draw_backend_literal:n { color.sc }
1220   \__draw_backend_literal:n { stroke }
1221   \__draw_backend_literal:n { grestore }
1222   \bool_if:NT \g__draw_draw_clip_bool
1223   {
1224     \__draw_backend_literal:e
1225     {
1226       \bool_if:NT \g__draw_draw_eor_bool { eo }
1227       clip
1228     }

```

```

1229     }
1230     \__draw_backend_literal:n { newpath }
1231     \bool_gset_false:N \g__draw_draw_clip_bool
1232   }
1233   \cs_new_protected:Npn \__draw_backend_closestroke:
1234   {
1235     \__draw_backend_closepath:
1236     \__draw_backend_stroke:
1237   }
1238   \cs_new_protected:Npn \__draw_backend_fill:
1239   {
1240     \__draw_backend_literal:e
1241     {
1242       \bool_if:NT \g__draw_draw_eor_bool { eo }
1243       fill
1244     }
1245     \bool_if:NT \g__draw_draw_clip_bool
1246     {
1247       \__draw_backend_literal:e
1248       {
1249         \bool_if:NT \g__draw_draw_eor_bool { eo }
1250         clip
1251       }
1252     }
1253     \__draw_backend_literal:n { newpath }
1254     \bool_gset_false:N \g__draw_draw_clip_bool
1255   }
1256   \cs_new_protected:Npn \__draw_backend_fillstroke:
1257   {
1258     \__draw_backend_literal:e
1259     {
1260       \bool_if:NT \g__draw_draw_eor_bool { eo }
1261       fill
1262     }
1263     \__draw_backend_literal:n { gsave }
1264     \__draw_backend_literal:n { color.sc }
1265     \__draw_backend_literal:n { stroke }
1266     \__draw_backend_literal:n { grestore }
1267     \bool_if:NT \g__draw_draw_clip_bool
1268     {
1269       \__draw_backend_literal:e
1270       {
1271         \bool_if:NT \g__draw_draw_eor_bool { eo }
1272         clip
1273       }
1274     }
1275     \__draw_backend_literal:n { newpath }
1276     \bool_gset_false:N \g__draw_draw_clip_bool
1277   }
1278   \cs_new_protected:Npn \__draw_backend_clip:
1279   { \bool_gset_true:N \g__draw_draw_clip_bool }
1280   \bool_new:N \g__draw_draw_clip_bool
1281   \cs_new_protected:Npn \__draw_backend_discardpath:
1282   {

```

```

1283   \bool_if:NT \g__draw_draw_clip_bool
1284     {
1285       \__draw_backend_literal:e
1286         {
1287           \bool_if:NT \g__draw_draw_eor_bool { eo }
1288             clip
1289         }
1290     }
1291   \__draw_backend_literal:n { newpath }
1292   \bool_gset_false:N \g__draw_draw_clip_bool
1293 }

```

(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:
1294 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1295 {
1296   \__draw_backend_literal:e
1297     {
1298     [
1299       \exp_args:Nf \use:n
1300         { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1301     ] ~
1302     \dim_to_decimal_in_bp:n {#2} ~ setdash
1303   }
1304 }
1305 \cs_new:Npn \__draw_backend_dash:n #1
1306 { ~ \dim_to_decimal_in_bp:n {#1} }
1307 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1308 {
1309   \__draw_backend_literal:e
1310     { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1311 }
1312 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1313 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1314 \cs_new_protected:Npn \__draw_backend_cap_but:
1315 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1316 \cs_new_protected:Npn \__draw_backend_cap_round:
1317 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1318 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1319 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1320 \cs_new_protected:Npn \__draw_backend_join_miter:
1321 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1322 \cs_new_protected:Npn \__draw_backend_join_round:
1323 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1324 \cs_new_protected:Npn \__draw_backend_join_bevel:
1325 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

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

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

```

1326 \cs_new_protected:Npn \__draw_backend_cm:nmmm #1#2#3#4
1327 {
1328   \__draw_backend_literal:n
1329   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1330 }

```

(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. A previous implementation suggested by Tom Rokici used `@endspecial/@beginspecial`. This avoids needing internals of `dvips`, but fails if there the box is used inside a scope (see <https://github.com/latex3/latex3/issues/1504>). Instead, we use the same method as `pgf`, which means tracking the position at the PostScript level. Also note that using `@endspecial` would close the scope it creates, meaning that after a box insertion, any local changes would be lost. Keeping `dvips` on track is non-trivial, hence the `[begin]/[end]` pair before the `save` and around the `restore`.

```

1331 \cs_new_protected:Npn \__draw_backend_box_use:Nmmmm #1#2#3#4#5
1332 {
1333   \__draw_backend_literal:n { save }
1334   \__draw_backend_literal:n { 72~Resolution~div~72~VResolution~div~neg~scale }
1335   \__draw_backend_literal:n { magscale { 1~DVImag~div~dup~scale } if }
1336   \__draw_backend_literal:n { draw.x~neg~draw.y~neg~translate }
1337   \__draw_backend_literal:n { [end] }
1338   \__draw_backend_literal:n { [begin] }
1339   \__draw_backend_literal:n { save }
1340   \__draw_backend_literal:n { currentpoint }
1341   \__draw_backend_literal:n { currentpoint~translate }
1342   \__draw_backend_cm:nmmm { 1 } { 0 } { 0 } { -1 }
1343   \__draw_backend_cm:nmmm {#2} {#3} {#4} {#5}
1344   \__draw_backend_cm:nmmm { 1 } { 0 } { 0 } { -1 }
1345   \__draw_backend_literal:n { neg~exch~neg~exch~translate }
1346   \__draw_backend_literal:n { [end] }
1347   \hbox_overlap_right:n { \box_use:N #1 }
1348   \__draw_backend_literal:n { [begin] }
1349   \__draw_backend_literal:n { restore }
1350   \__draw_backend_literal:n { [end] }
1351   \__draw_backend_literal:n { [begin] }
1352   \__draw_backend_literal:n { restore }
1353 }

```

(End of definition for __draw_backend_box_use:Nmmmm.)

```

1354 </dvips>

```

4.2 LuaTeX, pdfTeX, dviPDFx and XeTeX

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

```

1355 < *dviPDFx | luatex | pdftex | xetex >

```

4.2.1 Drawing

```

\__draw_backend_literal:n Pass data through using a dedicated interface.
\__draw_backend_literal:e 1356 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1357 \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: 1358 \cs_new_protected:Npn \__draw_backend_begin:
1359 { \__draw_backend_scope_begin: }
1360 \cs_new_protected:Npn \__draw_backend_end:
1361 { \__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: 1362 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1363 \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 1364 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
\__draw_backend_rectangle:nnnn 1365 {
1366 \__draw_backend_literal:e
1367 { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1368 }
1369 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1370 {
1371 \__draw_backend_literal:e
1372 { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1373 }
1374 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1375 {
1376 \__draw_backend_literal:e
1377 {
1378 \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1379 \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1380 \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1381 c
1382 }
1383 }
1384 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1385 {
1386 \__draw_backend_literal:e
1387 {
1388 \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1389 \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1390 re
1391 }
1392 }

(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: 1393 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
\g__draw_draw_eor_bool 1394 { \bool_gset_true:N \g__draw_draw_eor_bool }
1395 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1396 { \bool_gset_false:N \g__draw_draw_eor_bool }
1397 \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: 1398 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_closestroke: 1399 { \__draw_backend_literal:n { h } }
\__draw_backend_fill: 1400 \cs_new_protected:Npn \__draw_backend_stroke:
\__draw_backend_fillstroke: 1401 { \__draw_backend_literal:n { S } }
\__draw_backend_clip: 1402 \cs_new_protected:Npn \__draw_backend_closestroke:
\__draw_backend_discardpath: 1403 { \__draw_backend_literal:n { s } }
1404 \cs_new_protected:Npn \__draw_backend_fill:
1405 {
1406 \__draw_backend_literal:e
1407 { f \bool_if:NT \g__draw_draw_eor_bool * }
1408 }
1409 \cs_new_protected:Npn \__draw_backend_fillstroke:
1410 {
1411 \__draw_backend_literal:e
1412 { B \bool_if:NT \g__draw_draw_eor_bool * }
1413 }
1414 \cs_new_protected:Npn \__draw_backend_clip:
1415 {
1416 \__draw_backend_literal:e
1417 { W \bool_if:NT \g__draw_draw_eor_bool * }
1418 }
1419 \cs_new_protected:Npn \__draw_backend_discardpath:
1420 { \__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 1421 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
\__draw_backend_linewidth:n 1422 {
\__draw_backend_miterlimit:n 1423 \__draw_backend_literal:e
\__draw_backend_cap_butt: 1424 {
\__draw_backend_cap_round: 1425 [
\__draw_backend_cap_rectangle: 1426 \exp_args:Nf \use:n
\__draw_backend_join_miter: 1427 { \clist_map_function:nN {#1} \__draw_backend_dash:n }
\__draw_backend_join_round: 1428 ] ~
\__draw_backend_join_bevel: 1429 \dim_to_decimal_in_bp:n {#2} ~ d
1430 }
1431 }
1432 \cs_new:Npn \__draw_backend_dash:n #1
1433 { ~ \dim_to_decimal_in_bp:n {#1} }
1434 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1435 {
1436 \__draw_backend_literal:e

```

```

1437     { \dim_to_decimal_in_bp:n {#1} ~ w }
1438   }
1439 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1440   { \__draw_backend_literal:e { #1 ~ M } }
1441 \cs_new_protected:Npn \__draw_backend_cap_but:
1442   { \__draw_backend_literal:n { 0 ~ J } }
1443 \cs_new_protected:Npn \__draw_backend_cap_round:
1444   { \__draw_backend_literal:n { 1 ~ J } }
1445 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1446   { \__draw_backend_literal:n { 2 ~ J } }
1447 \cs_new_protected:Npn \__draw_backend_join_miter:
1448   { \__draw_backend_literal:n { 0 ~ j } }
1449 \cs_new_protected:Npn \__draw_backend_join_round:
1450   { \__draw_backend_literal:n { 1 ~ j } }
1451 \cs_new_protected:Npn \__draw_backend_join_bevel:
1452   { \__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 to decompose the matrix into rotations and a scaling, then use those operations as they are handled by the backend. (There is backend support for matrix operations in dvipdfmx/XqTeX, but as a matched pair so not suitable for the “stand alone” transformation set up here.) The specials used here are from xdvipdfmx originally: they are well-tested, but probably equivalent to the pdf: versions!

```

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

```

```

1481     x:rotate~
1482     \fp_compare:nNnTF {#4} = \c_zero_fp
1483       { 0 }
1484       { \fp_eval:n { round ( -#4 , 5 ) } }
1485   }
1486 }
1487 </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.

```

1488 <*dvipdfmx | xetex>
1489 \cs_new_protected:Npn \_draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1490 {
1491   \use:e
1492   {
1493     \_draw_backend_cm_decompose_auxi:nnnnN
1494     { \fp_eval:n { (#1 + #4) / 2 } }
1495     { \fp_eval:n { (#1 - #4) / 2 } }
1496     { \fp_eval:n { (#3 + #2) / 2 } }
1497     { \fp_eval:n { (#3 - #2) / 2 } }
1498   }
1499   #5
1500 }
1501 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1502 {
1503   \use:e

```



```

1504     {
1505       \__draw_backend_cm_decompose_auxii:nnnnN
1506       { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1507       { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1508       { \fp_eval:n { atand ( #3 , #2 ) } }
1509       { \fp_eval:n { atand ( #4 , #1 ) } }
1510     }
1511     #5
1512   }
1513 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1514 {
1515   \use:e
1516   {
1517     \__draw_backend_cm_decompose_auxiii:nnnnN
1518     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1519     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1520     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1521     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1522   }
1523   #5
1524 }
1525 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1526 {
1527   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1528     { #5 {#1} {#2} {#3} {#4} }
1529     { #5 {#1} {#3} {#2} {#4} }
1530 }
1531 \</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.

```

1532 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1533 {
1534   \__kernel_backend_scope_begin:
1535   <*luatex | pdftex>
1536   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1537   </luatex | pdftex>
1538   <*dviPDFmx | xetex>
1539   \__kernel_backend_literal:n
1540   { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1541   </dviPDFmx | xetex>
1542   \hbox_overlap_right:n { \box_use:N #1 }
1543   <*dviPDFmx | xetex>
1544   \__kernel_backend_literal:n { pdf:etrans }
1545   </dviPDFmx | xetex>
1546   \__kernel_backend_scope_end:
1547 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1548 </dviPDFmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

1549 `<*dvisvgm>`

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

`_draw_backend_literal:e` 1550 `\cs_new_eq:NN _draw_backend_literal:n _kernel_backend_literal_svg:n`
 1551 `\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:` 1552 `\cs_new_eq:NN _draw_backend_scope_begin: _kernel_backend_scope_begin:`
 1553 `\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

1554 `\cs_new_protected:Npn _draw_backend_begin:`
 1555 `{`
 1556 `_kernel_backend_scope_begin:`
 1557 `_kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }`
 1558 `}`
 1559 `\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:nmmn` in one go. For that we use a dedicated storage routine, which adds spaces as required.
`_draw_backend_curveto:nnnnn` 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` 1560 `\cs_new_protected:Npn _draw_backend_moveto:nn #1#2`
 1561 `{`
 1562 `_draw_backend_add_to_path:n`
 1563 `{ M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }`
 1564 `}`
 1565 `\cs_new_protected:Npn _draw_backend_lineto:nn #1#2`
 1566 `{`
 1567 `_draw_backend_add_to_path:n`
 1568 `{ L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }`
 1569 `}`
 1570 `\cs_new_protected:Npn _draw_backend_rectangle:nmmn #1#2#3#4`
 1571 `{`
 1572 `_draw_backend_add_to_path:n`
 1573 `{`
 1574 `M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}`
 1575 `h ~ \dim_to_decimal:n {#3} ~`
 1576 `v ~ \dim_to_decimal:n {#4} ~`
 1577 `h ~ \dim_to_decimal:n { -#3 } ~`
 1578 `Z`
 1579 `}`
 1580 `}`
 1581 `\cs_new_protected:Npn _draw_backend_curveto:nnnnn #1#2#3#4#5#6`
 1582 `{`

```

1583   \__draw_backend_add_to_path:n
1584   {
1585     C ~
1586     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1587     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1588     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1589   }
1590 }
1591 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1592 {
1593   \tl_gset:Ne \g__draw_backend_path_tl
1594   {
1595     \g__draw_backend_path_tl
1596     \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1597     #1
1598   }
1599 }
1600 \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:
1601 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1602   { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1603 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1604   { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

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

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

```

\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int
1605 \cs_new_protected:Npn \__draw_backend_closepath:
1606   { \__draw_backend_add_to_path:n { Z } }
1607 \cs_new_protected:Npn \__draw_backend_path:n #1
1608   {
1609     \bool_if:NTF \g__draw_draw_clip_bool
1610     {
1611       \int_gincr:N \g__kernel_clip_path_int
1612       \__draw_backend_literal:e
1613       {
1614         < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1615         { ?nl }
1616         <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1617         </clipPath > { ? nl }
1618         <
1619         use~xlink:href =
1620         "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1621         #1
1622         />
1623       }
1624       \__kernel_backend_scope:e

```

```

1625     {
1626         clip-path =
1627         "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1628     }
1629 }
1630 {
1631     \__draw_backend_literal:e
1632     { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1633 }
1634 \tl_gclear:N \g__draw_backend_path_tl
1635 \bool_gset_false:N \g__draw_draw_clip_bool
1636 }
1637 \int_new:N \g__draw_backend_path_int
1638 \cs_new_protected:Npn \__draw_backend_stroke:
1639 { \__draw_backend_path:n { style="fill:none" } }
1640 \cs_new_protected:Npn \__draw_backend_closestroke:
1641 {
1642     \__draw_backend_closepath:
1643     \__draw_backend_stroke:
1644 }
1645 \cs_new_protected:Npn \__draw_backend_fill:
1646 { \__draw_backend_path:n { style="stroke:none" } }
1647 \cs_new_protected:Npn \__draw_backend_fillstroke:
1648 { \__draw_backend_path:n { } }
1649 \cs_new_protected:Npn \__draw_backend_clip:
1650 { \bool_gset_true:N \g__draw_draw_clip_bool }
1651 \bool_new:N \g__draw_draw_clip_bool
1652 \cs_new_protected:Npn \__draw_backend_discardpath:
1653 {
1654     \bool_if:NT \g__draw_draw_clip_bool
1655     {
1656         \int_gincr:N \g__kernel_clip_path_int
1657         \__draw_backend_literal:e
1658         {
1659             < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1660             { ?nl }
1661             <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1662             < /clipPath >
1663         }
1664         \__kernel_backend_scope:e
1665         {
1666             clip-path =
1667             "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1668         }
1669     }
1670     \tl_gclear:N \g__draw_backend_path_tl
1671     \bool_gset_false:N \g__draw_draw_clip_bool
1672 }

```

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

```

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

```

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

```

1673 \cs_new_protected:Npn \__draw_backend_dash_pattern:mn #1#2

```

```

1674 {
1675   \use:e
1676   {
1677     \__draw_backend_dash_aux:nn
1678     { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1679     { \dim_to_decimal:n {#2} }
1680   }
1681 }
1682 \cs_new:Npn \__draw_backend_dash:n #1
1683 { , \dim_to_decimal_in_bp:n {#1} }
1684 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1685 {
1686   \__kernel_backend_scope:e
1687   {
1688     stroke-dasharray =
1689     "
1690       \tl_if_empty:nTF {#1}
1691       { none }
1692       { \use_none:n #1 }
1693     " ~
1694     stroke-offset=" #2 "
1695   }
1696 }
1697 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1698 { \__kernel_backend_scope:e { stroke-width=" \dim_to_decimal:n {#1} " } }
1699 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1700 { \__kernel_backend_scope:e { stroke-miterlimit=" #1 " } }
1701 \cs_new_protected:Npn \__draw_backend_cap_but:
1702 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1703 \cs_new_protected:Npn \__draw_backend_cap_round:
1704 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1705 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1706 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1707 \cs_new_protected:Npn \__draw_backend_join_miter:
1708 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1709 \cs_new_protected:Npn \__draw_backend_join_round:
1710 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1711 \cs_new_protected:Npn \__draw_backend_join_bevel:
1712 { \__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.

