

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}{2024-05-08}{}
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2024-05-08}{}
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2024-05-08}{}
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2024-05-08}{}
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2024-05-08}{}
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2024-05-08}{}
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.

```

\_kernel_backend_scope_end:
\_kernel_backend_scope_begin:n
\_kernel_backend_scope_begin:e
\_kernel_backend_scope:n
\_kernel_backend_scope:e
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/XqTeX

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

526 `<*dvipdfmx | xetex>`

```

\__color_backend_select:n 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 < *dvipdfmx | luatex | pdftex | xetex | dvips >
```

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

```
\g_color_backend_colorant_prop
```

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

(End of definition for \g_color_backend_colorant_prop.)

```
\_color_backend_devicen_colorants:n
```

```
\_color_backend_devicen_colorants:w
```

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

```
546 {
```

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

```
548   {
```

```
549     \c_space_tl
```

```
550     << ~
```

```
551       /Colorants ~
```

```
552       << ~
```

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

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

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

```
556       >> ~
```

```
557     >>
```

```
558   }
```

```
559 }
```

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

```
561 {
```

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

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

```
564   {
```

```
565     #1 ~
```

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

```
567   }
```

```
568   \_color_backend_devicen_colorants:w
```

```
569 }
```

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

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

```
571 < *dvips >
```

```
\_color_backend_select_separation:nn
```

```
\_color_backend_select_devicen:nn
```

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

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

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

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

```
\_color_backend_select_iccbased:nn
```

No support.

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

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

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.

```
\_color_backend_separation_init:nmnn
\_color_backend_separation_init:neenn
\_color_backend_separation_init_aux:nmnnn
lor_backend_separation_init_DeviceCMYK:nnn
lor_backend_separation_init_DeviceGray:nnn
lor_backend_separation_init_DeviceRGB:nnn
\_color_backend_separation_init_Device:Nn
  \_color_backend_separation_init:nnn
\_color_backend_separation_init_count:n
\_color_backend_separation_init_count:w
  \_color_backend_separation_init:nmnn
  \_color_backend_separation_init:w
  \_color_backend_separation_init:n
  \_color_backend_separation_init:nw
\_color_backend_separation_init_CIELAB:nnn

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

`_color_backend_select_separation:nn`

`_color_backend_select_devicen:nn` 778 `<*dvipdfmx | 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 `</dvipdfmx | xetex>`

782 `<*luatex | pdftex>`

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

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

785 `</luatex | pdftex>`

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

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

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

`_color_backend_init_resource:n` Resource initiation comes up a few times. For `dvipdfmx/X3TEX`, 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/X3TEX`.

```

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 <*dvipdfmx | xetex>

```

```

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

```

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

```

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

```

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

```

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

```

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

955 `</dviptfm | xetex>`

956 `<*luatex | pdftex>`

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

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

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

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

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

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

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

989 `</luatex | pdftex>`

990 `<*dvips>`

```

\__color_backend_fill_cmyk:n Fill color here is the same as general color except we skip the stroke part.
\__color_backend_fill_gray:n 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.
\__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
1020 { \__color_backend_fill:n { gray ~ #1 } }
1021 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1022 { \__color_backend_fill:n { rgb ~ #1 } }
1023 \cs_new_protected:Npn \__color_backend_fill:n #1
1024 {
1025 \__kernel_backend_literal:n { color~push~ #1 }
1026 }

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

```

`_color_backend_stroke_cmyk:n` For drawings in SVG, we use scopes for all stroke colors. The backend provides the necessary conversion for CMYK but only if that is set as the main color: a little bit of gymnastics as a result.

```

1027 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1028 {
1029   \_color_backend_fill_cmyk:n {#1}
1030   \_kernel_backend_scope:n { stroke = "{?color}" }
1031   \_color_backend_reset:
1032 }
1033 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1034 {
1035   \use:e
1036   {
1037     \_color_backend_stroke_gray_aux:n
1038     { \fp_eval:n { 100 * (#1) } }
1039   }
1040 }
1041 \cs_new_protected:Npn \_color_backend_stroke_gray_aux:n #1
1042 { \_color_backend:nnn {#1} {#1} {#1} }
1043 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1044 { \_color_backend_rgb:w #1 \s_color_stop }
1045 \cs_new_protected:Npn \_color_backend_stroke_rgb:w
1046 #1 ~ #2 ~ #3 \s_color_stop
1047 {
1048   \use:e
1049   {
1050     \_color_backend:nnn
1051     { \fp_eval:n { 100 * (#1) } }
1052     { \fp_eval:n { 100 * (#2) } }
1053     { \fp_eval:n { 100 * (#3) } }
1054   }
1055 }
1056 \cs_new_protected:Npe \_color_backend:nnn #1#2#3
1057 {
1058   \_kernel_backend_scope:n
1059   {
1060     stroke =
1061     "
1062     rgb
1063     (
1064       #1 \c_percent_str ,
1065       #2 \c_percent_str ,
1066       #3 \c_percent_str
1067     )
1068     "
1069   }
1070 }