```

1713 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1714 {
1715   \__kernel_backend_scope:n
1716   {
1717     transform =
1718     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1719   }
1720 }

```

(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.

```

1721 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
1722 {
1723   \_kernel_backend_scope_begin:
1724   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1725   \_kernel_backend_literal_svg:n
1726   {
1727     < g~
1728       stroke="none"~
1729       transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1730     >
1731   }
1732   \box_set_wd:Nn #1 { Opt }
1733   \box_set_ht:Nn #1 { Opt }
1734   \box_set_dp:Nn #1 { Opt }
1735   \box_use:N #1
1736   \_kernel_backend_literal_svg:n { </g> }
1737   \_kernel_backend_scope_end:
1738 }

```

(End of definition for _draw_backend_box_use:Nnnnn.)

```
1739 </dvisvgm>
```

```
1740 </package>
```

5 l3backend-graphics implementation

```

1741 <*package>
1742 <@@=graphics>

```

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

```

1743 \cs_new_protected:Npn \_graphics_backend_loaded:n #1
1744 {
1745   \cs_if_exist:NTF \hook_gput_code:nnn
1746   {
1747     \hook_gput_code:nnn
1748     { package / l3graphics / after }
1749     { backend }
1750     {#1}
1751   }
1752   {#1}
1753 }

```

(End of definition for _graphics_backend_loaded:n.)

5.1 dvips backend

```
1754 <*dvips>
```

`\l_graphics_search_ext_seq`

```

1755 \_graphics_backend_loaded:n
1756 { \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` Simply use the generic function.

```
\_graphics_backend_getbb_ps:n 1757 \__graphics_backend_loaded:n
                               1758 {
                               1759   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
                               1760   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
                               1761 }
```

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

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

```
\_graphics_backend_include_ps:n 1762 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
                               1763 {
                               1764   \__kernel_backend_literal:e
                               1765   {
                               1766     PSfile = #1 \c_space_tl
                               1767     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
                               1768     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
                               1769     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
                               1770     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
                               1771   }
                               1772 }
                               1773 \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`

```
1774 \__graphics_backend_loaded:n
1775 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }
```

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

```
1776 </dvips>
```

5.2 LuaTeX and pdfTeX backends

```
1777 < *luatex | pdftex >
```

`\l_graphics_search_ext_seq`

```
1778 \__graphics_backend_loaded:n
1779 {
1780   \seq_set_from_clist:Nn
1781   \l_graphics_search_ext_seq
1782   { .pdf , .eps , .ps , .png , .jpg , .jpeg }
1783 }
```

(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.

```
1784 \tl_new:N \l__graphics_attr_tl
```

(End of definition for \l__graphics_attr_tl.)

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

```

1785 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1786 {
1787   \int_zero:N \l__graphics_page_int
1788   \tl_clear:N \l__graphics_pagebox_tl
1789   \tl_set:Ne \l__graphics_attr_tl
1790   {
1791     \tl_if_empty:NF \l__graphics_decodearray_str
1792     { :D \l__graphics_decodearray_str }
1793     \bool_if:NT \l__graphics_interpolate_bool
1794     { :I }
1795     \str_if_empty:NF \l__graphics_pdf_str
1796     { :X \l__graphics_pdf_str }
1797   }
1798   \_graphics_backend_getbb_auxi:n {#1}
1799 }
1800 \cs_new_eq:NN \_graphics_backend_getbb_jpeg:n \_graphics_backend_getbb_jpg:n
1801 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1802 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1803 {
1804   \tl_clear:N \l__graphics_decodearray_str
1805   \bool_set_false:N \l__graphics_interpolate_bool
1806   \tl_set:Ne \l__graphics_attr_tl
1807   {
1808     : \l__graphics_pagebox_tl
1809     \int_compare:nNnT \l__graphics_page_int > 1
1810     { :P \int_use:N \l__graphics_page_int }
1811     \str_if_empty:NF \l__graphics_pdf_str
1812     { :X \l__graphics_pdf_str }
1813   }
1814   \_graphics_backend_getbb_auxi:n {#1}
1815 }
1816 \cs_new_protected:Npn \_graphics_backend_getbb_auxi:n #1
1817 {
1818   \_graphics_bb_restore:eF { #1 \l__graphics_attr_tl }
1819   { \_graphics_backend_getbb_auxii:n {#1} }
1820 }

```

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.

```

1821 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:n #1
1822 {
1823   \exp_args:Ne \_graphics_backend_getbb_auxiii:n
1824   { \_graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1825   \int_const:cn { c__graphics_ #1 \l__graphics_attr_tl _int }
1826   { \tex_the:D \tex_pdflastximage:D }

```



```

1827   \__graphics_bb_save:e { #1 \l__graphics_attr_tl }
1828   }
1829   \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1830   {
1831     \tex_immediate:D \tex_pdfximage:D
1832     \bool_lazy_any:nT
1833     {
1834       { \l__graphics_interpolate_bool }
1835       { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1836       { ! \str_if_empty_p:N \l__graphics_pdf_str }
1837     }
1838     {
1839       attr ~
1840       {
1841         \tl_if_empty:NF \l__graphics_decodearray_str
1842         { /Decode~[ \l__graphics_decodearray_str ] }
1843         \bool_if:NT \l__graphics_interpolate_bool
1844         { /Interpolate~true }
1845         \l__graphics_pdf_str
1846       }
1847     }
1848     \int_compare:nNnT \l__graphics_page_int > 0
1849     { page ~ \int_use:N \l__graphics_page_int }
1850     \tl_if_empty:NF \l__graphics_pagebox_tl
1851     { \l__graphics_pagebox_tl }
1852     {#1}
1853     \hbox_set:Nn \l__graphics_internal_box
1854     { \tex_pdfrefximage:D \tex_pdflastximage:D }
1855     \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1856     \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1857   }
1858   \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.

```

1859   \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1860   {
1861     \tex_pdfrefximage:D
1862     \int_use:c { c__graphics_ #1 \l__graphics_attr_tl _int }
1863   }
1864   \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1865   \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1866   \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.

```

1867   \sys_if_shell:T
1868   {

```

```

\l__graphics_backend_dir_str
\l__graphics_backend_name_str
\l__graphics_backend_ext_str

```

```

1869 \str_new:N \l__graphics_backend_dir_str
1870 \str_new:N \l__graphics_backend_name_str
1871 \str_new:N \l__graphics_backend_ext_str
1872 \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1873 {
1874   \file_parse_full_name:nNNN {#1}
1875   \l__graphics_backend_dir_str
1876   \l__graphics_backend_name_str
1877   \l__graphics_backend_ext_str
1878   \exp_args:Ne \__graphics_backend_getbb_eps:nn
1879   {
1880     \exp_args:Ne \__kernel_file_name_quote:n
1881     {
1882       \l__graphics_backend_name_str
1883       - \str_tail:N \l__graphics_backend_ext_str
1884       -converted-to.pdf
1885     }
1886   }
1887   {#1}
1888 }
1889 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_backend_getbb_eps:n
1890 \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1891 {
1892   \file_compare_timestamp:nNnT {#2} > {#1}
1893   {
1894     \sys_shell_now:n
1895     { repstopdf ~ #2 ~ #1 }
1896   }
1897   \tl_set:Nn \l__graphics_final_name_str {#1}
1898   \__graphics_backend_getbb_pdf:n {#1}
1899 }
1900 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1901 {
1902   \file_parse_full_name:nNNN {#1}
1903   \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1904   \exp_args:Ne \__graphics_backend_include_pdf:n
1905   {
1906     \exp_args:Ne \__kernel_file_name_quote:n
1907     {
1908       \l__graphics_backend_name_str
1909       - \str_tail:N \l__graphics_backend_ext_str
1910       -converted-to.pdf
1911     }
1912   }
1913 }
1914 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1915 }

```

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

__graphics_backend_get_pagecount:n Simply load and store.

```

1916 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1917 {
1918   \tex_pdfximage:D {#1}

```

```

1919     \int_const:cn { c__graphics_ #1 _pages_int }
1920     { \int_use:N \tex_pdflastximagepages:D }
1921   }

```

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

```

1922 </luatex | pdftex>

```

5.3 dvipdfmx backend

```

1923 <*dvipdfmx | xetex>

```

`\l_graphics_search_ext_seq`

```

1924 \__graphics_backend_loaded:n
1925 {
1926   \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1927   { .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }
1928 }

```

(End of definition for `\l_graphics_search_ext_seq`.)

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

```

\__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
1929 \__graphics_backend_loaded:n
1930 {
1931   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1932   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
1933 }
1934 <*dvipdfmx>
1935 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1936 {
1937   \int_zero:N \l__graphics_page_int
1938   \tl_clear:N \l__graphics_pagebox_tl
1939   \__graphics_extract_bb:n {#1}
1940 }
1941 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1942 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1943 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
1944 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1945 {
1946   \tl_clear:N \l__graphics_decodearray_str
1947   \bool_set_false:N \l__graphics_interpolate_bool
1948   \__graphics_extract_bb:n {#1}
1949 }
1950 </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.

```

1951 \int_new:N \g__graphics_track_int

```

(End of definition for `\g__graphics_track_int`.)

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.

```

\__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:enn
\__graphics_backend_include_auxiii:enn
1952 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1953 {
1954   \__kernel_backend_literal:e
1955   {
1956     PSfile = #1 \c_space_tl
1957     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1958     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1959     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1960     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1961   }
1962 }
1963 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1964 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1965 { \__graphics_backend_include_auxi:nn {#1} { image } }
1966 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1967 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
1968 \cs_new_eq:NN \__graphics_backend_include_bmp:n \__graphics_backend_include_jpg:n
1969 {*dvipdfmx}
1970 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1971 { \__graphics_backend_include_auxi:nn {#1} { epdf } }
1972 </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.

```

1973 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1974 {
1975   \__graphics_backend_include_auxii:enn
1976   {
1977     \tl_if_empty:NF \l__graphics_pagebox_tl
1978     { : \l__graphics_pagebox_tl }
1979     \int_compare:nNnT \l__graphics_page_int > 1
1980     { :P \int_use:N \l__graphics_page_int }
1981     \tl_if_empty:NF \l__graphics_decodearray_str
1982     { :D \l__graphics_decodearray_str }
1983     \bool_if:NT \l__graphics_interpolate_bool
1984     { :I }
1985   }
1986   {#1} {#2}
1987 }
1988 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1989 {
1990   \int_if_exist:cTF { c__graphics_ #2#1 _int }
1991   {
1992     \__kernel_backend_literal:e
1993     { pdf:useobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
1994   }
1995   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1996 }
1997 \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!

```

1998 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1999 {
2000   \int_gincr:N \g__graphics_track_int
2001   \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
2002   \__kernel_backend_literal:e
2003   {
2004     pdf:#3~
2005     @graphic \int_use:c { c__graphics_ #1#2 _int } ~
2006     \int_compare:nNnT \l__graphics_page_int > 1
2007     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2008     \tl_if_empty:NF \l__graphics_pagebox_tl
2009     {
2010       pagebox ~ \l__graphics_pagebox_tl \c_space_tl
2011       bbox ~
2012         \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2013         \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2014         \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2015         \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
2016     }
2017     (#1)
2018     \bool_lazy_or:nnT
2019     { \l__graphics_interpolate_bool }
2020     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
2021     {
2022       <<
2023       \tl_if_empty:NF \l__graphics_decodearray_str
2024       { /Decode~[ \l__graphics_decodearray_str ] }
2025       \bool_if:NT \l__graphics_interpolate_bool
2026       { /Interpolate~true }
2027     }
2028     }
2029   }
2030 }

```

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

`__graphics_backend_get_pagecount:n`

```

2031 <*dvipdfmx>
2032 \__graphics_backend_loaded:n
2033 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }
2034 </dvipdfmx>

```

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

```

2035 </dvipdfmx | xetex>

```

5.4 X_YTeX backend

```

2036 <*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_auxiii:VnN
\__graphics_backend_getbb_auxiiii:nnNn
\__graphics_backend_getbb_auxiv:nnNn
\__graphics_backend_getbb_auxiv:VnNn
\__graphics_backend_getbb_auxv:nnNn

```

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

```

2037 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2038 {
2039   \int_zero:N \l__graphics_page_int
2040   \tl_clear:N \l__graphics_pagebox_tl
2041   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2042 }
2043 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2044 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
2045 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2046 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2047 {
2048   \tl_clear:N \l__graphics_decodearray_str
2049   \bool_set_false:N \l__graphics_interpolate_bool
2050   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2051 }
2052 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2053 {
2054   \int_compare:nNnTF \l__graphics_page_int > 1
2055     { \__graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2056     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2057 }
2058 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2059 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2060 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2061 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2062 {
2063   \tl_if_empty:NTF \l__graphics_pagebox_tl
2064     { \__graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2065     { \__graphics_backend_getbb_auxv:nNnn }
2066     {#1} #2 {#3} {#4}
2067 }
2068 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2069 {
2070   \use:e
2071   {
2072     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2073     {
2074       #5
2075       \tl_if_blank:nF {#1}
2076         { \c_space_tl \__graphics_backend_getbb_pagebox:w #1 }
2077     }
2078   }
2079 }
2080 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2081 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2082 {
2083   \__graphics_bb_restore:nF {#1#3}
2084   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2085 }
2086 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2087 {
2088   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }

```

```

2089     \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2090     \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2091     \__graphics_bb_save:n {#1#3}
2092   }
2093 \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_YT_EX 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`.

```

2094 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2095 {
2096   \tex_XeTeXpdffile:D #1 ~
2097   \int_compare:nNnT \l__graphics_page_int > 0
2098     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2099   \exp_after:wN \__graphics_backend_getbb_pagebox:w \l__graphics_pagebox_tl
2100 }

```

(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.

```

2101 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
2102 {
2103   \int_const:cn { c__graphics_#1_pages_int }
2104   {
2105     \int_max:nn
2106     { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2107     { 1 }
2108   }
2109 }

```

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

```
2110 </xetex>
```

5.5 dvisvgm backend

```
2111 <*dvisvgm>
```

`\l_graphics_search_ext_seq`

```

2112 \__graphics_backend_loaded:n
2113 {
2114   \seq_set_from_clist:Nn
2115   \l_graphics_search_ext_seq
2116   { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }
2117 }

```

(End of definition for `\l_graphics_search_ext_seq`.)

`\graphics_backend_getbb_svg:n` 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.

```

\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

```

```

2118 \cs_new_protected:Npn \__graphics_backend_getbb_svg:n #1
2119 {
2120   \__graphics_bb_restore:nF {#1}
2121   {
2122     \ior_open:Nn \l__graphics_internal_ior {#1}
2123     \ior_if_eof:NTF \l__graphics_internal_ior
2124     { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2125     {
2126       \dim_zero:N \l__graphics_llx_dim
2127       \dim_zero:N \l__graphics_lly_dim
2128       \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2129       \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2130       \ior_str_map_inline:Nn \l__graphics_internal_ior
2131       {
2132         \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2133         {
2134           \__graphics_backend_getbb_svg_auxi:nNn
2135           { width } \l__graphics_urx_dim {##1}
2136         }
2137         \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2138         {
2139           \__graphics_backend_getbb_svg_auxi:nNn
2140           { height } \l__graphics_ury_dim {##1}
2141         }
2142         \bool_lazy_and:nnF
2143         { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2144         { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2145         { \ior_map_break: }
2146       }
2147       \__graphics_bb_save:n {#1}
2148     }
2149     \ior_close:N \l__graphics_internal_ior
2150   }
2151 }
2152 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2153 {
2154   \use:e
2155   {
2156     \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxii:w
2157     ##1 \tl_to_str:n {#1} = ##2 \tl_to_str:n {#1} = ##3
2158     \s__graphics_stop
2159   }
2160   {
2161     \tl_if_blank:nF {##2}
2162     {
2163       \peek_remove_spaces:n
2164       {
2165         \peek_meaning:NTF ' % '
2166         { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2167         {
2168           \peek_meaning:NTF " % "
2169           { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2170           { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2171         }
2172       }

```



```

2172         }
2173         ##2 \s__graphics_stop
2174     }
2175 }
2176 \use:e
2177 {
2178     \__graphics_backend_getbb_svg_auxii:w #3
2179     \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2180     \s__graphics_stop
2181 }
2182 }
2183 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2184 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii: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_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2187 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2188 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2189 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2190 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2191 {
2192     \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2193     \l__graphics_internal_dim #2 bp \scan_stop:
2194     \dim_set_eq:NN #1 \l__graphics_internal_dim
2195 }
2196 \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.

```

2197 \__graphics_backend_loaded:n
2198 {
2199     \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2200     \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
2201 }

```

(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.

```

2202 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2203 {
2204     \int_zero:N \l__graphics_page_int
2205     \tl_clear:N \l__graphics_pagebox_tl
2206     \__graphics_extract_bb:n {#1}
2207 }
2208 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2209 \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

```

2210 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2211 {
2212     \tl_clear:N \l__graphics_decodearray_str
2213     \bool_set_false:N \l__graphics_interpolate_bool

```

```

2214     \_graphics_extract_bb:n {#1}
2215   }

```

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

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

```

2216 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
2217   { \_graphics_backend_include:nn { PSfile } {#1} }
2218 \cs_new_eq:NN \_graphics_backend_include_ps:n \_graphics_backend_include_eps:n
2219 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
2220   { \_graphics_backend_include:nn { pdffile } {#1} }
2221 \cs_new_protected:Npn \_graphics_backend_include:nn #1#2
2222   {
2223     \_kernel_backend_literal:e
2224     {
2225       #1 = #2 \c_space_tl
2226       llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
2227       lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2228       urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2229       ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
2230     }
2231   }

```

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

`_graphics_backend_include_svg:n` `_graphics_backend_include_png:n` `_graphics_backend_include_jpg:n` `_graphics_backend_include_jpeg:n` `_graphics_backend_include_dequote:w` The backend here has built-in support for basic graphic inclusion (see `dvismgm.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 `dvismgm: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.

```

2232 \cs_new_protected:Npn \_graphics_backend_include_svg:n #1
2233   {
2234     \box_move_up:nn { \l_graphics_ury_dim }
2235     {
2236       \hbox:n
2237       {
2238         \_kernel_backend_literal:e
2239         {
2240           dvisvgm:img~
2241           \dim_to_decimal:n { \l_graphics_urx_dim } ~
2242           \dim_to_decimal:n { \l_graphics_ury_dim } ~
2243           \_graphics_backend_include_dequote:w #1 " #1 " \s_graphics_stop
2244         }
2245       }
2246     }
2247   }
2248 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_svg:n
2249 \cs_new_eq:NN \_graphics_backend_include_jpeg:n \_graphics_backend_include_svg:n
2250 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_svg:n
2251 \cs_new:Npn \_graphics_backend_include_dequote:w #1 " #2 " #3 \s_graphics_stop
2252   {#2}

```

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

```
\_graphics_backend_get_pagecount:n
```

```
2253 \_graphics_backend_loaded:n  
2254 { \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n }  
  
(End of definition for \_graphics_backend_get_pagecount:n.)  
2255 </dvisvgm>  
2256 </package>
```

6 I3backend-pdf implementation

```
2257 <*package>  
2258 <@@=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 Raatz 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.

```
2259 <*!dvisvgm>
```

```
\l__pdf_internal_box
```

```
2260 \box_new:N \l__pdf_internal_box  
  
(End of definition for \l__pdf_internal_box.)  
2261 </!dvisvgm>
```

6.2 dvips backend

```
2262 <*dvips>
```

```
\__pdf_backend_pdfmark:n  
\__pdf_backend_pdfmark:e
```

Used often enough it should be a separate function.

```
2263 \cs_new_protected:Npn \__pdf_backend_pdfmark:n #1  
2264 { \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }  
2265 \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
```

```
2266 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2  
2267 { \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }  
2268 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2  
2269 { \__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

```

\__pdf_backend_object_new:
\__pdf_backend_object_ref:n 2270 \cs_new_protected:Npn \__pdf_backend_object_new:
\__pdf_backend_object_id:n 2271 { \int_gincr:N \g__pdf_backend_object_int }
2272 \cs_new:Npn \__pdf_backend_object_ref:n #1 { { pdf.obj #1 } }
2273 \cs_new_eq:NN \__pdf_backend_object_id:n \__pdf_backend_object_ref:n

```

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

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

```

\__pdf_backend_object_write:nnn 2274 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
\__pdf_backend_object_write:nne 2275 {
\__pdf_backend_object_write_aux:nnn 2276   \__pdf_backend_object_write_aux:nnn
\__pdf_backend_object_write_array:nn 2277   { \__pdf_backend_object_ref:n {#1} }
\__pdf_backend_object_write_dict:nn 2278   {#2} {#3}
\__pdf_backend_object_write_fstream:nnn 2279 }
\__pdf_backend_object_write_stream:nnn 2280 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2281 \cs_new_protected:Npn \__pdf_backend_object_write_aux:nnn #1#2#3
2282 {
2283   \__pdf_backend_pdfmark:e
2284   {
2285     /_objdef ~ #1
2286     /type
2287     \str_case:nn {#2}
2288     {
2289       { array } { /array }
2290       { dict } { /dict }
2291       { fstream } { /stream }
2292       { stream } { /stream }
2293     }
2294     /OBJ
2295   }
2296   \use:c { __pdf_backend_object_write_ #2 :nn } {#1} {#3}
2297 }
2298 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2299 {
2300   \__pdf_backend_pdfmark:e
2301   { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2302 }
2303 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2304 {
2305   \__pdf_backend_pdfmark:e
2306   { #1 << \exp_not:n {#2} >> /PUT }
2307 }
2308 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2309 {
2310   \exp_args:Ne
2311   \__pdf_backend_object_write_fstream:nnn {#1} #2
2312 }
2313 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2314 {

```

```

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

```

(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
2336 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2337 {
2338   \int_gincr:N \g__pdf_backend_object_int
2339   \__pdf_backend_object_write_aux:nnn
2340   { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2341   {#1} {#2}
2342 }
2343 \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.

```

2344 \cs_new:Npn \__pdf_backend_object_last:
2345 { { 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.

```

2346 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2347 { { 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.

```

2348 \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.

```
2349 \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.

```
2350 \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 L^AT_EX 2_ε picture of zero size). Once the data is collected, use it to set up the annotation border.

```
2351 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
```

```
2352 {
```

```
2353   \exp_args:Nf \__pdf_backend_annotation_aux:nnnn
```

```
2354     { \dim_eval:n {#1} } {#2} {#3} {#4}
```

```
2355 }
```

```
2356 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
```

```
2357 {
```

```
2358   \box_move_down:nn {#3}
```

```
2359   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
```

```
2360   \box_move_up:nn {#2}
```

```
2361     {
```

```
2362       \hbox:n
```

```
2363         {
```

```
2364           \__kernel_kern:n {#1}
```

```
2365           \__kernel_backend_postscript:n { pdf.save.ur }
```

```
2366           \__kernel_kern:n { -#1 }
```

```
2367         }
```

```
2368       }
```

```
2369   \int_gincr:N \g__pdf_backend_object_int
```

```
2370   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
```

```
2371   \__pdf_backend_pdfmark:e
```

```
2372     {
```

```
2373       /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
```

```
2374       pdf.rect
```

```
2375       #4 ~
```

```
2376       /ANN
```

```
2377     }
```

```
2378   }
```

(End of definition for __pdf_backend_annotation:nnnn.)

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

```
2379 \cs_new:Npn \__pdf_backend_annotation_last:
```

```
2380   { { 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.
²³⁸¹ `\int_new:N \g__pdf_backend_link_int`
(End of definition for `\g__pdf_backend_link_int`.)

`\g__pdf_backend_link_dict_tl` To pass information to the end-of-link function.
²³⁸² `\tl_new:N \g__pdf_backend_link_dict_tl`
(End of definition for `\g__pdf_backend_link_dict_tl`.)

`\g__pdf_backend_link_sf_int` Needed to save/restore space factor, which is needed to deal with the fact we need a box.
²³⁸³ `\int_new:N \g__pdf_backend_link_sf_int`
(End of definition for `\g__pdf_backend_link_sf_int`.)

`\g__pdf_backend_link_math_bool` Needed to save/restore math mode.
²³⁸⁴ `\bool_new:N \g__pdf_backend_link_math_bool`
(End of definition for `\g__pdf_backend_link_math_bool`.)

`\g__pdf_backend_link_bool` Track link formation: we cannot nest at all.
²³⁸⁵ `\bool_new:N \g__pdf_backend_link_bool`
(End of definition for `\g__pdf_backend_link_bool`.)

`\l__pdf_breaklink_pdfmark_tl` Swappable content for link breaking.
²³⁸⁶ `\tl_new:N \l__pdf_breaklink_pdfmark_tl`
²³⁸⁷ `\tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdfmark }`
(End of definition for `\l__pdf_breaklink_pdfmark_tl`.)

`__pdf_breaklink_postscript:n` To allow dropping material unless link breaking is active.
²³⁸⁸ `\cs_new_protected:Npn __pdf_breaklink_postscript:n #1 { }`
(End of definition for `__pdf_breaklink_postscript:n`.)

`__pdf_breaklink_usebox:N` Swappable box unpacking or use.
²³⁸⁹ `\cs_new_eq:MN __pdf_breaklink_usebox:N \box_use:N`
(End of definition for `__pdf_breaklink_usebox:N`.)

`__pdf_backend_link_begin_goto:nw` Links are created like annotations but with dedicated code to allow for adjusting the size
`__pdf_backend_link_begin_user:nw` of the rectangle. In contrast to `hyperref`, we grab the link content as a box which can
`__pdf_backend_link:nw` then unbox: this allows the same interface as for `pdfTeX`.
`__pdf_backend_link_aux:nw` Notice that the link setup here uses `/Action` not `/A`. That is because Distiller *requires*
`__pdf_backend_link_end:` this trigger word, rather than a “raw” PDF dictionary key (Ghostscript can handle either
`__pdf_backend_link_end_aux:` form).
`__pdf_backend_link_minima:` Taking the idea of `evenboxes` from `hypdvips`, we implement a minimum box height
`__pdf_backend_link_outerbox:n` and depth for link placement. This means that “underlining” with a hyperlink will
`__pdf_backend_link_sf_save:` generally give an even appearance. However, to ensure that the full content is always
`__pdf_backend_link_sf_restore:` 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.

```

2390 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2391 {
2392   \__pdf_backend_link_begin:nw
2393   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2394 }
2395 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
2396 { \__pdf_backend_link_begin:nw {#1#2} }
2397 \cs_new_protected:Npn \__pdf_backend_link_begin:nw #1
2398 {
2399   \bool_if:NF \g__pdf_backend_link_bool
2400   { \__pdf_backend_link_begin_aux:nw {#1} }
2401 }

```

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.

```

2402 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2403 {
2404   \bool_gset_true:N \g__pdf_backend_link_bool
2405   \__kernel_backend_postscript:n
2406   { /pdf.link.dict ( #1 ) def }
2407   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2408   \__pdf_backend_link_sf_save:
2409   \mode_if_math:TF
2410   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2411   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2412   \hbox_set:Nw \l__pdf_backend_content_box
2413   \__pdf_backend_link_sf_restore:
2414   \bool_if:NT \g__pdf_backend_link_math_bool
2415   { \c_math_toggle_token }
2416 }
2417 \cs_new_protected:Npn \__pdf_backend_link_end:
2418 {
2419   \bool_if:NT \g__pdf_backend_link_bool
2420   { \__pdf_backend_link_end_aux: }
2421 }
2422 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2423 {
2424   \bool_if:NT \g__pdf_backend_link_math_bool
2425   { \c_math_toggle_token }
2426   \__pdf_backend_link_sf_save:
2427   \hbox_set_end:
2428   \__pdf_backend_link_minima:
2429   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2430   \exp_args:Ne \__pdf_backend_link_outerbox:n
2431   {
2432     \int_if_odd:nTF { \value { page } }
2433     { \oddsidemargin }
2434     { \evensidemargin }
2435   }
2436   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }

```



```

2437     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2438 \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2439 \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2440 \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2441 \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2442   {
2443     \hbox:n
2444       { \__kernel_backend_postscript:n { pdf.save.linkur } }
2445   }
2446 \int_gincr:N \g__pdf_backend_object_int
2447 \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2448 \__kernel_backend_postscript:e
2449   {
2450     mark
2451     /objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2452     \g__pdf_backend_link_dict_tl \c_space_tl
2453     pdf.rect
2454     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2455   }
2456 \__pdf_backend_link_sf_restore:
2457 \bool_gset_false:N \g__pdf_backend_link_bool
2458 }
2459 \cs_new_protected:Npn \__pdf_backend_link_minima:
2460   {
2461     \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2462     \__kernel_backend_postscript:e
2463     {
2464       /pdf.linkdp.pad ~
2465       \dim_to_decimal:n
2466         {
2467           \dim_max:nn
2468             {
2469               \box_dp:N \l__pdf_backend_model_box
2470               - \box_dp:N \l__pdf_backend_content_box
2471             }
2472           { Opt }
2473         } ~
2474         pdf.pt.dvi ~ def
2475       /pdf.linkht.pad ~
2476       \dim_to_decimal:n
2477         {
2478           \dim_max:nn
2479             {
2480               \box_ht:N \l__pdf_backend_model_box
2481               - \box_ht:N \l__pdf_backend_content_box
2482             }
2483           { Opt }
2484         } ~
2485         pdf.pt.dvi ~ def
2486     }
2487   }
2488 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2489   {
2490     \__kernel_backend_postscript:e

```

```

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

```

(End of definition for `__pdf_backend_link_begin_goto:nmw` 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.

```

2524 \use_none:n
2525 {
2526   \cs_if_exist:NT \@makecol@hook
2527   {
2528     \tl_put_right:Nn \@makecol@hook
2529     {
2530       \box_if_empty:NF \l_shipout_box
2531       {
2532         \vbox_set:Nn \l_shipout_box
2533         {
2534           \__kernel_backend_postscript:n
2535           {
2536             pdf.globaldict /pdf.brokenlink.rect ~ known
2537             { pdf.bordertracking.continue }
2538             if
2539           }

```

```

2540         \vbox_unpack_drop:N \l_shipout_box
2541         \__kernel_backend_postscript:n
2542         { pdf.bordertracking.endpage }
2543     }
2544 }
2545 }
2546 \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2547 \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2548 \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2549 }
2550 }

```

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

```

2551 \cs_new:Npn \__pdf_backend_link_last:
2552 { { 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.

```

2553 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2554 {
2555     \__kernel_backend_postscript:e
2556     {
2557         /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2558     }
2559 }

```

(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:mnmn`
`_pdf_backend_destination_aux:mnmn`

```

2560 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2561 {
2562     \__kernel_backend_postscript:n { pdf.dest.anchor }
2563     \__pdf_backend_pdfmark:e
2564     {
2565         /View
2566         [
2567             \str_case:nnF {#2}
2568             {
2569                 { xyz } { /XYZ ~ pdf.dest.point ~ null }
2570                 { fit } { /Fit }
2571                 { fitb } { /FitB }
2572                 { fitbh } { /FitBH ~ pdf.dest.y }
2573                 { fitbv } { /FitBV ~ pdf.dest.x }
2574                 { fith } { /FitH ~ pdf.dest.y }
2575                 { fitv } { /FitV ~ pdf.dest.x }
2576                 { fitr } { /Fit }
2577             }
2578             {
2579                 /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2580             }

```

```

2581     ]
2582     /Dest ( \exp_not:n {#1} ) cvn
2583     /DEST
2584   }
2585 }
2586 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2587 {
2588   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2589   { \dim_eval:n {#2} } {#1} {#3} {#4}
2590 }
2591 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2592 {
2593   \vbox_to_zero:n
2594   {
2595     \__kernel_kern:n {#4}
2596     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2597     \tex_vss:D
2598   }
2599   \__kernel_kern:n {#1}
2600   \vbox_to_zero:n
2601   {
2602     \__kernel_kern:n { -#3 }
2603     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2604     \tex_vss:D
2605   }
2606   \__kernel_kern:n { -#1 }
2607   \__pdf_backend_pdfmark:n
2608   {
2609     /View
2610     [
2611       /FitR ~
2612       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2613       pdf.urx ~ pdf.ury ~ pdf.dest2device
2614     ]
2615     /Dest ( #2 ) cvn
2616     /DEST
2617   }
2618 }

```

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

6.2.4 Structure

__pdf_backend_compresslevel:n
 __pdf_backend_compress_objects:n

Doable for the usual ps2pdf method.

```

2619 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2620 {
2621   \int_compare:nNnT {#1} = 0
2622   {
2623     \__kernel_backend_literal_postscript:n
2624     {
2625       /setdistillerparams ~ where
2626       { pop << /CompressPages ~ false >> setdistillerparams }
2627     if

```

```

2628     }
2629   }
2630 }
2631 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2632 {
2633   \bool_if:nF {#1}
2634   {
2635     \__kernel_backend_literal_postscript:n
2636     {
2637       /setdistillerparams ~ where
2638       { pop << /CompressStreams ~ false >> setdistillerparams }
2639       if
2640     }
2641   }
2642 }

```

(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

```

2643 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2644 {
2645   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2646 }
2647 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2648 {
2649   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2650 }

```

(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: 2651 \cs_new:Npn __pdf_backend_version_major: { -1 }
 2652 \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

__pdf_backend_bdc:nn Simple wrappers.

```

\__pdf_backend_emc:
2653 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2654 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2655 \cs_new_protected:Npn \__pdf_backend_emc:
2656 { \__pdf_backend_pdfmark:n { /EMC } }

```

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

2657 </dvips>

6.3 LuaTeX and pdfTeX backend

2658 $\langle *luatex | pdftex \rangle$

6.3.1 Annotations

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

```
2659 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2660 {
2661    $\langle *luatex \rangle$ 
2662     \tex_pdfextension:D annot ~
2663    $\langle /luatex \rangle$ 
2664    $\langle *pdftex \rangle$ 
2665     \tex_pdfannot:D
2666    $\langle /pdftex \rangle$ 
2667     width ~ \dim_eval:n {#1} ~
2668     height ~ \dim_eval:n {#2} ~
2669     depth ~ \dim_eval:n {#3} ~
2670     {#4}
2671 }
```

(End of definition for $\backslash_pdf_backend_annotation:nnnn$.)

$\backslash_pdf_backend_annotation_last:$ 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.

```
2672 \cs_new:Npe \__pdf_backend_annotation_last:
2673 {
2674   \exp_not:N \int_value:w
2675    $\langle *luatex \rangle$ 
2676     \exp_not:N \tex_pdffeedback:D lastannot ~
2677    $\langle /luatex \rangle$ 
2678    $\langle *pdftex \rangle$ 
2679     \exp_not:N \tex_pdflastannot:D
2680    $\langle /pdftex \rangle$ 
2681     \c_space_tl 0 ~ R
2682 }
```

(End of definition for $\backslash_pdf_backend_annotation_last:$.)

$\backslash_pdf_backend_link_begin_goto:nnw$ Links are all created using the same internals.

```
\_pdf_backend_link_begin_user:nnw
\_pdf_backend_link_begin:nnnw
\_pdf_backend_link_end:
2683 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2684 { \__pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2685 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
2686 { \__pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2687 \cs_new_protected:Npn \__pdf_backend_link_begin:nnnw #1#2#3
2688 {
2689    $\langle *luatex \rangle$ 
2690     \tex_pdfextension:D startlink ~
2691    $\langle /luatex \rangle$ 
2692    $\langle *pdftex \rangle$ 
2693     \tex_pdfstartlink:D
2694    $\langle /pdftex \rangle$ 
2695     attr {#1}
2696     #2 {#3}
```

```

2697 }
2698 \cs_new_protected:Npn \__pdf_backend_link_end:
2699 {
2700 <*luatex>
2701   \tex_pdfextension:D endlink \scan_stop:
2702 </luatex>
2703 <*pdftex>
2704   \tex_pdfendlink:D
2705 </pdftex>
2706 }

```

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

__pdf_backend_link_last: Formatted for direct use.

```

2707 \cs_new:Npe \__pdf_backend_link_last:
2708 {
2709   \exp_not:N \int_value:w
2710 <*luatex>
2711   \exp_not:N \tex_pdffeedback:D lastlink ~
2712 </luatex>
2713 <*pdftex>
2714   \exp_not:N \tex_pdflastlink:D
2715 </pdftex>
2716   \c_space_tl 0 ~ R
2717 }

```

(End of definition for __pdf_backend_link_last:.)