```

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

`_color_backend_fill_separation:mn` At present, these are no-ops.

```

1071 \cs_new_protected:Npn \_color_backend_fill_separation:mn #1#2 { }
1072 \cs_new_protected:Npn \_color_backend_stroke_separation:mn #1#2 { }
1073 \cs_new_eq:NN \_color_backend_fill_devicen:mn \_color_backend_fill_separation:mn

```

```

1074 \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:
1075 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
1076 \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
1077 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3 { }
1078 \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.)
1079 </dvisvgm>
1080 </package>

```

3.5 Font handling integration

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

```

1081 <*lua>
1082 local l = lpeg
1083 local spaces = 1.P' '^0
1084 local digit16 = 1.R('09', 'af', 'AF')
1085
1086 local octet = digit16 * digit16 / function(s)
1087   return string.format('%.3g ', tonumber(s, 16) / 255)
1088 end
1089
1090 if luaotfload and luaotfload.set_transparent_colorstack then
1091   local htmlcolor = 1.Cs(octet * octet * octet * -1 * 1.Cc'rg')
1092   local color_export = {
1093     token.create'tex_endlocalcontrol:D',
1094     token.create'tex_hpack:D',
1095     token.new(0, 1),
1096     token.create'color_export:nnN',
1097     token.new(0, 1),
1098     ', ',
1099     token.new(0, 2),
1100     token.new(0, 1),
1101     'backend',
1102     token.new(0, 2),
1103     token.create'l_tmpa_tl',
1104     token.create'exp_after:wN',
1105     token.create'__color_select:nn',
1106     token.create'l_tmpa_tl',
1107     token.new(0, 2),
1108   }
1109   local group_end = token.create'group_end:'
1110   local value = (1 - 1.P'')^0
1111   luatexbase.add_to_callback('luaotfload.parse_color', function (value)

```

```

1112 % Also allow HTML colors to preserve compatibility
1113     local html = htmlcolor:match(value)
1114     if html then return html end
1115
1116 % If no l3color named color with this name is known, check for defined xcolor colors
1117     local l3color_prop = token.get_macro(string.format('l__color_named_%s_prop', value))
1118     if l3color_prop == nil or l3color_prop == '' then
1119         local legacy_color_macro = token.create(string.format('\\color@%s', value))
1120         if legacy_color_macro.cmdname ~= 'undefined_cs' then
1121             token.put_next(legacy_color_macro)
1122             return token.scan_argument()
1123         end
1124     end
1125
1126     tex.runtoks(function()
1127         token.get_next()
1128         color_export[6] = value
1129         tex.sprint(-2, color_export)
1130     end)
1131     local list = token.scan_list()
1132     if not list.head or list.head.next
1133         or list.head.subtype ~= node.subtype'pdf_colorstack' then
1134         error'Unexpected backend behavior'
1135     end
1136     local cmd = list.head.data
1137     node.free(list)
1138     return cmd
1139 end, 'l3color')
1140 end
1141 </lua>
1142 <*luatex>
1143 <*package>
1144 \lua_load_module:n {l3backend-luatex}
1145 </package>
1146 </luatex>

```

4 l3backend-draw implementation

```

1147 <*package>
1148 <@@=draw>

```

4.1 dvips backend

```

1149 <*dvips>

```

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

```

\__draw_backend_literal:e 1150 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_postscript:n
1151 \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

`__draw_backend_end:`

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.)

```

1152 \cs_new_protected:Npn \__draw_backend_begin:
1153 {
1154   \__draw_backend_literal:n { [begin] }
1155   \__draw_backend_literal:n { /draw.x~currentpoint~/draw.y~exch~def~def }
1156   \__draw_backend_literal:n { @beginspecial }
1157 }
1158 \cs_new_protected:Npn \__draw_backend_end:
1159 {
1160   \__draw_backend_literal:n { @endspecial }
1161   \__draw_backend_literal:n { [end] }
1162 }

```

(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:

```

1163 \cs_new_protected:Npn \__draw_backend_scope_begin:
1164 { \__draw_backend_literal:n { save } }
1165 \cs_new_protected:Npn \__draw_backend_scope_end:
1166 { \__draw_backend_literal:n { restore } }

```

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

__draw_backend_moveto:nn
 __draw_backend_lineto:nn
 __draw_backend_rectangle:nnnn
 __draw_backend_curveto:nnnnnn

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.