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

```

2718 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2719 {
2720 <*luatex>
2721   \tex_pdfvariable:D linkmargin
2722 </luatex>
2723 <*pdftex>
2724   \tex_pdflinkmargin:D
2725 </pdftex>
2726   \dim_eval:n {#1} \scan_stop:
2727 }

```

(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:nmmn

```

2728 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2729 {
2730 <*luatex>
2731   \tex_pdfextension:D dest ~
2732 </luatex>
2733 <*pdftex>
2734   \tex_pdfdest:D
2735 </pdftex>
2736   name {#1}
2737   \str_case:nnF {#2}

```

```

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

```

(End of definition for __pdf_backend_destination:nn and __pdf_backend_destination:nmmm.)

6.3.2 Catalogue entries

```

\__pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
2765 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2766 {
2767   \*luatex
2768   \tex_pdfextension:D catalog
2769   \*pdftex
2770   \tex_pdfcatalog:D
2771   \*pdftex
2772   { / #1 ~ #2 }
2773 }
2774 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2775 {
2776   \*luatex
2777   \tex_pdfextension:D info
2778   \*pdftex
2779   \tex_pdfinfo:D
2780   \*pdftex
2781   { / #1 ~ #2 }
2782 }

```

(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.

```
2785 \prop_new:N \g__pdf_backend_object_prop
```

(End of definition for `\g__pdf_backend_object_prop`.)

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

```
\__pdf_backend_object_ref:n 2786 \cs_new_protected:Npn \__pdf_backend_object_new:
```

```
\__pdf_backend_object_id:n
```

```
2787 {
2788 <*luatex>
2789 \tex_pdfextension:D obj ~
2790 </luatex>
2791 <*pdftex>
2792 \tex_pdfobj:D
2793 </pdftex>
2794 reserveobjnum ~
2795 \int_gset:Nn \g__pdf_backend_object_int
2796 <*luatex>
2797 { \tex_pdffeedback:D lastobj }
2798 </luatex>
2799 <*pdftex>
2800 { \tex_pdflastobj:D }
2801 </pdftex>
2802 }
2803 \cs_new:Npn \__pdf_backend_object_ref:n #1 { #1 ~ 0 ~ R }
2804 \cs_new:Npn \__pdf_backend_object_id:n #1 {#1}
```

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

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

```
\__pdf_backend_object_write:nne 2805 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
```

```
\__pdf_backend_object_write:nn
```

```
\__pdf_exp_not_i:nn
```

```
\__pdf_exp_not_ii:nn
```

```
2806 {
2807 <*luatex>
2808 \tex_immediate:D \tex_pdfextension:D obj ~
2809 </luatex>
2810 <*pdftex>
2811 \tex_immediate:D \tex_pdfobj:D
2812 </pdftex>
2813 useobjnum ~ #1
2814 \__pdf_backend_object_write:nn {#2} {#3}
2815 }
2816 \cs_new:Npn \__pdf_backend_object_write:nn #1#2
2817 {
2818 \str_case:nn {#1}
2819 {
2820 { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2821 { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2822 { fstream }
2823 {
2824 stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2825 file ~ { \__pdf_exp_not_ii:nn #2 }
2826 }
2827 { stream }
```

```

2828     {
2829         stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2830         { \_pdf_exp_not_ii:nn #2 }
2831     }
2832 }
2833 }
2834 \cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nne }
2835 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2836 \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 2837 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2838 {
2839 <*luatex>
2840     \tex_immediate:D \tex_pdfextension:D obj ~
2841 </luatex>
2842 <*pdftex>
2843     \tex_immediate:D \tex_pdfobj:D
2844 </pdftex>
2845     \_pdf_backend_object_write:nn {#1} {#2}
2846 }
2847 \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.

```

2848 \cs_new:Npe \_pdf_backend_object_last:
2849 {
2850     \exp_not:N \int_value:w
2851 <*luatex>
2852     \exp_not:N \tex_pdffeedback:D lastobj ~
2853 </luatex>
2854 <*pdftex>
2855     \exp_not:N \tex_pdflastobj:D
2856 </pdftex>
2857     \c_space_tl 0 ~ R
2858 }

```

(End of definition for _pdf_backend_object_last:.)

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

```

2859 \cs_new:Npe \_pdf_backend_pageobject_ref:n #1
2860 {
2861     \exp_not:N \int_value:w
2862 <*luatex>
2863     \exp_not:N \tex_pdffeedback:D pageref
2864 </luatex>
2865 <*pdftex>
2866     \exp_not:N \tex_pdfpageref:D
2867 </pdftex>
2868     \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2869 }

```

(End of definition for _pdf_backend_pageobject_ref:n.)

6.3.4 Structure

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
\__pdf_backend_objcompresslevel:n
2870 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2871 {
2872   \tex_global:D
2873   \*luatex
2874   \tex_pdfvariable:D compresslevel
2875   \*pdfTeX
2876   \tex_pdfcompresslevel:D
2877   \*pdfTeX
2878   \int_value:w \int_eval:n {#1} \scan_stop:
2879 }
2880
2881 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2882 {
2883   \bool_if:nTF {#1}
2884     { \__pdf_backend_objcompresslevel:n { 2 } }
2885     { \__pdf_backend_objcompresslevel:n { 0 } }
2886 }
2887 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2888 {
2889   \tex_global:D
2890   \*luatex
2891   \tex_pdfvariable:D objcompresslevel
2892   \*pdfTeX
2893   \tex_pdfobjcompresslevel:D
2894   \*pdfTeX
2895   #1 \scan_stop:
2896 }
2897

```

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

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

```

2898 \cs_new_protected:Npe \__pdf_backend_version_major_gset:n #1
2899 {
2900   \*luatex
2901   \int_compare:nNnT \tex_luaTeXversion:D > { 106 }
2902   {
2903     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2904     \exp_not:N \int_eval:n {#1} \scan_stop:
2905   }
2906   \*pdfTeX
2907   \cs_if_exist:NT \tex_pdfmajorversion:D
2908   {
2909     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2910     \exp_not:N \int_eval:n {#1} \scan_stop:
2911   }
2912   \*pdfTeX
2913 }
2914 }
2915 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2916 {

```

```

2917     \tex_global:D
2918 <*luatex>
2919     \tex_pdfvariable:D minorversion
2920 </luatex>
2921 <*pdftex>
2922     \tex_pdfminorversion:D
2923 </pdftex>
2924     \int_eval:n {#1} \scan_stop:
2925 }

```

(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:
2926 \cs_new:Npe \__pdf_backend_version_major:
2927 {
2928 <*luatex>
2929     \int_compare:nNnTF \tex luatexversion:D > { 106 }
2930     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2931     { 1 }
2932 </luatex>
2933 <*pdftex>
2934     \cs_if_exist:NTF \tex_pdfmajorversion:D
2935     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2936     { 1 }
2937 </pdftex>
2938 }
2939 \cs_new:Npn \__pdf_backend_version_minor:
2940 {
2941     \tex_the:D
2942 <*luatex>
2943     \tex_pdfvariable:D minorversion
2944 </luatex>
2945 <*pdftex>
2946     \tex_pdfminorversion:D
2947 </pdftex>
2948 }

```

(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:
2949 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2950 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2951 \cs_new_protected:Npn \__pdf_backend_emc:
2952 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2953 </luatex | pdftex>

```

6.4 dvipdfmx backend

2954 `*dvipdfmx|xetex)`

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

```

\__pdf_backend:e
2955 \cs_new_protected:Npe \__pdf_backend:n #1
2956 { \__kernel_backend_literal:n { pdf: #1 } }
2957 \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
2958 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2959 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2960 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2961 { \__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_prop` For tracking objects to allow finalisation.

```
2962 \prop_new:N \g__pdf_backend_object_prop
```

(End of definition for `\g__pdf_backend_object_prop`.)

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

```

\__pdf_backend_object_ref:n
\__pdf_backend_object_id:n
2963 \cs_new_protected:Npn \__pdf_backend_object_new:
2964 { \int_gincr:N \g__pdf_backend_object_int }
2965 \cs_new:Npn \__pdf_backend_object_ref:n #1 { @pdf.obj #1 }
2966 \cs_new_eq:NN \__pdf_backend_object_id:n \__pdf_backend_object_ref:n

```

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

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

```

\__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
2967 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2968 {
2969   \use:c { __pdf_backend_object_write_ #2 :nn }
2970   { \__pdf_backend_object_ref:n {#1} } {#3}
2971 }
2972 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2973 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2974 {
2975   \__pdf_backend:e
2976   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2977 }
2978 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2979 {
2980   \__pdf_backend:e
2981   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2982 }
2983 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2

```

```

2984 { \_pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2985 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nn #1#2
2986 { \_pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2987 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nnnn #1#2#3#4
2988 {
2989   \_pdf_backend:e
2990   {
2991     #1 stream ~ #2 ~
2992     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2993   }
2994 }

```

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

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

```

2995 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2996 {
2997   \int_gincr:N \g__pdf_backend_object_int
2998   \exp_args:Nne \use:c { \_pdf_backend_object_write_ #1 :nn }
2999   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
3000   {#2}
3001 }
3002 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { ne }

```

(End of definition for _pdf_backend_object_now:nn.)

_pdf_backend_object_last:

```

3003 \cs_new:Npn \_pdf_backend_object_last:
3004 { @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_YT_EX.

```

3005 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
3006 { @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.

```

3007 \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.

```

3008 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
3009 {
3010   \int_gincr:N \g__pdf_backend_object_int
3011   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
3012   \_pdf_backend:e
3013   {
3014     ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
3015     width ~ \dim_eval:n {#1} ~
3016     height ~ \dim_eval:n {#2} ~

```

```

3017         depth ~ \dim_eval:n {#3} ~
3018         << /Type /Annot #4 >>
3019     }
3020 }

```

(End of definition for _pdf_backend_annotation:nmnn.)

_pdf_backend_annotation_last:

```

3021 \cs_new:Npn \_pdf_backend_annotation_last:
3022 { @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.

```

3023 \int_new:N \g__pdf_backend_link_int

```

(End of definition for \g__pdf_backend_link_int.)

_pdf_backend_link_begin_goto:nmw All created using the same internals.

_pdf_backend_link_begin_user:nmw

_pdf_backend_link_begin:n

_pdf_backend_link_end:

```

3024 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nmw #1#2
3025 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
3026 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nmw #1#2
3027 { \_pdf_backend_link_begin:n {#1#2} }
3028 \cs_new_protected:Npe \_pdf_backend_link_begin:n #1
3029 {
3030     \exp_not:N \int_gincr:N \exp_not:N \g__pdf_backend_link_int
3031     \_pdf_backend:e
3032     {
3033         bann ~
3034         @pdf.lnk
3035         \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
3036         \c_space_tl
3037         <<
3038         /Type /Annot
3039         #1
3040         >>
3041     }
3042 }
3043 \cs_new_protected:Npn \_pdf_backend_link_end:
3044 { \_pdf_backend:n { eann } }

```

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

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

```

3045 \cs_new:Npn \_pdf_backend_link_last:
3046 { @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.

```

3047 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
3048 { \_kernel_backend_literal:e { dvipdfmx:config-g~ \dim_eval:n {#1} } }

```

(End of definition for _pdf_backend_link_margin:n.)

`_pdf_backend_destination:nn`
`_pdf_backend_destination:nmmn`
`_pdf_backend_destination_aux:nmmn`

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

```

3049 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
3050 {
3051   \_pdf_backend:e
3052   {
3053     dest ~ ( \exp_not:n {#1} )
3054     [
3055       @thispage
3056       \str_case:nnF {#2}
3057       {
3058         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
3059         { fit } { /Fit }
3060         { fitb } { /FitB }
3061         { fitbh } { /FitBH }
3062         { fitbv } { /FitBV ~ @xpos }
3063         { fith } { /FitH ~ @ypos }
3064         { fitv } { /FitV ~ @xpos }
3065         { fitr } { /Fit }
3066       }
3067       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
3068     ]
3069   }
3070 }
3071 \cs_new_protected:Npn \_pdf_backend_destination:nmmn #1#2#3#4
3072 {
3073   \exp_args:Ne \_pdf_backend_destination_aux:nmmn
3074   { \dim_eval:n {#2} } {#1} {#3} {#4}
3075 }
3076 \cs_new_protected:Npn \_pdf_backend_destination_aux:nmmn #1#2#3#4
3077 {
3078   \vbox_to_zero:n
3079   {
3080     \_kernel_kern:n {#4}
3081     \hbox:n
3082     {
3083       \_pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
3084       \_pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
3085     }
3086     \tex_vss:D
3087   }
3088   \_kernel_kern:n {#1}
3089   \vbox_to_zero:n
3090   {
3091     \_kernel_kern:n { -#3 }
3092     \hbox:n
3093     {
3094       \_pdf_backend:n
3095       {
3096         dest ~ (#2)
3097         [
3098           @thispage

```



```

3099             /FitR ~
3100             @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
3101             @xpos ~ @ypos
3102         ]
3103     }
3104 }
3105 \tex_vss:D
3106 }
3107 \__kernel_kern:n { -#1 }
3108 }

```

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

6.4.4 Structure

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

```

3109 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
3110 { \__kernel_backend_literal:e { dvipdfmx:config~z~ \int_eval:n {#1} } }
3111 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
3112 {
3113     \bool_if:nF {#1}
3114     { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
3115 }

```

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

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

```

3116 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
3117 {
3118     \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
3119     \__kernel_backend_literal:e { pdf:majorversion~ \__pdf_backend_version_major: }
3120 }
3121 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
3122 {
3123     \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
3124     \__kernel_backend_literal:e { pdf:minorversion~ \__pdf_backend_version_minor: }
3125 }

```

(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:

```

3126 \cs_new:Npn \__pdf_backend_version_major: { 1 }
3127 \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:

```

3128 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
3129 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
3130 \cs_new_protected:Npn \__pdf_backend_emc:
3131 { \__kernel_backend_literal_page:n { EMC } }

```

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

3132 `</dvipdfmx | xetex>`

6.5 dvisvgm backend

3133 `<*dvisvgm>`

6.5.1 Annotations

`_pdf_backend_annotation:nnnn`

3134 `\cs_new_protected:Npn _pdf_backend_annotation:nnnn #1#2#3#4 { }`

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

`_pdf_backend_annotation_last:`

3135 `\cs_new:Npn _pdf_backend_annotation_last: { }`

(End of definition for `_pdf_backend_annotation_last:.`)

`_pdf_backend_link_begin_goto:nnw`

`_pdf_backend_link_begin_user:nnw`

`_pdf_backend_link_begin:nnnw`

`_pdf_backend_link_end:`

3136 `\cs_new_protected:Npn _pdf_backend_link_begin_goto:nnw #1#2 { }`

3137 `\cs_new_protected:Npn _pdf_backend_link_begin_user:nnw #1#2 { }`

3138 `\cs_new_protected:Npn _pdf_backend_link_begin:nnnw #1#2#3 { }`

3139 `\cs_new_protected:Npn _pdf_backend_link_end: { }`

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

`_pdf_backend_link_last:`

3140 `\cs_new:Npe _pdf_backend_link_last: { }`

(End of definition for `_pdf_backend_link_last:.`)

`_pdf_backend_link_margin:n`

A simple task: pass the data to the primitive.

3141 `\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`

3142 `\cs_new_protected:Npn _pdf_backend_destination:nn #1#2 { }`

3143 `\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

`_pdf_backend_catalog_gput:nn`

No-op.

`_pdf_backend_info_gput:nn`

3144 `\cs_new_protected:Npn _pdf_backend_catalog_gput:nn #1#2 { }`

3145 `\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:
\_pdf_backend_object_ref:n 3146 \cs_new_protected:Npn \_pdf_backend_object_new: { }
\_pdf_backend_object_id:n 3147 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
  \_pdf_backend_object_write:nnn 3148 \cs_new:Npn \_pdf_backend_object_id:n #1 { }
  \_pdf_backend_object_write:ne 3149 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3 { }
\_pdf_backend_object_now:nn 3150 \cs_new_protected:Npn \_pdf_backend_object_write:nne #1#2#3 { }
\_pdf_backend_object_now:ne 3151 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
\_pdf_backend_object_last: 3152 \cs_new_protected:Npn \_pdf_backend_object_now:ne #1#2 { }
  \_pdf_backend_pageobject_ref:n 3153 \cs_new:Npn \_pdf_backend_object_last: { }
  3154 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1 { }
```

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

6.5.4 Structure

These are all no-ops.

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

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

Data not available!

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

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

Data not available!

```
\_pdf_backend_version_major: 3159 \cs_new:Npn \_pdf_backend_version_major: { -1 }
\_pdf_backend_version_minor: 3160 \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 3161 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2 { }
\_pdf_backend_emc: 3162 \cs_new_protected:Npn \_pdf_backend_emc: { }
```

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

```
3163 </dvisvgm>
```

6.6 PDF Page size (media box)

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

```
3164 <*dvipdfmx | dvips>
```

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

```
3165 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
3166 {
3167   \_kernel_backend_first_shipout:n
3168   {
3169     \_kernel_backend_literal:e
3170     {
3171       <*dvipdfmx>
3172         pdf:pagesize ~
3173         width ~ \dim_eval:n {#1} ~
3174         height ~ \dim_eval:n {#2}
3175       </dvipdfmx>
3176       <*dvips>
3177         papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
3178       </dvips>
3179     }
3180   }
3181 }

(End of definition for \_pdf_backend_pagesize_gset:nn.)

3182 </dvipdfmx | dvips>
3183 <*luatex | pdftex | xetex>
```

`_pdf_backend_pagesize_gset:nn` Pass to the primitives.

```
3184 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
3185 {
3186   \dim_gset:Nn \tex_pagewidth:D {#1}
3187   \dim_gset:Nn \tex_pageheight:D {#2}
3188 }

(End of definition for \_pdf_backend_pagesize_gset:nn.)

3189 </luatex | pdftex | xetex>
3190 <*dvisvgm>
```

`_pdf_backend_pagesize_gset:nn` A no-op.

```
3191 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2 { }

(End of definition for \_pdf_backend_pagesize_gset:nn.)

3192 </dvisvgm>
3193 </package>
```

7 I3backend-opacity implementation

```
3194 <*package>
3195 <@@=opacity>
```

Although opacity is not color, it needs to be managed in a somewhat similar way: using a dedicated stack if possible. Depending on the backend, that may not be possible. There is also the need to cover fill/stroke setting as well as more general running opacity. It is easiest to describe the value used in terms of opacity, although commonly this is referred to as transparency.

3196 <dvips>

_opacity_backend_select:n No stack so set values directly. The need to deal with Distiller and Ghostscript separately
_opacity_backend_fill:n means we use a common auxiliary: the two systems require different PostScript for
_opacity_backend_stroke:n transparency. This is of course not quite as efficient as doing one test for setting all
_opacity_backend:nnn transparency, but it keeps things clearer here. Thanks to Alex Grahn for the detail on
testing for GhostScript.

```
3197 \cs_new_protected:Npn \_opacity_backend_select:n #1
3198 {
3199   \_opacity_backend:nnn {#1} { fill } { ca }
3200   \_opacity_backend:nnn {#1} { stroke } { CA }
3201 }
3202 \cs_new_protected:Npn \_opacity_backend_fill:n #1
3203 {
3204   \_opacity_backend:nnn
3205     { #1 }
3206     { fill }
3207     { ca }
3208 }
3209 \cs_new_protected:Npn \_opacity_backend_stroke:n #1
3210 {
3211   \_opacity_backend:nnn
3212     { #1 }
3213     { stroke }
3214     { CA }
3215 }
3216 \cs_new_protected:Npn \_opacity_backend:nnn #1#2#3
3217 {
3218   \_kernel_backend_postscript:n
3219   {
3220     product ~ (Ghostscript) ~ search
3221     {
3222       pop ~ pop ~ pop ~
3223       #1 ~ .set #2 constantalpha
3224     }
3225     {
3226       pop ~
3227       mark ~
3228       /#3 ~ #1
3229       /SetTransparency ~
3230       pdfmark
3231     }
3232     ifelse
3233   }
3234 }
```

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

3235 </dvips>

3236 <dvipdfmx | luatex | pdftex | xetex>

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

```
3237 \bool_lazy_and:nnT
3238 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
```

```

3239 { \pdfmanagement_if_active_p: }
3240 {
3241 <*luatex | pdftex>
3242   \__kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3243   { page ~ direct } { /opacity 1 ~ gs }
3244 </luatex | pdftex>
3245   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3246   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3247 }

```

(End of definition for \c__opacity_backend_stack_int.)

\l__opacity_backend_fill_tl \l__opacity_backend_stroke_tl We use tl here for speed: at the backend, this should be reasonable. Both need to start off fully opaque.

```

3248 \tl_new:N \l__opacity_backend_fill_tl
3249 \tl_new:N \l__opacity_backend_stroke_tl
3250 \tl_set:Nn \l__opacity_backend_fill_tl { 1 }
3251 \tl_set:Nn \l__opacity_backend_stroke_tl { 1 }

```

(End of definition for \l__opacity_backend_fill_tl and \l__opacity_backend_stroke_tl.)

__opacity_backend_select:n Much the same as color.

```

\__opacity_backend_reset:
3252 \cs_new_protected:Npn \__opacity_backend_select:n #1
3253 {
3254   \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3255   \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3256   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3257   { opacity #1 }
3258   { << /ca ~ #1 /CA ~ #1 >> }
3259 <*dvipdfmx | xetex>
3260   \__kernel_backend_literal_pdf:n
3261 </dvipdfmx | xetex>
3262 <*luatex | pdftex>
3263   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3264 </luatex | pdftex>
3265   { /opacity #1 ~ gs }
3266   \group_insert_after:N \__opacity_backend_reset:
3267 }
3268 \cs_new_protected:Npn \__opacity_backend_reset:
3269 {
3270 <*dvipdfmx | xetex>
3271   \__kernel_backend_literal_pdf:n
3272   { /opacity1 ~ gs }
3273 </dvipdfmx | xetex>
3274 <*luatex | pdftex>
3275   \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int
3276 </luatex | pdftex>
3277 }

```

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

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

```

\__opacity_backend_fill_stroke:nn
3278 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3279 {

```

```

3280     \exp_args:Nno \__opacity_backend_fill_stroke:nn
3281     { #1 }
3282     { \l__opacity_backend_stroke_tl }
3283   }
3284 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3285 {
3286   \exp_args:No \__opacity_backend_fill_stroke:nn
3287   { \l__opacity_backend_fill_tl }
3288   { #1 }
3289 }
3290 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3291 {
3292   \str_if_eq:nnTF {#1} {#2}
3293   { \__opacity_backend_select:n {#1} }
3294   {
3295     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3296     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3297     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3298     { opacity.fill #1 }
3299     { << /ca ~ #1 >> }
3300     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3301     { opacity.stroke #2 }
3302     { << /CA ~ #2 >> }
3303 <*/dvipdfmx|xetex>
3304   \__kernel_backend_literal_pdf:n
3305 </dvipdfmx|xetex>
3306 <*/luatex|pdfTeX>
3307   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3308 </luatex|pdfTeX>
3309   { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3310   \group_insert_after:N \__opacity_backend_reset:
3311 }
3312 }

```

(End of definition for __opacity_backend_fill:n, __opacity_backend_stroke:n, and __opacity_backend_fill_stroke:nn.)

__opacity_backend_select:n Redefine them to stubs if pdfmanagement is either not loaded or deactivated.

```

\__opacity_backend_fill_stroke:nn 3313 \bool_lazy_and:nnF
3314 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3315 { \pdfmanagement_if_active_p: }
3316 {
3317   \cs_gset_protected:Npn \__opacity_backend_select:n #1 { }
3318   \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2 { }
3319 }

```

(End of definition for __opacity_backend_select:n and __opacity_backend_fill_stroke:nn.)

```

3320 </dvipdfmx|luatex|pdfTeX|xetex>

```

```

3321 <*/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 3322 \cs_new_protected:Npn \__opacity_backend_select:n #1
\__opacity_backend_stroke:n 3323 { \__opacity_backend:nn {#1} { } }
\__opacity_backend:nn

```

```

3324 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3325 { \__opacity_backend:nn {#1} { fill- } }
3326 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3327 { \__opacity_backend:nn {#1} { stroke- } }
3328 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3329 { \__kernel_backend_scope:e { #2 opacity = " #1 " } }

```

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

```
3330 </dvisvgm>
```

```
3331 </package>
```

7.1 Font handling integration

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

```
3332 <*lua>
```

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

```

3333 local pdfmanagement_active do
3334   local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3335   local cmd = pdfmanagement_if_active_p.cmdname
3336   if cmd == 'undefined_cs' then
3337     pdfmanagement_active = false
3338   else
3339     token.put_next(pdfmanagement_if_active_p)
3340     pdfmanagement_active = token.scan_int() ~= 0
3341   end
3342 end
3343
3344 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3345   luaotfload.set_transparent_colorstack(function() return token.create'c__opacity_backend_st
3346
3347   local transparent_register = {
3348     token.create'pdfmanagement_add:nnn',
3349     token.new(0, 1),
3350     'Page/Resources/ExtGState',
3351     token.new(0, 2),
3352     token.new(0, 1),
3353     '',
3354     token.new(0, 2),
3355     token.new(0, 1),
3356     '<</ca ',
3357     '',
3358     '/CA ',
3359     '',
3360     '>>',
3361     token.new(0, 2),
3362   }
3363   luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)
3364     value = (octet * -1):match(value)
3365     if not value then
3366       tex.error'Invalid transparency value'
3367     return

```



```

3368     end
3369     value = value:sub(1, -2)
3370     local result = 'opacity' .. value
3371     tex.runtoks(function()
3372         transparent_register[6], transparent_register[10], transparent_register[12] = result,
3373         tex.sprint(-2, transparent_register)
3374     end)
3375     return '/' .. result .. ' gs'
3376 end, 'l3opacity')
3377 end
3378 </lua>

```

8 l3backend-header implementation

```
3379 <*dvips & header>
```

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

```
3380 /color.sc { } def
```

(End of definition for color.sc.)

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

```
3381 TeXDict begin
3382 /TeXcolorseparation { setcolor } def
3383 end
```

(End of definition for TeXcolorseparation and separation.)

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

```
3384 true setglobal
3385 /pdf.globaldict 4 dict def
3386 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 3387 /pdf.cvs { 65534 string cvs } def
3388 /pdf.dvi.pt { 72.27 mul Resolution div } def
3389 /pdf.pt.dvi { 72.27 div Resolution mul } def
3390 /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 3391 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad 3392 /pdf.linkdp.pad { 0 } def
3393 /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	3394 /pdf.rect
pdf.save.linkur	3395 { /Rect [pdf.llx pdf.lly pdf.urx pdf.ury] } def
pdf.llx	3396 /pdf.save.ll
pdf.lly	3397 {
pdf.urx	3398 currentpoint
pdf.ury	3399 /pdf.lly exch def
	3400 /pdf.llx exch def
	3401 }
	3402 def
	3403 /pdf.save.ur
	3404 {
	3405 currentpoint
	3406 /pdf.ury exch def
	3407 /pdf.urx exch def
	3408 }
	3409 def
	3410 /pdf.save.linkll
	3411 {
	3412 currentpoint
	3413 pdf.linkmargin add
	3414 pdf.linkdp.pad add
	3415 /pdf.lly exch def
	3416 pdf.linkmargin sub
	3417 /pdf.llx exch def
	3418 }
	3419 def
	3420 /pdf.save.linkur
	3421 {
	3422 currentpoint
	3423 pdf.linkmargin sub
	3424 pdf.linkht.pad sub
	3425 /pdf.ury exch def
	3426 pdf.linkmargin add
	3427 /pdf.urx exch def
	3428 }
	3429 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
pdf.dest.x	function as it comes up a lot, and as this makes it easier to adjust if we need additional
pdf.dest.y	effects. We also need a more complex approach to convert a co-ordinate pair correctly
pdf.dest.point	when defining a rectangle: this can otherwise be out when using a landscape page.
pdf.dest2device	(Thanks to Alexander Grahn for the approach here.)
pdf.dev.x	3430 /pdf.dest.anchor
pdf.dev.y	3431 {
pdf.tmpa	3432 currentpoint exch
pdf.tmpb	3433 pdf.dvi.pt 72 add
pdf.tmpc	3434 /pdf.dest.x exch def
pdf.tmpd	3435 pdf.dvi.pt
	3436 vsize 72 sub exch sub

```

3437 /pdf.dest.y exch def
3438 }
3439 def
3440 /pdf.dest.point
3441 { pdf.dest.x pdf.dest.y } def
3442 /pdf.dest2device
3443 {
3444 /pdf.dest.y exch def
3445 /pdf.dest.x exch def
3446 matrix currentmatrix
3447 matrix defaultmatrix
3448 matrix invertmatrix
3449 matrix concatmatrix
3450 cvx exec
3451 /pdf.dev.y exch def
3452 /pdf.dev.x exch def
3453 /pdf.tmpd exch def
3454 /pdf.tmpc exch def
3455 /pdf.tmpb exch def
3456 /pdf.tmpa exch def
3457 pdf.dest.x pdf.tmpa mul
3458 pdf.dest.y pdf.tmpc mul add
3459 pdf.dev.x add
3460 pdf.dest.x pdf.tmpb mul
3461 pdf.dest.y pdf.tmpd mul add
3462 pdf.dev.y add
3463 }
3464 def

```

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

<pre> pdf.bordertracking pdf.bordertracking.begin pdf.bordertracking.end pdf.leftboundary pdf.rightboundary pdf.brokenlink.rect pdf.brokenlink.skip pdf.brokenlink.dict pdf.bordertracking.endpage pdf.bordertracking.continue pdf.originx pdf.originy </pre>	<p>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.</p> <pre> 3465 /pdf.bordertracking false def 3466 /pdf.bordertracking.begin 3467 { 3468 SDict /pdf.bordertracking true put 3469 SDict /pdf.leftboundary undef 3470 SDict /pdf.rightboundary undef 3471 /a where 3472 { 3473 /a 3474 { 3475 currentpoint pop 3476 SDict /pdf.rightboundary known dup 3477 { 3478 SDict /pdf.rightboundary get 2 index lt 3479 { not } 3480 if 3481 } 3482 if 3483 { pop } </pre>
---	---

```

3484         { SDict exch /pdf.rightboundary exch put }
3485     ifelse
3486     moveto
3487     currentpoint pop
3488     SDict /pdf.leftboundary known dup
3489     {
3490         SDict /pdf.leftboundary get 2 index gt
3491         { not }
3492         if
3493     }
3494     if
3495     { pop }
3496     { SDict exch /pdf.leftboundary exch put }
3497     ifelse
3498 }
3499 put
3500 }
3501 if
3502 }
3503 def
3504 /pdf.bordertracking.end
3505 {
3506     /a where { /a { moveto } put } if
3507     /x where { /x { 0 exch rmoveto } put } if
3508     SDict /pdf.leftboundary known
3509     { pdf.outerbox 0 pdf.leftboundary put }
3510     if
3511     SDict /pdf.rightboundary known
3512     { pdf.outerbox 2 pdf.rightboundary put }
3513     if
3514     SDict /pdf.bordertracking false put
3515 }
3516 def
3517 /pdf.bordertracking.endpage
3518 {
3519 pdf.bordertracking
3520     {
3521 pdf.bordertracking.end
3522 true setglobal
3523 pdf.globaldict
3524     /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3525 pdf.globaldict
3526     /pdf.brokenlink.skip pdf.baselineskip put
3527 pdf.globaldict
3528     /pdf.brokenlink.dict
3529 pdf.link.dict pdf.cvs put
3530 false setglobal
3531 mark pdf.link.dict cvx exec /Rect
3532 [
3533 pdf.llx
3534 pdf.lly
3535 pdf.outerbox 2 get pdf.linkmargin add
3536 currentpoint exch pop
3537 pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub

```

```

3538     ]
3539     /ANN pdf.pdfmark
3540   }
3541   if
3542 }
3543   def
3544 /pdf.bordertracking.continue
3545   {
3546     /pdf.link.dict pdf.globaldict
3547     /pdf.brokenlink.dict get def
3548     /pdf.outerbox pdf.globaldict
3549     /pdf.brokenlink.rect get def
3550     /pdf.baselineskip pdf.globaldict
3551     /pdf.brokenlink.skip get def
3552     pdf.globaldict dup dup
3553     /pdf.brokenlink.dict undef
3554     /pdf.brokenlink.skip undef
3555     /pdf.brokenlink.rect undef
3556     currentpoint
3557     /pdf.originy exch def
3558     /pdf.originx exch def
3559     /a where
3560     {
3561       /a
3562       {
3563         moveto
3564         SDict
3565         begin
3566         currentpoint pdf.originy ne exch
3567         pdf.originx ne or
3568         {
3569           pdf.save.linkll
3570           /pdf.lly
3571           pdf.lly pdf.outerbox 1 get sub def
3572           pdf.bordertracking.begin
3573         }
3574         if
3575         end
3576       }
3577       put
3578     }
3579   if
3580 /x where
3581   {
3582     /x
3583     {
3584       0 exch rmoveto
3585       SDict
3586       begin
3587       currentpoint
3588       pdf.originy ne exch pdf.originx ne or
3589       {
3590         pdf.save.linkll
3591         /pdf.lly

```

```

3592         pdf.lly pdf.outerbox 1 get sub def
3593         pdf.bordertracking.begin
3594     }
3595     if
3596     end
3597 }
3598 put
3599 }
3600 if
3601 }
3602 def

```

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

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

```

3603 /pdf.breaklink
3604 {
3605     pop
3606     counttomark 2 mod 0 eq
3607     {
3608         counttomark /pdf.count exch def
3609         {
3610             pdf.count 0 eq { exit } if
3611             counttomark 2 roll
3612             1 index /Rect eq
3613             {
3614                 dup 4 array copy
3615                 dup dup
3616                 1 get
3617                 pdf.outerbox pdf.rect.ht
3618                 pdf.linkmargin 2 mul add sub
3619                 3 exch put
3620                 dup
3621                 pdf.outerbox 2 get
3622                 pdf.linkmargin add
3623                 2 exch put
3624                 dup dup
3625                 3 get
3626                 pdf.outerbox pdf.rect.ht
3627                 pdf.linkmargin 2 mul add add
3628                 1 exch put
3629                 /pdf.currentrect exch def
3630                 pdf.breaklink.write
3631                 {
3632                     pdf.currentrect
3633                     dup
3634                     pdf.outerbox 0 get
3635                     pdf.linkmargin sub
3636                     0 exch put
3637                     dup

```

```

3638         pdf.outerbox 2 get
3639         pdf.linkmargin add
3640         2 exch put
3641     dup dup
3642         1 get
3643         pdf.baselineskip add
3644         1 exch put
3645     dup dup
3646         3 get
3647         pdf.baselineskip add
3648         3 exch put
3649     /pdf.currentrect exch def
3650     pdf.breaklink.write
3651 }
3652 1 index 3 get
3653 pdf.linkmargin 2 mul add
3654 pdf.outerbox pdf.rect.ht add
3655 2 index 1 get sub
3656 pdf.baselineskip div round cvi 1 sub
3657     exch
3658     repeat
3659     pdf.currentrect
3660     dup
3661         pdf.outerbox 0 get
3662         pdf.linkmargin sub
3663         0 exch put
3664     dup dup
3665         1 get
3666         pdf.baselineskip add
3667         1 exch put
3668     dup dup
3669         3 get
3670         pdf.baselineskip add
3671         3 exch put
3672     dup 2 index 2 get 2 exch put
3673     /pdf.currentrect exch def
3674     pdf.breaklink.write
3675     SDict /pdf.pdfmark.good false put
3676     exit
3677 }
3678 { pdf.count 2 sub /pdf.count exch def }
3679 ifelse
3680 }
3681 loop
3682 }
3683 if
3684 /ANN
3685 }
3686 def
3687 /pdf.breaklink.write
3688 {
3689     counttomark 1 sub
3690     index /_objdef eq
3691     {

```