```

1167 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1168 {
1169   \__draw_backend_literal:e
1170   {
1171     \dim_to_decimal_in_bp:n {#1} ~
1172     \dim_to_decimal_in_bp:n {#2} ~ moveto
1173   }
1174 }
1175 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1176 {
1177   \__draw_backend_literal:e
1178   {
1179     \dim_to_decimal_in_bp:n {#1} ~
1180     \dim_to_decimal_in_bp:n {#2} ~ lineto
1181   }
1182 }
1183 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1184 {
1185   \__draw_backend_literal:e
1186   {
1187     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1188     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~

```



```

1189         moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1190     }
1191 }
1192 \cs_new_protected:Npn \__draw_backend_curveto:nnnnn #1#2#3#4#5#6
1193 {
1194     \__draw_backend_literal:e
1195     {
1196         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1197         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1198         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1199         curveto
1200     }
1201 }

```

(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: 1202 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
\g__draw_draw_eor_bool        1203 { \bool_gset_true:N \g__draw_draw_eor_bool }
                               1204 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
                               1205 { \bool_gset_false:N \g__draw_draw_eor_bool }
                               1206 \bool_new:N \g__draw_draw_eor_bool

```

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

```

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

```

```

1207 \cs_new_protected:Npn \__draw_backend_closepath:
1208 { \__draw_backend_literal:n { closepath } }
1209 \cs_new_protected:Npn \__draw_backend_stroke:
1210 {
1211     \__draw_backend_literal:n { gsave }
1212     \__draw_backend_literal:n { color.sc }
1213     \__draw_backend_literal:n { stroke }
1214     \__draw_backend_literal:n { grestore }
1215     \bool_if:NT \g__draw_draw_clip_bool
1216     {
1217         \__draw_backend_literal:e
1218         {
1219             \bool_if:NT \g__draw_draw_eor_bool { eo }
1220             clip
1221         }
1222     }
1223     \__draw_backend_literal:n { newpath }
1224     \bool_gset_false:N \g__draw_draw_clip_bool
1225 }
1226 \cs_new_protected:Npn \__draw_backend_closestroke:
1227 {
1228     \__draw_backend_closepath:

```

```

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

```

```

1283     }
1284     \__draw_backend_literal:n { newpath }
1285     \bool_gset_false:N \g__draw_draw_clip_bool
1286   }

```

(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:
1287 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1288 {
1289   \__draw_backend_literal:e
1290   {
1291     [
1292       \exp_args:Nf \use:n
1293       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1294     ] ~
1295     \dim_to_decimal_in_bp:n {#2} ~ setdash
1296   }
1297 }
1298 \cs_new:Npn \__draw_backend_dash:n #1
1299 { ~ \dim_to_decimal_in_bp:n {#1} }
1300 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1301 {
1302   \__draw_backend_literal:e
1303   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1304 }
1305 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1306 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1307 \cs_new_protected:Npn \__draw_backend_cap_but:
1308 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1309 \cs_new_protected:Npn \__draw_backend_cap_round:
1310 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1311 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1312 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1313 \cs_new_protected:Npn \__draw_backend_join_miter:
1314 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1315 \cs_new_protected:Npn \__draw_backend_join_round:
1316 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1317 \cs_new_protected:Npn \__draw_backend_join_bevel:
1318 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

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

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

```

1319 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1320 {
1321   \__draw_backend_literal:n
1322   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1323 }

```

(End of definition for __draw_backend_cm:nnnn.)

`_draw_backend_box_use:Nnnnn`

Inside a picture `@beginspecial/@endspecial` are active, which is normally a good thing but means that the position and scaling would be off if the box was inserted directly. To deal with that, there are a number of possible approaches. 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`.

```
1324 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
1325 {
1326   \_draw_backend_literal:n { save }
1327   \_draw_backend_literal:n { 72~Resolution~div~72~VResolution~div~neg~scale }
1328   \_draw_backend_literal:n { magscale { 1~DVImag~div~dup~scale } if }
1329   \_draw_backend_literal:n { draw.x~neg~draw.y~neg~translate }
1330   \_draw_backend_literal:n { [end] }
1331   \_draw_backend_literal:n { [begin] }
1332   \_draw_backend_literal:n { save }
1333   \_draw_backend_literal:n { currentpoint }
1334   \_draw_backend_literal:n { currentpoint~translate }
1335   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1336   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1337   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1338   \_draw_backend_literal:n { neg~exch~neg~exch~translate }
1339   \_draw_backend_literal:n { [end] }
1340   \hbox_overlap_right:n { \box_use:N #1 }
1341   \_draw_backend_literal:n { [begin] }
1342   \_draw_backend_literal:n { restore }
1343   \_draw_backend_literal:n { [end] }
1344   \_draw_backend_literal:n { [begin] }
1345   \_draw_backend_literal:n { restore }
1346 }

(End of definition for \_draw_backend_box_use:Nnnnn.)

1347 </dvips>
```

4.2 Lua_T_EX, pdf_T_EX, dvipdfmx and X_T_EX

Lua_T_EX, pdf_T_EX, dvipdfmx and X_T_EX directly produce PDF output and understand a shared set of specials for drawing commands.

```
1348 <*dvipdfmx | luatex | pdftex | xetex>
```