```

3692         counttomark -2 roll
3693         dup wcheck
3694         {
3695             readonly
3696             counttomark 2 roll
3697         }
3698         { pop pop }
3699         ifelse
3700     }
3701     if
3702     counttomark 1 add copy
3703     pop pdf.currentrect
3704     /ANN pdfmark
3705 }
3706 def

```

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

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

```

3707 /pdf.pdfmark
3708 {
3709     SDict /pdf.pdfmark.good true put
3710     dup /ANN eq
3711     {
3712         pdf.pdfmark.store
3713         pdf.pdfmark.dict
3714         begin
3715             Subtype /Link eq
3716             currentdict /Rect known and
3717             SDict /pdf.outerbox known and
3718             SDict /pdf.baselineskip known and
3719             {
3720                 Rect 3 get
3721                 pdf.linkmargin 2 mul add
3722                 pdf.outerbox pdf.rect.ht add
3723                 Rect 1 get sub
3724                 pdf.baselineskip div round cvi 0 gt
3725                 { pdf.breaklink }
3726                 if
3727             }
3728             if
3729         end
3730         SDict /pdf.outerbox undef
3731         SDict /pdf.baselineskip undef
3732         currentdict /pdf.pdfmark.dict undef
3733     }
3734     if
3735     pdf.pdfmark.good
3736     { pdfmark }
3737     { cleartomark }

```



```

3738     ifelse
3739   }
3740   def
3741 /pdf.pdfmark.store
3742   {
3743     /pdf.pdfmark.dict 65534 dict def
3744     counttomark 1 add copy
3745     pop
3746     {
3747       dup mark eq
3748       {
3749         pop
3750         exit
3751       }
3752       {
3753         pdf.pdfmark.dict
3754         begin def end
3755       }
3756     } ifelse
3757   }
3758   loop
3759 }
3760 def

```

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

```
3761 </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>	1212, 1231, 1254, 1276, 1292, 1396, 1635, 1671, 2411, 2457
<code>\bool_gset_true:N</code>	1210, 1279, 1394, 1650, 2404, 2410
<code>\bool_if:NTF</code>	66, 578, 1222, 1226, 1242, 1245, 1249, 1260, 1267, 1271, 1283, 1287, 1407, 1412, 1417, 1609, 1654, 1793, 1843, 1983, 2025, 2399, 2414, 2419, 2424
<code>\bool_if:nTF</code>	2633, 2883, 3113
<code>\bool_lazy_and:nnTF</code>	791, 2142, 3237, 3313
<code>\bool_lazy_any:nTF</code>	1832
<code>\bool_lazy_or:nnTF</code>	2018
<code>\bool_new:N</code>	1213, 1280, 1397, 1651, 2384, 2385
<code>\bool_set_false:N</code>	1805, 1947, 2049, 2213
box commands:	
<code>\box_dp:N</code>	217, 219, 267, 269, 324, 326, 373, 375, 377, 379, 2436, 2469, 2470, 2495
<code>\box_ht:N</code>	219, 269, 326, 377, 379, 1856, 2090, 2441, 2480, 2481, 2497
<code>\box_if_empty:NTF</code>	2530
<code>\box_move_down:nn</code>	2358, 2436
<code>\box_move_up:nn</code>	2234, 2360, 2441
<code>\box_new:N</code>	2260, 2348, 2349
<code>\box_set_dp:Nn</code>	1734
<code>\box_set_ht:Nn</code>	1733
<code>\box_set_wd:Nn</code>	281, 1732
<code>\box_use:N</code>	224, 242, 256, 272, 299, 313, 329, 345, 357, 408, 422, 441, 1347, 1542, 1735, 2389
<code>\box_wd:N</code>	218, 226, 268, 274, 325, 331, 374, 376, 1855, 2089
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>	1300, 1427, 1678
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

<code>__color_backend_iccbased_-</code>	<code>__color_backend_separation_-</code>
<code>device:nnn</code> 914 , 914	<code>init_CIELAB:nnn</code> 576 , 688 , 758 , 805 , 830
<code>__color_backend_iccbased_-</code>	<code>__color_backend_separation_-</code>
<code>init:nnn</code> 753 , 753 , 896 , 896 , 1084 , 1085	<code>init_CIELAB:nnnnnn</code> 759
<code>__color_backend_init_resource:n</code>	<code>__color_backend_separation_-</code>
..... 788 , 788 , 817 , 888 , 912 , 927	<code>init_count:n</code> 576 , 635 , 638
<code>__color_backend_reset:</code>	<code>__color_backend_separation_-</code>
..... 488 , 503 , 511 , 523 , 527 , 532 , 953 , 954 , 987 , 988 , 1013 , 1082	<code>init_count:w</code> ... 576 , 639 , 640 , 644
<code>__color_backend_rgb:w</code> 1051	<code>__color_backend_separation_-</code>
<code>__color_backend_select:n</code>	<code>init_Device:Nn</code>
..... 488 , 489 , 491 , 493 , 495 , 496 , 527 , 527 , 529 , 530 , 531 , 573 576 , 620 , 622 , 624 , 625
<code>__color_backend_select:mn</code>	<code>\l__color_backend_stack_int</code>
..... 511 , 512 , 514 , 516 , 517 , 784 449 , 521 , 524 , 966 , 978
<code>__color_backend_select_cmyk:n</code> ..	<code>__color_backend_stroke:n</code>
..... 488 , 488 , 511 , 511 , 527 , 529 931 , 936 , 938 , 939 , 940 , 957 , 970 , 972 , 974 , 975 , 984
<code>__color_backend_select_devicen:nn</code>	<code>__color_backend_stroke_cmyk:n</code> ..
..... 572 , 574 , 756 , 757 , 778 , 786 931 , 938 , 957 , 969 , 991 , 1001 , 1027 , 1027
<code>__color_backend_select_gray:n</code> ..	<code>__color_backend_stroke_cmyk:w</code> ..
..... 488 , 490 , 511 , 513 , 527 , 530 , 537 1027 , 1029
<code>__color_backend_select_iccbased:nn</code>	<code>__color_backend_stroke_devicen:nn</code>
..... 575 , 575 , 760 , 760 , 778 , 787 941 , 952 , 981 , 986 , 1007 , 1012 , 1078 , 1081
<code>__color_backend_select_named:n</code> ..	<code>__color_backend_stroke_gray:n</code> ..
..... 488 , 492 , 534 , 534 931 , 939 , 957 , 971 , 991 , 1003 , 1027 , 1040
<code>__color_backend_select_rgb:n</code> ...	<code>__color_backend_stroke_gray_-</code>
..... 488 , 494 , 511 , 515 , 527 , 531	<code>aux:n</code> 1027 , 1044 , 1048
<code>__color_backend_select_separation:nn</code>	<code>__color_backend_stroke_reset:</code> ..
..... 572 , 572 , 574 , 756 , 756 , 757 , 778 , 779 , 783 , 786 , 787 953 , 954 , 987 , 988 , 1013 , 1014 , 1082 , 1083
<code>__color_backend_separation_-</code>	<code>__color_backend_stroke_rgb:n</code> ...
<code>init:n</code> 576 , 657 , 670 931 , 940 , 957 , 973 , 991 , 1005 , 1027 , 1050
<code>__color_backend_separation_-</code>	<code>__color_backend_stroke_rgb:w</code> ...
<code>init:nn</code> 805 , 815 , 819 1027 , 1052
<code>__color_backend_separation_-</code>	<code>__color_backend_stroke_separation:nn</code>
<code>init:nnn</code> 576 , 611 , 632 941 , 946 , 952 , 981 , 983 , 986 , 1007 , 1009 , 1012 , 1078 , 1079 , 1081
<code>__color_backend_separation_-</code>	<code>\l__color_backend_stroke_tl</code>
<code>init:nnnn</code> 576 , 634 , 646 507 , 520 , 967 , 977
<code>__color_backend_separation_-</code>	<code>\g__color_model_int</code> 583 , 592 , 740 , 768 , 817 , 823 , 824 , 878 , 879 , 888 , 912
<code>init:nnnnn</code> 576 , 576 , 597 , 690 , 758 , 758 , 805 , 805 , 845	<code>\c__color_model_range_CIELAB_tl</code> ..
<code>__color_backend_separation_-</code> 695 , 730 , 841 , 848
<code>init:nw</code> 576 , 661 , 672 , 686	<code>color.sc</code> 3380
<code>__color_backend_separation_-</code>	<code>cs commands:</code>
<code>init:w</code> 576 , 648 , 663 , 668	<code>\cs_generate_variant:Nn</code>
<code>__color_backend_separation_-</code> 62 , 65 , 98 , 147 , 152 , 163 , 194 , 200 , 597 , 1158 , 1357 , 1551 , 1997 , 2060 , 2080 , 2265 , 2280 , 2343 , 2834 , 2847 , 2957 , 2972 , 3002
<code>__color_backend_separation_-</code>	
<code>init_/DeviceCMYK:nnn</code> 576	
<code>__color_backend_separation_-</code>	
<code>init_/DeviceGray:nnn</code> 576	
<code>__color_backend_separation_-</code>	
<code>init_/DeviceRGB:nnn</code> 576	
<code>__color_backend_separation_-</code>	
<code>init_aux:nnnnnn</code> 576 , 582 , 598	

<code>\cs_gset:Npe</code> ..	2645, 2649, 3118, 3123	1421, 1434, 1439, 1441, 1443, 1445,
<code>\cs_gset_protected:Npn</code> ...	3317, 3318	1447, 1449, 1451, 1453, 1464, 1489,
<code>\cs_if_exist:NTF</code>		1501, 1513, 1525, 1532, 1554, 1560,
.....	27, 49, 1745, 2526, 2908, 2934	1565, 1570, 1581, 1591, 1601, 1603,
<code>\cs_if_exist_p:N</code>	792, 3238, 3314	1605, 1607, 1638, 1640, 1645, 1647,
<code>\cs_if_exist_use:NTF</code>	38, 610	1649, 1652, 1673, 1684, 1697, 1699,
<code>\cs_new:Npe</code>		1701, 1703, 1705, 1707, 1709, 1711,
.....	545, 2672, 2707, 2848, 2859, 2926, 3140	1713, 1721, 1743, 1762, 1785, 1802,
<code>\cs_new:Npn</code>	560, 619, 621,	1816, 1821, 1829, 1859, 1872, 1890,
.....	623, 625, 632, 638, 640, 646, 663,	1900, 1916, 1935, 1944, 1952, 1964,
.....	670, 672, 890, 1305, 1432, 1682,	1970, 1973, 1988, 1998, 2037, 2046,
.....	1858, 2093, 2251, 2272, 2344, 2346,	2052, 2058, 2061, 2068, 2081, 2086,
.....	2379, 2551, 2651, 2652, 2803, 2804,	2094, 2101, 2118, 2152, 2183, 2184,
.....	2816, 2835, 2836, 2939, 2965, 3003,	2186, 2188, 2190, 2196, 2202, 2210,
.....	3005, 3021, 3045, 3126, 3127, 3135,	2216, 2219, 2221, 2232, 2263, 2266,
.....	3147, 3148, 3153, 3154, 3159, 3160	2268, 2270, 2274, 2281, 2298, 2303,
<code>\cs_new_eq:NN</code>	46, 56, 58, 529, 530,	2308, 2313, 2323, 2328, 2336, 2351,
.....	531, 574, 757, 786, 787, 933, 934,	2356, 2388, 2390, 2395, 2397, 2402,
.....	935, 938, 939, 940, 951, 952, 953,	2417, 2422, 2459, 2488, 2507, 2516,
.....	954, 985, 986, 987, 988, 1011, 1012,	2553, 2560, 2586, 2591, 2619, 2631,
.....	1013, 1080, 1081, 1082, 1157, 1356,	2643, 2647, 2653, 2655, 2659, 2683,
.....	1362, 1363, 1550, 1552, 1553, 1559,	2685, 2687, 2698, 2718, 2728, 2751,
.....	1759, 1760, 1773, 1775, 1800, 1801,	2765, 2775, 2786, 2805, 2837, 2870,
.....	1864, 1865, 1866, 1889, 1914, 1931,	2881, 2887, 2915, 2949, 2951, 2958,
.....	1932, 1941, 1942, 1943, 1963, 1966,	2960, 2963, 2967, 2973, 2978, 2983,
.....	1967, 1968, 2033, 2043, 2044, 2045,	2985, 2987, 2995, 3008, 3024, 3026,
.....	2199, 2200, 2208, 2209, 2218, 2248,	3043, 3047, 3049, 3071, 3076, 3109,
.....	2249, 2250, 2254, 2273, 2389, 2966	3111, 3116, 3121, 3128, 3130, 3134,
<code>\cs_new_protected:Npe</code>		3136, 3137, 3138, 3139, 3141, 3142,
.....	576, 1063, 2898, 2955, 3028	3143, 3144, 3145, 3146, 3149, 3150,
<code>\cs_new_protected:Npn</code>	47, 53, 60, 63,	3151, 3152, 3155, 3156, 3157, 3158,
.....	71, 77, 82, 84, 88, 99, 109, 119, 128,	3161, 3162, 3165, 3184, 3191, 3197,
.....	137, 150, 153, 155, 157, 161, 166,	3202, 3209, 3216, 3252, 3268, 3278,
.....	175, 185, 195, 206, 228, 230, 245,	3284, 3290, 3322, 3324, 3326, 3328
.....	261, 276, 278, 304, 318, 333, 335,	<code>\cs_set_eq:NN</code>
.....	348, 362, 412, 425, 452, 466, 476,	2547, 2548
.....	488, 490, 492, 494, 496, 503, 511,	<code>\cs_set_protected:Npn</code>
.....	513, 515, 517, 523, 527, 532, 534,	2156
.....	572, 575, 598, 688, 734, 753, 756,	
.....	758, 759, 760, 779, 783, 788, 805,	
.....	819, 830, 852, 896, 914, 931, 936,	
.....	941, 946, 957, 959, 961, 963, 969,	
.....	971, 973, 975, 981, 983, 991, 993,	
.....	995, 997, 1001, 1003, 1005, 1007,	
.....	1009, 1014, 1017, 1019, 1021, 1023,	
.....	1027, 1029, 1040, 1048, 1050, 1052,	
.....	1078, 1079, 1083, 1084, 1085, 1159,	
.....	1165, 1170, 1172, 1174, 1182, 1190,	
.....	1199, 1209, 1211, 1214, 1216, 1233,	
.....	1238, 1256, 1278, 1281, 1294, 1307,	
.....	1312, 1314, 1316, 1318, 1320, 1322,	
.....	1324, 1326, 1331, 1358, 1360, 1364,	
.....	1369, 1374, 1384, 1393, 1395, 1398,	
.....	1400, 1402, 1404, 1409, 1414, 1419,	

D

dim commands:

<code>\dim_compare:nNnTF</code>	2132, 2137
<code>\dim_compare_p:nNn</code>	2143, 2144
<code>\dim_eval:n</code>	
.....	2354, 2589, 2667, 2668, 2669,
.....	2726, 2761, 2762, 2763, 3015, 3016,
.....	3017, 3048, 3074, 3173, 3174, 3177
<code>\dim_gset:Nn</code>	3186, 3187
<code>\dim_max:nn</code>	2467, 2478
<code>\dim_set:Nn</code>	
.....	1855, 1856, 2089, 2090, 2128, 2129
<code>\dim_set_eq:NN</code>	2194
<code>\dim_to_decimal:n</code> ..	373, 374, 375,
.....	376, 377, 379, 1563, 1568, 1574,
.....	1575, 1576, 1577, 1586, 1587, 1588,
.....	1679, 1698, 2241, 2242, 2465, 2476,