4.2.1 Drawing

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

`_draw_backend_literal:e`

```
1349 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_pdf:n
1350 \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: 1351 \cs_new_protected:Npn \_draw_backend_begin:
1352 { \_draw_backend_scope_begin: }
1353 \cs_new_protected:Npn \_draw_backend_end:
1354 { \_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: 1355 \cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:
1356 \cs_new_eq:NN \_draw_backend_scope_end: \_kernel_backend_scope_end:
```

(End of definition for `_draw_backend_scope_begin:` and `_draw_backend_scope_end:.`)

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

`_draw_backend_lineto:nn`
`_draw_backend_curveto:nnnnnn`
`_draw_backend_rectangle:nnnn`

```
1357 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1358 {
1359   \_draw_backend_literal:e
1360   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1361 }
1362 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1363 {
1364   \_draw_backend_literal:e
1365   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1366 }
1367 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1368 {
1369   \_draw_backend_literal:e
1370   {
1371     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1372     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1373     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1374     c
1375   }
1376 }
1377 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1378 {
1379   \_draw_backend_literal:e
1380   {
1381     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1382     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1383     re
1384   }
1385 }
```

(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: 1386 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
\g__draw_draw_eor_bool 1387 { \bool_gset_true:N \g__draw_draw_eor_bool }
1388 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1389 { \bool_gset_false:N \g__draw_draw_eor_bool }
1390 \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:` 1391 `\cs_new_protected:Npn _draw_backend_closepath:`
`_draw_backend_closestroke:` 1392 `{ _draw_backend_literal:n { h } }`
`_draw_backend_fill:` 1393 `\cs_new_protected:Npn _draw_backend_stroke:`
`_draw_backend_fillstroke:` 1394 `{ _draw_backend_literal:n { S } }`
`_draw_backend_clip:` 1395 `\cs_new_protected:Npn _draw_backend_closestroke:`
`_draw_backend_discardpath:` 1396 `{ _draw_backend_literal:n { s } }`
1397 `\cs_new_protected:Npn _draw_backend_fill:`
1398 `{`
1399 `_draw_backend_literal:e`
1400 `{ f \bool_if:NT \g__draw_draw_eor_bool * }`
1401 `}`
1402 `\cs_new_protected:Npn _draw_backend_fillstroke:`
1403 `{`
1404 `_draw_backend_literal:e`
1405 `{ B \bool_if:NT \g__draw_draw_eor_bool * }`
1406 `}`
1407 `\cs_new_protected:Npn _draw_backend_clip:`
1408 `{`
1409 `_draw_backend_literal:e`
1410 `{ W \bool_if:NT \g__draw_draw_eor_bool * }`
1411 `}`
1412 `\cs_new_protected:Npn _draw_backend_discardpath:`
1413 `{ _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` 1414 `\cs_new_protected:Npn _draw_backend_dash_pattern:nn #1#2`
`_draw_backend_linewidth:n` 1415 `{`
`_draw_backend_miterlimit:n` 1416 `_draw_backend_literal:e`
`_draw_backend_cap_butt:` 1417 `{`
`_draw_backend_cap_round:` 1418 `[`
`_draw_backend_cap_rectangle:` 1419 `\exp_args:Nf \use:n`
`_draw_backend_join_miter:` 1420 `{ \clist_map_function:nN {#1} _draw_backend_dash:n }`
`_draw_backend_join_round:` 1421 `] ~`
`_draw_backend_join_bevel:` 1422 `\dim_to_decimal_in_bp:n {#2} ~ d`
1423 `}`
1424 `}`
1425 `\cs_new:Npn _draw_backend_dash:n #1`
1426 `{ ~ \dim_to_decimal_in_bp:n {#1} }`
1427 `\cs_new_protected:Npn _draw_backend_linewidth:n #1`
1428 `{`
1429 `_draw_backend_literal:e`
1430 `{ \dim_to_decimal_in_bp:n {#1} ~ w }`
1431 `}`
1432 `\cs_new_protected:Npn _draw_backend_miterlimit:n #1`
1433 `{ _draw_backend_literal:e { #1 ~ M } }`
1434 `\cs_new_protected:Npn _draw_backend_cap_butt:`
1435 `{ _draw_backend_literal:n { 0 ~ J } }`
1436 `\cs_new_protected:Npn _draw_backend_cap_round:`

```

1437 { \_draw_backend_literal:n { 1 ~ J } }
1438 \cs_new_protected:Npn \_draw_backend_cap_rectangle:
1439 { \_draw_backend_literal:n { 2 ~ J } }
1440 \cs_new_protected:Npn \_draw_backend_join_miter:
1441 { \_draw_backend_literal:n { 0 ~ j } }
1442 \cs_new_protected:Npn \_draw_backend_join_round:
1443 { \_draw_backend_literal:n { 1 ~ j } }
1444 \cs_new_protected:Npn \_draw_backend_join_bevel:
1445 { \_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 Lua \TeX /pdf \TeX and dvipdfmx/X \TeX . In the former, we have a direct method to maintain alignment: the backend can use a matrix itself. For dvipdfmx/X \TeX , we can to decompose the matrix into rotations and a scaling, then use those operations as they are handled by the backend. (There is backend support for matrix operations in dvipdfmx/X \TeX , 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!

```

1446 \cs_new_protected:Npn \_draw_backend_cm:nnnn #1#2#3#4
1447 {
1448 <*luatex | pdftex>
1449   \_kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1450 </luatex | pdftex>
1451 <*dvipdfmx | xetex>
1452   \_draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1453   \_draw_backend_cm_aux:nnnn
1454 </dvipdfmx | xetex>
1455 }
1456 <*dvipdfmx | xetex>
1457 \cs_new_protected:Npn \_draw_backend_cm_aux:nnnn #1#2#3#4
1458 {
1459   \_kernel_backend_literal:e
1460   {
1461     x:rotate~
1462     \fp_compare:nNnTF {#1} = \c_zero_fp
1463       { 0 }
1464       { \fp_eval:n { round ( -#1 , 5 ) } }
1465   }
1466   \_kernel_backend_literal:e
1467   {
1468     x:scale~
1469     \fp_eval:n { round ( #2 , 5 ) } ~
1470     \fp_eval:n { round ( #3 , 5 ) }
1471   }
1472   \_kernel_backend_literal:e
1473   {
1474     x:rotate~
1475     \fp_compare:nNnTF {#4} = \c_zero_fp
1476       { 0 }
1477       { \fp_eval:n { round ( -#4 , 5 ) } }
1478   }
1479 }
1480 </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.

```

1481 <*dvipdfmx | xetex>
1482 \cs_new_protected:Npn \_draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1483 {
1484   \use:e
1485   {
1486     \_draw_backend_cm_decompose_auxi:nnnnN
1487     { \fp_eval:n { (#1 + #4) / 2 } }
1488     { \fp_eval:n { (#1 - #4) / 2 } }
1489     { \fp_eval:n { (#3 + #2) / 2 } }
1490     { \fp_eval:n { (#3 - #2) / 2 } }
1491   }
1492   #5
1493 }
1494 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1495 {
1496   \use:e
1497   {
1498     \_draw_backend_cm_decompose_auxii:nnnnN
1499     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1500     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1501     { \fp_eval:n { atand ( #3 , #2 ) } }
1502     { \fp_eval:n { atand ( #4 , #1 ) } }
1503   }
1504   #5

```



```

1505 }
1506 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1507 {
1508   \use:e
1509   {
1510     \__draw_backend_cm_decompose_auxiii:nnnnN
1511     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1512     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1513     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1514     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1515   }
1516   #5
1517 }
1518 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1519 {
1520   \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1521     { #5 {#1} {#2} {#3} {#4} }
1522     { #5 {#1} {#3} {#2} {#4} }
1523 }
1524 </dviptdpmx | 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.

```

1525 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1526 {
1527   \__kernel_backend_scope_begin:
1528   < *luatex | pdftex >
1529   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1530   < /luatex | pdftex >
1531   < *dviptdpmx | xetex >
1532   \__kernel_backend_literal:n
1533   { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1534   < /dviptdpmx | xetex >
1535   \hbox_overlap_right:n { \box_use:N #1 }
1536   < *dviptdpmx | xetex >
1537   \__kernel_backend_literal:n { pdf:etrans }
1538   < /dviptdpmx | xetex >
1539   \__kernel_backend_scope_end:
1540 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1541 < /dviptdpmx | luatex | pdftex | xetex >

```

4.3 dvisvgm backend

```

1542 < *dvisvgm >

```

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

__draw_backend_literal:e

```

1543 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1544 \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: 1545 \cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:  
1546 \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 coordinate system is translated. That is done inside a scope, which as described below

```
1547 \cs_new_protected:Npn \_draw_backend_begin:  
1548   {  
1549     \_kernel_backend_scope_begin:  
1550     \_kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }  
1551   }  
1552 \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 values as they are given, the entire path needs to be collected up before being output in one go. For that we use a dedicated storage routine, which adds spaces as required. `_draw_backend_lineto:nn` Since paths should be fully expanded there is no need to worry about the internal x-type expansion. `_draw_backend_rectangle:nmmn` `_draw_backend_curveto:nmmmmn` `_draw_backend_add_to_path:n`