2494, 2495, 2496, 2497, 2501, 2557
 \dim_to_decimal_in_bp:n
 217, 218, 219, 267, 268, 269,
 324, 325, 326, 1178, 1179, 1186,
 1187, 1194, 1195, 1203, 1204, 1205,
 1302, 1306, 1310, 1367, 1372, 1378,
 1379, 1380, 1388, 1389, 1429, 1433,
 1437, 1683, 1767, 1768, 1769, 1770,
 1957, 1958, 1959, 1960, 2012, 2013,
 2014, 2015, 2226, 2227, 2228, 2229
 \dim_zero:N 2126, 2127
 \c_max_dim
 .. 2128, 2129, 2132, 2137, 2143, 2144
 draw internal commands:
 __draw_backend_add_to_path:n ...
 1560,
 1562, 1567, 1572, 1583, 1591, 1606
 __draw_backend_begin:
 .. 1159, 1159, 1358, 1358, 1554, 1554
 __draw_backend_box_use:Nnnnn
 .. 1331, 1331, 1532, 1532, 1721, 1721
 __draw_backend_cap_but:
 .. 1294, 1314, 1421, 1441, 1673, 1701
 __draw_backend_cap_rectangle: ..
 .. 1294, 1318, 1421, 1445, 1673, 1705
 __draw_backend_cap_round:
 .. 1294, 1316, 1421, 1443, 1673, 1703
 __draw_backend_clip:
 .. 1214, 1278, 1398, 1414, 1605, 1649
 __draw_backend_closepath:
 1214, 1214,
 1235, 1398, 1398, 1605, 1605, 1642
 __draw_backend_closestroke: ...
 .. 1214, 1233, 1398, 1402, 1605, 1640
 __draw_backend_cm:nnnn
 ... 1326, 1326, 1342, 1343, 1344,
 1453, 1453, 1536, 1713, 1713, 1724
 __draw_backend_cm_aux:nnnn
 1453, 1460, 1464
 __draw_backend_cm_decompose:nnnnN
 1459, 1488, 1489
 __draw_backend_cm_decompose_
 auxi:nnnnN 1488, 1493, 1501
 __draw_backend_cm_decompose_
 auxii:nnnnN 1488, 1505, 1513
 __draw_backend_cm_decompose_
 auxiii:nnnnN 1488, 1517, 1525
 __draw_backend_curveto:nnnnnn ..
 .. 1174, 1199, 1364, 1374, 1560, 1581
 __draw_backend_dash:n
 1294, 1300, 1305,
 1421, 1427, 1432, 1673, 1678, 1682
 __draw_backend_dash_aux:nn
 1673, 1677, 1684
 __draw_backend_dash_pattern:nn ..
 .. 1294, 1294, 1421, 1421, 1673, 1673
 __draw_backend_discardpath: ...
 .. 1214, 1281, 1398, 1419, 1605, 1652
 __draw_backend_end:
 .. 1159, 1165, 1358, 1360, 1554, 1559
 __draw_backend_evenodd_rule: ...
 .. 1209, 1209, 1393, 1393, 1601, 1601
 __draw_backend_fill:
 .. 1214, 1238, 1398, 1404, 1605, 1645
 __draw_backend_fillstroke:
 .. 1214, 1256, 1398, 1409, 1605, 1647
 __draw_backend_join_bevel:
 .. 1294, 1324, 1421, 1451, 1673, 1711
 __draw_backend_join_miter:
 .. 1294, 1320, 1421, 1447, 1673, 1707
 __draw_backend_join_round:
 .. 1294, 1322, 1421, 1449, 1673, 1709
 __draw_backend_lineto:nn
 .. 1174, 1182, 1364, 1369, 1560, 1565
 __draw_backend_linewidth:n
 .. 1294, 1307, 1421, 1434, 1673, 1697
 __draw_backend_literal:n
 1157, 1157, 1158, 1161,
 1162, 1163, 1167, 1168, 1171, 1173,
 1176, 1184, 1192, 1201, 1215, 1218,
 1219, 1220, 1221, 1224, 1230, 1240,
 1247, 1253, 1258, 1263, 1264, 1265,
 1266, 1269, 1275, 1285, 1291, 1296,
 1309, 1313, 1315, 1317, 1319, 1321,
 1323, 1325, 1328, 1333, 1334, 1335,
 1336, 1337, 1338, 1339, 1340, 1341,
 1345, 1346, 1348, 1349, 1350, 1351,
 1352, 1356, 1356, 1357, 1366, 1371,
 1376, 1386, 1399, 1401, 1403, 1406,
 1411, 1416, 1420, 1423, 1436, 1440,
 1442, 1444, 1446, 1448, 1450, 1452,
 1550, 1550, 1551, 1612, 1631, 1657
 __draw_backend_miterlimit:n ...
 .. 1294, 1312, 1421, 1439, 1673, 1699
 __draw_backend_moveto:nn
 .. 1174, 1174, 1364, 1364, 1560, 1560
 __draw_backend_nonzero_rule: ...
 .. 1209, 1211, 1393, 1395, 1601, 1603
 __draw_backend_path:n
 1605, 1607, 1639, 1646, 1648
 \g__draw_backend_path_int 1620, 1637
 \g__draw_backend_path_tl
 .. 1560, 1616, 1632, 1634, 1661, 1670
 __draw_backend_rectangle:nnnn ..
 .. 1174, 1190, 1364, 1384, 1560, 1570
 __draw_backend_scope_begin: 1170,
 1170, 1359, 1362, 1362, 1552, 1552

[__draw_backend_scope_end](#): [1170](#),
[1172](#), [1361](#), [1362](#), [1363](#), [1552](#), [1553](#)
[__draw_backend_stroke](#): [1214](#), [1216](#),
[1236](#), [1398](#), [1400](#), [1605](#), [1638](#), [1643](#)
[\g_draw_draw_clip_bool](#) .. [1214](#), [1605](#)
[\g_draw_draw_eor_bool](#)
... [1209](#), [1226](#), [1242](#), [1249](#), [1260](#),
[1271](#), [1287](#), [1393](#), [1407](#), [1412](#), [1417](#)
[\g_draw_draw_path_int](#) [1605](#)

E

[\errmessage](#) [38](#)
[\evensidemargin](#) [2434](#)
exp commands:
[\exp_after:wN](#) [2099](#)
[\exp_args:Ne](#) [580](#),
[634](#), [815](#), [1823](#), [1878](#), [1880](#), [1904](#),
[1906](#), [2310](#), [2325](#), [2430](#), [2588](#), [3073](#)
[\exp_args:Nf](#) [1299](#), [1426](#), [2353](#)
[\exp_args:Nne](#) [2998](#)
[\exp_args:NNf](#) [229](#), [277](#), [334](#)
[\exp_args:Nno](#) [3280](#)
[\exp_args:No](#) [3286](#)
[\exp_not:N](#) . [547](#), [553](#), [554](#), [555](#), [580](#),
[582](#), [583](#), [586](#), [587](#), [592](#), [2674](#), [2676](#),
[2679](#), [2709](#), [2711](#), [2714](#), [2850](#), [2852](#),
[2855](#), [2861](#), [2863](#), [2866](#), [2903](#), [2904](#),
[2910](#), [2911](#), [2930](#), [2935](#), [3030](#), [3035](#)
[\exp_not:n](#) [48](#), [96](#), [107](#), [145](#),
[904](#), [2301](#), [2306](#), [2582](#), [2820](#), [2821](#),
[2835](#), [2836](#), [2976](#), [2981](#), [2992](#), [3053](#)
[\ExplBackendFileDate](#) [1](#)

F

file commands:
[\file_compare_timestamp:nNnTF](#) . [1892](#)
[\file_parse_full_name:nNNN](#) [1874](#), [1902](#)
[\fmtversion](#) [51](#)
fp commands:
[\fp_compare:nNnTF](#)
. [236](#), [283](#), [289](#), [341](#), [1469](#), [1482](#), [1527](#)
[\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](#), [1471](#),
[1476](#), [1477](#), [1484](#), [1494](#), [1495](#), [1496](#),
[1497](#), [1506](#), [1507](#), [1508](#), [1509](#), [1518](#),
[1519](#), [1520](#), [1521](#), [2579](#), [2748](#), [3067](#)
[\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](#), [1469](#), [1482](#)

G

graphics commands:
[\l_graphics_search_ext_seq](#)
..... [1755](#), [1778](#), [1924](#), [2112](#)
graphics internal commands:
[\l_graphics_attr_tl](#) [1784](#),
[1789](#), [1806](#), [1818](#), [1825](#), [1827](#), [1862](#)
[__graphics_backend_dequote:w](#) ...
..... [1785](#), [1824](#), [1858](#)
[\l_graphics_backend_dir_str](#) . [1867](#)
[\l_graphics_backend_ext_str](#) . [1867](#)
[__graphics_backend_get_pagecount:n](#)
..... [1774](#), [1775](#), [1916](#), [1916](#),
[2031](#), [2033](#), [2101](#), [2101](#), [2253](#), [2254](#)
[__graphics_backend_getbb_auxi:n](#)
..... [1785](#), [1798](#), [1814](#), [1816](#)
[__graphics_backend_getbb_-](#)
auxi:nN ... [2037](#), [2041](#), [2050](#), [2052](#)
[__graphics_backend_getbb_-](#)
auxii:n [1785](#), [1819](#), [1821](#)
[__graphics_backend_getbb_-](#)
auxiii:nN .. [2037](#), [2055](#), [2058](#), [2060](#)
[__graphics_backend_getbb_-](#)
auxiii:n [1785](#), [1823](#), [1829](#)
[__graphics_backend_getbb_-](#)
auxiii:nNnn . [2037](#), [2056](#), [2059](#), [2061](#)
[__graphics_backend_getbb_-](#)
auxiv:nnNnn . [2037](#), [2064](#), [2068](#), [2080](#)
[__graphics_backend_getbb_-](#)
auxv:nNnn .. [2037](#), [2065](#), [2072](#), [2081](#)
[__graphics_backend_getbb_-](#)
auxvi:nNnn [2084](#), [2086](#)
[__graphics_backend_getbb_bmp:n](#) .
..... [1929](#), [1943](#), [2037](#), [2045](#)
[__graphics_backend_getbb_eps:n](#) .
..... [1757](#), [1759](#), [1867](#),
[1872](#), [1889](#), [1929](#), [1931](#), [2197](#), [2199](#)
[__graphics_backend_getbb_eps:nm](#)
..... [1867](#)
[__graphics_backend_getbb_eps:nn](#)
..... [1878](#), [1890](#)
[__graphics_backend_getbb_jpeg:n](#)
..... [1785](#), [1800](#),
[1929](#), [1941](#), [2037](#), [2043](#), [2202](#), [2208](#)
[__graphics_backend_getbb_jpg:n](#) .
[1785](#), [1785](#), [1800](#), [1801](#), [1929](#), [1935](#),
[1941](#), [1942](#), [1943](#), [2037](#), [2037](#), [2043](#),
[2044](#), [2045](#), [2202](#), [2202](#), [2208](#), [2209](#)
[__graphics_backend_getbb_-](#)
pagebox:w .. [2037](#), [2076](#), [2093](#), [2099](#)
[__graphics_backend_getbb_pdf:n](#) .
..... [1785](#), [1802](#), [1898](#),
[1929](#), [1944](#), [2037](#), [2046](#), [2210](#), [2210](#)

<code>__graphics_backend_getbb_png:n</code> .	1867 , 1914 , 1952 , 1963 , 2216 , 2218
.....	1785 , 1801 ,
1929 , 1942 , 2037 , 2044 , 2202 , 2209	
<code>__graphics_backend_getbb_ps:n</code> ..	<code>__graphics_backend_include_-</code>
.....	svg:n .. 2232 , 2232 , 2248 , 2249 , 2250
1867 , 1889 , 1929 , 1932 , 2197 , 2200	<code>__graphics_backend_loaded:n</code> ...
<code>__graphics_backend_getbb_svg:n</code> .	1743 , 1743 , 1755 , 1757 , 1774 , 1778 ,
.....	1924 , 1929 , 2032 , 2112 , 2197 , 2253
2118 , 2118	<code>\l__graphics_backend_name_str</code> . 1867
<code>__graphics_backend_getbb_svg_-</code>	<code>__graphics_bb_restore:nTF</code>
auxi:nNn 1818 , 2083 , 2120
2118 , 2134 , 2139 , 2152	<code>__graphics_bb_save:n</code> 1827 , 2091 , 2147
<code>__graphics_backend_getbb_svg_-</code>	<code>\l__graphics_decodearray_str</code> ...
auxii:w 1791 , 1792 ,
2118 , 2156 , 2178 , 2183	1804 , 1835 , 1841 , 1842 , 1946 , 1981 ,
<code>__graphics_backend_getbb_svg_-</code>	1982 , 2020 , 2023 , 2024 , 2048 , 2212
auxiii:Nw	<code>__graphics_extract_bb:n</code>
2118 , 2166 , 2184 1939 , 1948 , 2206 , 2214
<code>__graphics_backend_getbb_svg_-</code>	<code>\l__graphics_final_name_str</code> .. 1897
auxiv:Nw	<code>__graphics_get_pagecount:n</code>
2118 , 2169 , 2186 1775 , 2033 , 2254
<code>__graphics_backend_getbb_svg_-</code>	<code>\l__graphics_internal_box</code>
auxv:Nw 1853 , 1855 , 1856 , 2088 , 2089 , 2090
2118 , 2170 , 2188	<code>\l__graphics_internal_dim</code> 2193 , 2194
<code>__graphics_backend_getbb_svg_-</code>	<code>\l__graphics_internal_ior</code>
auxvi:Nn 2118 , 2185 , 2187 , 2189 , 2190 2122 , 2123 , 2130 , 2149
<code>__graphics_backend_getbb_svg_-</code>	<code>\l__graphics_interpolate_bool</code> ...
auxvii:w 1793 , 1805 , 1834 , 1843 ,
2118 , 2192 , 2196	1947 , 1983 , 2019 , 2025 , 2049 , 2213
<code>__graphics_backend_include:nn</code> ..	<code>\l__graphics_llx_dim</code>
..... 1767 , 1957 , 2012 , 2126 , 2226
2216 , 2217 , 2220 , 2221	<code>\l__graphics_lly_dim</code>
<code>__graphics_backend_include_-</code> 1768 , 1958 , 2013 , 2127 , 2227
auxi:nn ...	<code>\l__graphics_page_int</code>
1952 , 1965 , 1971 , 1973 1787 , 1809 , 1810 , 1848 ,
<code>__graphics_backend_include_-</code>	1849 , 1937 , 1979 , 1980 , 2006 , 2007 ,
auxii:nnn ..	2039 , 2054 , 2055 , 2097 , 2098 , 2204
1952 , 1975 , 1988 , 1997	<code>\l__graphics_pagebox_tl</code>
<code>__graphics_backend_include_-</code> 55 , 1788 , 1808 ,
auxiii:nnn	1850 , 1851 , 1938 , 1977 , 1978 , 2008 ,
1952 , 1995 , 1998	2010 , 2040 , 2063 , 2064 , 2099 , 2205
<code>__graphics_backend_include_-</code>	<code>\l__graphics_pdf_str</code>
bmp:n 1795 , 1796 , 1811 , 1812 , 1836 , 1845
1952 , 1968	<code>__graphics_read_bb:n</code>
<code>__graphics_backend_include_-</code>	.. 1759 , 1760 , 1931 , 1932 , 2199 , 2200
dequote:w	<code>\g__graphics_track_int</code>
2232 , 2243 , 2251 1951 , 2000 , 2001
<code>__graphics_backend_include_-</code>	<code>\l__graphics_urx_dim</code>
eps:n 1769 , 1855 , 1959 , 2014 , 2089 ,
1762 , 1773 , 1867 , 1900 , 1914 ,	2128 , 2132 , 2135 , 2143 , 2228 , 2241
1952 , 1952 , 1963 , 2216 , 2216 , 2218	<code>\l__graphics_ury_dim</code>
<code>__graphics_backend_include_-</code>	1770 , 1856 , 1960 , 2015 , 2090 , 2129 ,
jpeg:n . 1859 , 1864 , 1966 , 2232 , 2249	2137 , 2140 , 2144 , 2229 , 2234 , 2242
<code>__graphics_backend_include_-</code>	group commands:
jpg:n	<code>\group_begin:</code>
1859 , 1864 , 1865 , 1866 , 1952 , 172 , 191
1964 , 1966 , 1967 , 1968 , 2232 , 2250	<code>\group_end:</code>
<code>__graphics_backend_include_-</code> 180
jpsseg:n	
1952	
<code>__graphics_backend_include_-</code>	
pdf:n	
1859 , 1865 , 1904 ,	
1952 , 1970 , 2094 , 2094 , 2216 , 2219	
<code>__graphics_backend_include_-</code>	
png:n	
.. 1859 , 1866 , 1952 , 1967 , 2232 , 2248	
<code>__graphics_backend_include_ps:n</code>	
.....	
1762 , 1773 ,	

\group_insert_after:N ... 3266, 3310

H

hbox commands:
\hbox:n 2236, 2359, 2362,
2437, 2443, 2596, 2603, 3081, 3092
\hbox_overlap_right:n 224,
256, 272, 313, 329, 357, 441, 1347, 1542
\hbox_set:Nn .. 1853, 2088, 2429, 2461
\hbox_set:Nw 2412
\hbox_set_end: 2427
\hbox_unpack:N 2548
hook commands:
\hook_gput_code:nnn .. 54, 1745, 1747

I

int commands:
\int_compare:nNnTF
..... 1809, 1848, 1979, 2006,
2054, 2097, 2520, 2621, 2901, 2929
\int_const:Nn
..... 454, 1825, 1919, 2001, 2103
\int_eval:n 474, 484, 630, 639, 652,
654, 658, 671, 2645, 2649, 2879,
2904, 2911, 2924, 3110, 3118, 3123
\int_gincr:N 198,
364, 1611, 1656, 2000, 2271, 2338,
2369, 2446, 2964, 2997, 3010, 3030
\int_gset:Nn 173, 192, 2509, 2795
\int_gset_eq:NN 181, 2370, 2447, 3011
\int_if_exist:NTF 1990
\int_if_odd:nTF 2432
\int_max:nn 2105
\int_new:N 164, 165, 411, 449, 1637,
1951, 2350, 2381, 2383, 3007, 3023
\int_set_eq:NN 169, 188, 2521
\int_step_function:nnnN 656
\int_use:N 366, 397, 583,
592, 740, 768, 817, 823, 824, 878,
879, 888, 912, 1614, 1620, 1627,
1659, 1667, 1810, 1849, 1862, 1920,
1980, 1993, 2005, 2007, 2098, 2106,
2340, 2345, 2373, 2380, 2451, 2552,
2999, 3004, 3014, 3022, 3035, 3046
\int_value:w
..... 2674, 2709, 2850, 2861, 2879
\int_zero:N ... 1787, 1937, 2039, 2204
ior commands:
\ior_close:N 2149
\ior_if_eof:NTF 2123
\ior_map_break: 2145
\ior_open:Nn 2122
\ior_str_map_inline:Nn 2130