```
\g__draw_backend_path_tl 1553 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2  
1554   {  
1555     \_draw_backend_add_to_path:n  
1556     { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }  
1557   }  
1558 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2  
1559   {  
1560     \_draw_backend_add_to_path:n  
1561     { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }  
1562   }  
1563 \cs_new_protected:Npn \_draw_backend_rectangle:nmmn #1#2#3#4  
1564   {  
1565     \_draw_backend_add_to_path:n  
1566     {  
1567       M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}  
1568       h ~ \dim_to_decimal:n {#3} ~  
1569       v ~ \dim_to_decimal:n {#4} ~  
1570       h ~ \dim_to_decimal:n { -#3 } ~  
1571       Z  
1572     }  
1573   }  
1574 \cs_new_protected:Npn \_draw_backend_curveto:nmmmmn #1#2#3#4#5#6  
1575   {  
1576     \_draw_backend_add_to_path:n  
1577     {  
1578       C ~  
1579       \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~  
1580       \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~  
1581       \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}  
1582     }  
1583   }
```

```

1583 }
1584 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1585 {
1586   \tl_gset:Ne \g__draw_backend_path_tl
1587   {
1588     \g__draw_backend_path_tl
1589     \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1590     #1
1591   }
1592 }
1593 \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: 1594 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1595   { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1596 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1597   { \__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_closepath: 1598 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_stroke: 1599   { \__draw_backend_add_to_path:n { Z } }
\__draw_backend_closestroke: 1600 \cs_new_protected:Npn \__draw_backend_path:n #1
\__draw_backend_fill: 1601   {
\__draw_backend_fillstroke: 1602   \bool_if:NTF \g__draw_draw_clip_bool
\__draw_backend_clip: 1603     {
\__draw_backend_discardpath: 1604       \int_gincr:N \g__kernel_clip_path_int
\g__draw_draw_clip_bool 1605       \__draw_backend_literal:e
\g__draw_draw_path_int 1606         {
1607           < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1608             { ?nl }
1609           <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1610           < /clipPath > { ? nl }
1611           <
1612             use~xlink:href =
1613               "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1614               #1
1615             />
1616           }
1617         \__kernel_backend_scope:e
1618         {
1619           clip-path =
1620             "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1621         }
1622       }
1623     }
1624   \__draw_backend_literal:e

```

```

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

```

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

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_dash_aux:nn
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1666 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1667 {
1668     \use:e
1669     {
1670         \__draw_backend_dash_aux:nn
1671         { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1672         { \dim_to_decimal:n {#2} }
1673     }

```

```

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

```

1706 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1707 {
1708   \__kernel_backend_scope:n
1709   {
1710     transform =
1711     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1712   }
1713 }

```

(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.

```

1714 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1715 {
1716   \__kernel_backend_scope_begin:
1717   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}

```

```

1718   \__kernel_backend_literal_svg:n
1719   {
1720     < g~
1721       stroke="none"~
1722       transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1723     >
1724   }
1725   \box_set_wd:Nn #1 { Opt }
1726   \box_set_ht:Nn #1 { Opt }
1727   \box_set_dp:Nn #1 { Opt }
1728   \box_use:N #1
1729   \__kernel_backend_literal_svg:n { </g> }
1730   \__kernel_backend_scope_end:
1731 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```
1732 </dvisvgm>
```

```
1733 </package>
```

5 l3backend-graphics implementation

```

1734 <*package>
1735 <@@=graphics>

```

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

```

1736 \cs_new_protected:Npn \__graphics_backend_loaded:n #1
1737 {
1738   \cs_if_exist:NTF \hook_gput_code:nnn
1739   {
1740     \hook_gput_code:nnn
1741     { package / l3graphics / after }
1742     { backend }
1743     {#1}
1744   }
1745   {#1}
1746 }

```

(End of definition for __graphics_backend_loaded:n.)

5.1 dvips backend

```
1747 <*dvips>
```

\l_graphics_search_ext_seq

```

1748 \__graphics_backend_loaded:n
1749 { \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

```

1750 \__graphics_backend_loaded:n
1751 {
1752   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1753   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
1754 }

```

(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
1755 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1756 {
1757   \__kernel_backend_literal:e
1758   {
1759     PSfile = #1 \c_space_tl
1760     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1761     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1762     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1763     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1764   }
1765 }
1766 \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`

```
1767 \__graphics_backend_loaded:n
1768 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }
```

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

```
1769 </dvips>
```

5.2 LuaTeX and pdfTeX backends

```
1770 < *luatex | pdftex >
```

`\l_graphics_search_ext_seq`

```
1771 \__graphics_backend_loaded:n
1772 {
1773   \seq_set_from_clist:Nn
1774   \l_graphics_search_ext_seq
1775   { .pdf , .eps , .ps , .png , .jpg , .jpeg }
1776 }
```

(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.

```
1777 \tl_new:N \l__graphics_attr_tl
```

(End of definition for `\l__graphics_attr_tl`.)

`__graphics_backend_getbb_jpg:n`

`__graphics_backend_getbb_jpeg:n`

`__graphics_backend_getbb_pdf:n`

`__graphics_backend_getbb_png:n`

`__graphics_backend_getbb_auxi:n`

`__graphics_backend_getbb_auxii:n`

`__graphics_backend_getbb_auxiii:n`

`__graphics_backend_dequote:w`

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

```
1778 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
```

```

1779 {
1780   \int_zero:N \l__graphics_page_int
1781   \tl_clear:N \l__graphics_pagebox_tl
1782   \tl_set:Ne \l__graphics_attr_tl
1783   {
1784     \tl_if_empty:NF \l__graphics_decodearray_str
1785     { :D \l__graphics_decodearray_str }
1786     \bool_if:NT \l__graphics_interpolate_bool
1787     { :I }
1788     \str_if_empty:NF \l__graphics_pdf_str
1789     { :X \l__graphics_pdf_str }
1790   }
1791   \__graphics_backend_getbb_auxi:n {#1}
1792 }
1793 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1794 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1795 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1796 {
1797   \tl_clear:N \l__graphics_decodearray_str
1798   \bool_set_false:N \l__graphics_interpolate_bool
1799   \tl_set:Ne \l__graphics_attr_tl
1800   {
1801     : \l__graphics_pagebox_tl
1802     \int_compare:nNnT \l__graphics_page_int > 1
1803     { :P \int_use:N \l__graphics_page_int }
1804     \str_if_empty:NF \l__graphics_pdf_str
1805     { :X \l__graphics_pdf_str }
1806   }
1807   \__graphics_backend_getbb_auxi:n {#1}
1808 }
1809 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1810 {
1811   \__graphics_bb_restore:eF { #1 \l__graphics_attr_tl }
1812   { \__graphics_backend_getbb_auxiii:n {#1} }
1813 }

```

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.

```

1814 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1815 {
1816   \exp_args:Ne \__graphics_backend_getbb_auxiiii:n
1817   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1818   \int_const:cn { c__graphics_ #1 \l__graphics_attr_tl _int }
1819   { \tex_the:D \tex_pdflastximage:D }
1820   \__graphics_bb_save:e { #1 \l__graphics_attr_tl }
1821 }
1822 \cs_new_protected:Npn \__graphics_backend_getbb_auxiiii:n #1
1823 {
1824   \tex_immediate:D \tex_pdfximage:D
1825   \bool_lazy_any:nT
1826   {
1827     { \l__graphics_interpolate_bool }

```



```

1828     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1829     { ! \str_if_empty_p:N \l__graphics_pdf_str }
1830   }
1831   {
1832     attr ~
1833     {
1834       \tl_if_empty:NF \l__graphics_decodearray_str
1835       { /Decode~[ \l__graphics_decodearray_str ] }
1836       \bool_if:NT \l__graphics_interpolate_bool
1837       { /Interpolate~true }
1838       \l__graphics_pdf_str
1839     }
1840   }
1841   \int_compare:nNnT \l__graphics_page_int > 0
1842   { page ~ \int_use:N \l__graphics_page_int }
1843   \tl_if_empty:NF \l__graphics_pagebox_tl
1844   { \l__graphics_pagebox_tl }
1845   {#1}
1846   \hbox_set:Nn \l__graphics_internal_box
1847   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1848   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1849   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1850 }
1851 \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.

```

1852 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1853 {
1854   \tex_pdfrefximage:D
1855   \int_use:c { c__graphics_ #1 \l__graphics_attr_tl_int }
1856 }
1857 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1858 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1859 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

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

__graphics_backend_getbb_eps:n
 __graphics_backend_getbb_ps:n
 __graphics_backend_getbb_eps:nm
 __graphics_backend_include_eps:n
 __graphics_backend_include_ps:n
 \l__graphics_backend_dir_str
 \l__graphics_backend_name_str
 \l__graphics_backend_ext_str

EPS graphics may be included in LuaTeX/pdfTeX by conversion to PDF: this requires restricted shell escape. Modelled on the epstopdf L^AT_EX₂ ϵ package, but simplified, conversion takes place here if we have shell access.

```

1860 \sys_if_shell:T
1861 {
1862   \str_new:N \l__graphics_backend_dir_str
1863   \str_new:N \l__graphics_backend_name_str
1864   \str_new:N \l__graphics_backend_ext_str
1865   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1866   {
1867     \file_parse_full_name:nNNN {#1}
1868     \l__graphics_backend_dir_str
1869     \l__graphics_backend_name_str

```

```

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

```

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

__graphics_backend_get_pagecount:n Simply load and store.

```

1909 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1910 {
1911     \tex_pdfximage:D {#1}
1912     \int_const:cn { c__graphics_ #1 _pages_int }
1913     { \int_use:N \tex_pdflastximagepages:D }
1914 }

```

(End of definition for __graphics_backend_get_pagecount:n.)

```

1915 </luatex | pdftex>