K

kernel internal commands:
__kernel_backend_align_begin: ..
..... 71, 71, 209, 233, 248
__kernel_backend_align_end: ...
..... 71, 77, 223, 241, 255
__kernel_backend_first_shipout:n
..... 49, 53, 56, 58, 68, 580, 3167
\g_kernel_backend_header_bool ..
..... 66, 578
__kernel_backend_literal:n . 46,
46, 47, 48, 61, 64, 69, 73, 80, 83,
85, 151, 154, 156, 158, 162, 338,
351, 498, 504, 528, 533, 600, 736,
780, 932, 937, 943, 948, 999, 1025,
1466, 1473, 1479, 1539, 1544, 1764,
1954, 1992, 2002, 2223, 2238, 2956,
3048, 3110, 3114, 3119, 3124, 3169
__kernel_backend_literal_page:n
..... 99, 99,
109, 153, 153, 2950, 2952, 3129, 3131
__kernel_backend_literal_pdf:n .
..... 88, 88, 98, 150, 150,
152, 264, 321, 1356, 3260, 3271, 3304
__kernel_backend_literal_-
postscript:n 60,
60, 62, 74, 75, 79, 210, 211, 213,
214, 222, 234, 249, 1157, 2623, 2635
__kernel_backend_literal_svg:n .
. 161, 161, 163, 168, 179, 187, 197,
365, 367, 384, 762, 1550, 1725, 1736
__kernel_backend_matrix:n
..... 137, 137, 147, 286, 307, 1456
__kernel_backend_postscript:n ..
..... 63, 63, 65,
500, 1002, 1004, 1006, 1010, 2264,
2315, 2330, 2359, 2365, 2405, 2437,
2444, 2448, 2462, 2490, 2534, 2541,
2547, 2555, 2562, 2596, 2603, 3218
__kernel_backend_scope:n
..... 166, 195, 200, 394,
399, 1065, 1557, 1602, 1604, 1624,
1664, 1686, 1698, 1700, 1702, 1704,
1706, 1708, 1710, 1712, 1715, 3329
__kernel_backend_scope_begin: ..
82, 82, 119, 119, 155, 155, 166, 166,
208, 232, 247, 263, 280, 306, 320,
337, 350, 1362, 1534, 1552, 1556, 1723
__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, 1363, 1546, 1553, 1559, 1737

\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, 1611, 1614, 1627, 1656, 1659, 1667

__kernel_color_backend_stack_-
init:Nnn 452, 452, 3242

__kernel_color_backend_stack_-
pop:n 466, 476, 524, 3275

__kernel_color_backend_stack_-
push:nn
466, 466, 521, 966, 978, 3263, 3307

__kernel_dependency_version_-
check:Nn 1

__kernel_dependency_version_-
check:nn 27, 29

__kernel_file_name_quote:n
1880, 1906

__kernel_kern:n
2364, 2366, 2595, 2599,
2602, 2606, 3080, 3088, 3091, 3107

L

lua commands:
\lua_load_module:n 1151

M

\MessageBreak 40

mode commands:
\mode_if_horizontal:TF ... 2511, 2518
\mode_if_math:TF 2409

msg commands:
\msg_error:nnn 538, 2124
\msg_new:nnn 540

O

\oddsidemargin 2433

opacity internal commands:
__opacity_backend:nn
3322, 3323, 3325, 3327, 3328

__opacity_backend:nnn
3197, 3199, 3200, 3204, 3211, 3216

__opacity_backend_fill:n
3197, 3202, 3278, 3278, 3322, 3324

__opacity_backend_fill_stroke:nn
3278, 3280, 3286, 3290, 3313, 3318

\l__opacity_backend_fill_tl
3248, 3254, 3287, 3295

__opacity_backend_reset:
3252, 3266, 3268, 3310

__opacity_backend_select:n
3197, 3197, 3252,
3252, 3293, 3313, 3317, 3322, 3322

\c__opacity_backend_stack_int ...
3237, 3263, 3275, 3307

__opacity_backend_stroke:n
3197, 3209, 3278, 3284, 3322, 3326

\l__opacity_backend_stroke_tl ...
3248, 3255, 3282, 3296

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 . 2955, 2955, 2957,
2959, 2961, 2975, 2980, 2989, 3012,
3031, 3044, 3051, 3083, 3084, 3094

__pdf_backend_annotation:nnnn ..
2351, 2351,
2659, 2659, 3008, 3008, 3134, 3134

__pdf_backend_annotation_-
aux:nnnn 2353, 2356

\g__pdf_backend_annotation_int ..
2350, 2370, 2380, 3007, 3011, 3022

__pdf_backend_annotation_last: .
2379, 2379,
2672, 2672, 3021, 3021, 3135, 3135

__pdf_backend_bdc:nn 2653, 2653,
2949, 2949, 3128, 3128, 3161, 3161

__pdf_backend_catalog_gput:nn ..
2266, 2266,
2765, 2765, 2958, 2958, 3144, 3144

__pdf_backend_compress_objects:n
2619, 2631,
2870, 2881, 3109, 3111, 3155, 3156

__pdf_backend_compresslevel:n ..
2619, 2619,
2870, 2870, 3109, 3109, 3155, 3155

\l__pdf_backend_content_box 2348,
2412, 2436, 2439, 2441, 2470, 2481

__pdf_backend_destination:nn ...
2560, 2560,
2728, 2728, 3049, 3049, 3142, 3142

_pdf_backend_destination:nnnn .	_pdf_backend_link_sf_restore: .
..... 2560, 2586, 2390, 2413, 2456, 2516
2728, 2751, 3049, 3071, 3142, 3143	_pdf_backend_link_sf_save: ...
_pdf_backend_destination_- 2390, 2408, 2426, 2507
aux:nnnn .. 2560, 2588, 2591, 3049, 3073, 3076	\l_pdf_backend_model_box . 2349,
_pdf_backend_emc: .. 2653, 2655,	2429, 2461, 2469, 2480, 2495, 2497
2949, 2951, 3128, 3130, 3161, 3162	_pdf_backend_objcompresslevel:n
_pdf_backend_info_gput:nn 2870, 2884, 2885, 2887
..... 2266, 2268,	_pdf_backend_object_id:n
2765, 2775, 2958, 2960, 3144, 3145 2270, 2273,
_pdf_backend_link:nw	2786, 2804, 2963, 2966, 3146, 3148
2390	\g_pdf_backend_object_int
_pdf_backend_link_aux:nw ... 2390	... 2271, 2338, 2340, 2345, 2369,
_pdf_backend_link_begin:n	2370, 2373, 2446, 2447, 2795, 2964,
..... 3024, 3025, 3027, 3028	2997, 2999, 3004, 3010, 3011, 3014
_pdf_backend_link_begin:nnw ..	_pdf_backend_object_last:
... 2683, 2684, 2686, 2687, 3136, 3138 2344, 2344,
_pdf_backend_link_begin:nw ...	2848, 2848, 3003, 3003, 3146, 3153
..... 2392, 2396, 2397	_pdf_backend_object_new:
_pdf_backend_link_begin_aux:nw 2270, 2270,
..... 2400, 2402	2786, 2786, 2963, 2963, 3146, 3146
_pdf_backend_link_begin_-	_pdf_backend_object_now:nn ...
goto:nnw	2336, 2336, 2343, 2837, 2837, 2847,
2683, 2683, 3024, 3024, 3136, 3136	2995, 2995, 3002, 3146, 3151, 3152
_pdf_backend_link_begin_-	\g_pdf_backend_object_prop
user:nnw 2785, 2962
2683, 2685, 3024, 3026, 3136, 3137	_pdf_backend_object_ref:n
\g_pdf_backend_link_bool	2270, 2272, 2273, 2277, 2786, 2803,
..... 2385, 2399, 2404, 2419, 2457	2963, 2965, 2966, 2970, 3146, 3147
\g_pdf_backend_link_dict_tl ...	_pdf_backend_object_write:nn ..
..... 2382, 2407, 2452 2805, 2814, 2816, 2845, 3146
_pdf_backend_link_end:	_pdf_backend_object_write:nnn .
..... 2390, 2417,	2274, 2274, 2280, 2805, 2805, 2834,
2683, 2698, 3024, 3043, 3136, 3139	2967, 2967, 2972, 3146, 3149, 3150
_pdf_backend_link_end_aux: ...	_pdf_backend_object_write_-
..... 2390, 2420, 2422	array:nn ... 2274, 2298, 2967, 2973
\g_pdf_backend_link_int	_pdf_backend_object_write_-
..... 2381, 2447,	aux:nnn 2274, 2276, 2281, 2339
2451, 2552, 3023, 3030, 3035, 3046	_pdf_backend_object_write_-
_pdf_backend_link_last:	dict:nn 2274, 2303, 2967, 2978
..... 2551, 2551,	_pdf_backend_object_write_-
2707, 2707, 3045, 3045, 3140, 3140	fstream:nn . 2274, 2308, 2967, 2983
_pdf_backend_link_margin:n ...	_pdf_backend_object_write_-
..... 2553, 2553,	fstream:nnn
2718, 2718, 3047, 3047, 3141, 3141	2311, 2313
\g_pdf_backend_link_math_bool ..	_pdf_backend_object_write_-
..... 2384, 2410, 2411, 2414, 2424	stream:nn .. 2274, 2323, 2967, 2985
_pdf_backend_link_minima:	_pdf_backend_object_write_-
..... 2390, 2428, 2459	stream:nnn
_pdf_backend_link_outerbox:n ..	2274, 2326, 2328
..... 2390, 2430, 2488	_pdf_backend_object_write_-
\g_pdf_backend_link_sf_int	stream:nnnn . 2967, 2984, 2986, 2987
..... 2383, 2509, 2520, 2521	_pdf_backend_pageobject_ref:n .
 2346, 2346,
	2859, 2859, 3005, 3005, 3146, 3154

_pdf_backend_pagesize_gset:nn . . .	3165, 3165, 3184, 3184, 3191, 3191	pdf.linkmargin	3391
_pdf_backend_pdfmark:n . . .	2263, 2263, 2265, 2267, 2269, 2283, 2300, 2305, 2371, 2563, 2607, 2654, 2656	pdf.llx	3394
_pdf_backend_version_major: . . .	2645, 2651, 2651, 2926, 2926, 3118, 3119, 3126, 3126, 3159, 3159	pdf.lly	3394
_pdf_backend_version_major_gset:n	2643, 2643, 2898, 2898, 3116, 3116, 3157, 3157	pdf.originx	3465
_pdf_backend_version_minor: . . .	2649, 2651, 2652, 2926, 2939, 3123, 3124, 3126, 3127, 3159, 3160	pdf.originy	3465
_pdf_backend_version_minor_gset:n	2643, 2647, 2898, 2915, 3116, 3121, 3157, 3158	pdf.outerbox	3707
\l_pdf_breaklink_pdfmark_tl	2386, 2454, 2546	pdf.pdfmark	3707
_pdf_breaklink_postscript:n	2388, 2388, 2438, 2440, 2547	pdf.pdfmark.dict	3707
_pdf_breaklink_usebox:N	2389, 2389, 2439, 2548	pdf.pdfmark.good	3707
_pdf_exp_not_i:nn	2805, 2824, 2829, 2835	pdf.pt.dvi	3387
_pdf_exp_not_ii:nn	2805, 2825, 2830, 2836	pdf.rect	3394
\l_pdf_internal_box	2260	pdf.rect.ht	3387
pdf.baselineskip	3707	pdf.rightboundary	3465
pdf.bordertracking	3465	pdf.save.linkll	3394
pdf.bordertracking.begin	3465	pdf.save.linkur	3394
pdf.bordertracking.continue	3465	pdf.save.ll	3394
pdf.bordertracking.end	3465	pdf.save.ur	3394
pdf.bordertracking.endpage	3465	pdf.tmpa	3430
pdf.breaklink	3603	pdf.tmpb	3430
pdf.breaklink.write	3603	pdf.tmpc	3430
pdf.brokenlink.dict	3465	pdf.tmpd	3430
pdf.brokenlink.rect	3465	pdf.urx	3394
pdf.brokenlink.skip	3465	pdf.ury	3394
pdf.count	3603	pdfmanagement commands:	
pdf.currentrect	3603	_pdfmanagement_add:nnn	797, 3245, 3256, 3297, 3300
pdf.cvs	3387	_pdfmanagement_if_active_p:	792, 793, 3238, 3239, 3314, 3315
pdf.dest.anchor	3430	peek commands:	
pdf.dest.point	3430	_peek_meaning:NNTF	2165, 2168
pdf.dest.x	3430	_peek_remove_spaces:n	2163
pdf.dest.y	3430	prg commands:	
pdf.dest2device	3430	_prg_replicate:nn	177, 628, 649, 659, 860
pdf.dev.x	3430	prop commands:	
pdf.dev.y	3430	_prop_gput:Nnn	586, 827
pdf.dvi.pt	3387	_prop_if_in:NnTF	563
pdf.globaldict	3384	_prop_item:Nn	566
pdf.leftboundary	3465	_prop_new:N	544, 2785, 2962
pdf.linkdp.pad	3391	\ProvidesExplFile	2
pdf.linkht.pad	3391		
		Q	
		quark commands:	
		_quark_if_recursion_tail_stop:n	562
		_q_recursion_stop	555
		_q_recursion_tail	554
		S	
		scan commands:	
		_scan_stop:	122, 131, 484, 2193, 2196, 2701, 2726, 2749, 2763, 2879, 2896, 2904, 2911, 2924

scan internal commands:	
\ <code>\s_color_stop</code>	639, 640, 644, 648, 661, 664, 668, 672, 686, 861, 890, 894, 1028, 1030, 1051, 1053
\ <code>\s_graphics_stop</code>	1824, 1858, 2158, 2173, 2180, 2184, 2186, 2188, 2243, 2251
separation	<u>3381</u>
seq commands:	
\ <code>\seq_set_from_clist:Nn</code>	1756, 1780, 1926, 2114
shipout commands:	
\ <code>\l_shipout_box</code>	2530, 2532, 2540
skip commands:	
\ <code>\skip_horizontal:n</code>	226, 274, 331
str commands:	
\ <code>\c_hash_str</code>	397, 1620, 1627, 1667
\ <code>\c_percent_str</code>	1071, 1072, 1073
\ <code>\str_case:nn</code>	866, 2287, 2818
\ <code>\str_case:nnTF</code>	2567, 2737, 3056
\ <code>\str_convert_pdfname:n</code>	587, 607, 816
\ <code>\str_if_empty:NTF</code>	1795, 1811
\ <code>\str_if_empty_p:N</code>	1836
\ <code>\str_if_eq:nnTF</code>	536, 766, 3292
\ <code>\str_new:N</code>	1869, 1870, 1871
\ <code>\str_tail:N</code>	1883, 1909
sys commands:	
\ <code>\sys_if_shell:TF</code>	1867
\ <code>\sys_shell_now:n</code>	1894
T	
\TeX and \LaTeX 2 ϵ commands:	
\ <code>\@ifl@t@r</code>	49, 51
\ <code>\@makecol@hook</code>	2526, 2528
\ <code>\special</code>	2
tex commands:	
\ <code>\tex_afterassignment:D</code>	2192
\ <code>\tex_baselineskip:D</code>	2501
\ <code>\tex_endinput:D</code>	44
\ <code>\tex_global:D</code>	2872, 2889, 2903, 2910, 2917
\ <code>\tex_immediate:D</code>	1831, 2808, 2811, 2840, 2843
\ <code>\tex_luatexversion:D</code>	2901, 2929
\ <code>\tex_pageheight:D</code>	3187
\ <code>\tex_pagewidth:D</code>	3186
\ <code>\tex_pdfannot:D</code>	2665
\ <code>\tex_pdfcatalog:D</code>	2771
\ <code>\tex_pdfcolorstack:D</code>	472, 482
\ <code>\tex_pdfcolorstackinit:D</code>	460
\ <code>\tex_pdfcompresslevel:D</code>	2877
\ <code>\tex_pdfdest:D</code>	2734, 2757
\ <code>\tex_pdfendlink:D</code>	2704
\ <code>\tex_pdfextension:D</code>	91, 102, 112, 122, 131, 140, 469, 479, 2662, 2690, 2701, 2731, 2754, 2768, 2778, 2789, 2808, 2840
\ <code>\tex_pdffeedback:D</code>	457, 2676, 2711, 2797, 2852, 2863
\ <code>\tex_pdfinfo:D</code>	2781
\ <code>\tex_pdflastannot:D</code>	2679
\ <code>\tex_pdflastlink:D</code>	2714
\ <code>\tex_pdflastobj:D</code>	2800, 2855
\ <code>\tex_pdflastximage:D</code>	1826, 1854
\ <code>\tex_pdflastximagepages:D</code>	1920
\ <code>\tex_pdflinkmargin:D</code>	2724
\ <code>\tex_pdfliteral:D</code>	94, 105, 115
\ <code>\tex_pdfmajorversion:D</code>	2908, 2910, 2934, 2935
\ <code>\tex_pdfminorversion:D</code>	2922, 2946
\ <code>\tex_pdfobj:D</code>	2792, 2811, 2843
\ <code>\tex_pdfobjcompresslevel:D</code>	2894
\ <code>\tex_pdfpageref:D</code>	2866
\ <code>\tex_pdfrefximage:D</code>	1854, 1861
\ <code>\tex_pdfrestore:D</code>	134
\ <code>\tex_pdfsave:D</code>	125
\ <code>\tex_pdfsetmatrix:D</code>	143
\ <code>\tex_pdfstartlink:D</code>	2693
\ <code>\tex_pdfvariable:D</code>	2721, 2874, 2891, 2903, 2919, 2930, 2943
\ <code>\tex_pdfximage:D</code>	1831, 1918
\ <code>\tex_spacefactor:D</code>	2512, 2521
\ <code>\tex_special:D</code>	46
\ <code>\tex_the:D</code>	1826, 2930, 2935, 2941
\ <code>\tex_vss:D</code>	2597, 2604, 3086, 3105
\ <code>\tex_XeTeXpdffile:D</code>	2050, 2096
\ <code>\tex_XeTeXpdfpagecount:D</code>	2106
\ <code>\tex_XeTeXpicfile:D</code>	2041
TeXcolorseparation	<u>3381</u>
<code>\textwidth</code>	2496
tl commands:	
\ <code>\c_space_tl</code>	288, 293, 296, 549, 554, 592, 695, 769, 979, 1596, 1766, 1767, 1768, 1769, 1956, 1957, 1958, 1959, 2007, 2010, 2012, 2013, 2014, 2015, 2076, 2098, 2225, 2226, 2227, 2228, 2452, 2681, 2716, 2857, 2868, 3014, 3036
\ <code>\tl_clear:N</code>	1788, 1804, 1938, 1946, 2040, 2048, 2205, 2212
\ <code>\tl_gclear:N</code>	1634, 1670
\ <code>\tl_gset:Nn</code>	1593, 2407
\ <code>\tl_if_blank:nTF</code>	462, 547, 643, 660, 667, 685, 811, 893, 2075, 2161
\ <code>\tl_if_empty:NTF</code>	1596, 1791, 1841, 1850, 1977, 1981, 2008, 2023, 2063
\ <code>\tl_if_empty:nTF</code>	905, 1690