```

5.3 dvipdfmx backend

1916 `*dvipdfmx | xetex`

`\l_graphics_search_ext_seq`

```
1917 \__graphics_backend_loaded:n
1918 {
1919   \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1920   { .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }
1921 }
```

(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`

```
1922 \__graphics_backend_loaded:n
```

`__graphics_backend_getbb_jpg:n`

```
1923 {
```

`__graphics_backend_getbb_jpeg:n`

```
1924   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
```

`__graphics_backend_getbb_pdf:n`

```
1925   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
```

`__graphics_backend_getbb_png:n`

```
1926 }
```

`__graphics_backend_getbb_bmp:n`

```
1927 \*dvipdfmx
```

```
1928 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
```

```
1929 {
```

```
1930   \int_zero:N \l__graphics_page_int
```

```
1931   \tl_clear:N \l__graphics_pagebox_tl
```

```
1932   \__graphics_extract_bb:n {#1}
```

```
1933 }
```

```
1934 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
```

```
1935 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
```

```
1936 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
```

```
1937 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
```

```
1938 {
```

```
1939   \tl_clear:N \l__graphics_decodearray_str
```

```
1940   \bool_set_false:N \l__graphics_interpolate_bool
```

```
1941   \__graphics_extract_bb:n {#1}
```

```
1942 }
```

```
1943 \end{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.

```
1944 \int_new:N \g__graphics_track_int
```

(End of definition for `\g__graphics_track_int`.)

`__graphics_backend_include_eps:n`

The special syntax depends on the file type. There is a difference in how PDF graphics are best handled between dvipdfmx and Xe_{La}TeX: for the latter it is better to use the primitive route. The relevant code for that is included later in this file.

`__graphics_backend_include_ps:n`

`__graphics_backend_include_jpg:n`

```
1945 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
```

`__graphics_backend_include_jpeg:n`

```
1946 {
```

`__graphics_backend_include_pdf:n`

```
1947   \__kernel_backend_literal:e
```

`__graphics_backend_include_png:n`

```
1948   {
```

`__graphics_backend_include_bmp:n`

```
1949     PSfile = #1 \c_space_tl
```

`__graphics_backend_include_auxi:n`

```
1950     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
```

`__graphics_backend_include_auxii:nn`

```
1951     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
```

`__graphics_backend_include_auxiii:enn`

```
1952     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
```

`__graphics_backend_include_auxiiii:nnn`

```

1953     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1954   }
1955 }
1956 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1957 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1958   { \__graphics_backend_include_auxi:nn {#1} { image } }
1959 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1960 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
1961 \cs_new_eq:NN \__graphics_backend_include_bmp:n \__graphics_backend_include_jpg:n
1962 <*dviPDFmx>
1963 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1964   { \__graphics_backend_include_auxi:nn {#1} { epdf } }
1965 </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.

```

1966 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1967   {
1968     \__graphics_backend_include_auxii:enn
1969     {
1970       \tl_if_empty:NF \l__graphics_pagebox_tl
1971       { : \l__graphics_pagebox_tl }
1972       \int_compare:nNnT \l__graphics_page_int > 1
1973       { :P \int_use:N \l__graphics_page_int }
1974       \tl_if_empty:NF \l__graphics_decodearray_str
1975       { :D \l__graphics_decodearray_str }
1976       \bool_if:NT \l__graphics_interpolate_bool
1977       { :I }
1978     }
1979     {#1} {#2}
1980   }
1981 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1982   {
1983     \int_if_exist:cTF { c__graphics_ #2#1 _int }
1984     {
1985       \__kernel_backend_literal:e
1986       { pdf:usexobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
1987     }
1988     { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1989   }
1990 \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!

```

1991 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1992   {
1993     \int_gincr:N \g__graphics_track_int
1994     \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
1995     \__kernel_backend_literal:e
1996     {
1997       pdf:#3~

```

```

1998     @graphic \int_use:c { c__graphics_ #1#2 _int } ~
1999     \int_compare:nNnT \l__graphics_page_int > 1
2000     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2001     \tl_if_empty:NF \l__graphics_pagebox_tl
2002     {
2003         pagebox ~ \l__graphics_pagebox_tl \c_space_tl
2004         bbox ~
2005             \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2006             \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2007             \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2008             \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
2009     }
2010     (#1)
2011     \bool_lazy_or:nnT
2012     { \l__graphics_interpolate_bool }
2013     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
2014     {
2015         <<
2016         \tl_if_empty:NF \l__graphics_decodearray_str
2017         { /Decode~[ \l__graphics_decodearray_str ] }
2018         \bool_if:NT \l__graphics_interpolate_bool
2019         { /Interpolate~true }
2020         >>
2021     }
2022 }
2023 }

```

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

`__graphics_backend_get_pagecount:n`

```

2024 < *dvipdfmx >
2025 \__graphics_backend_loaded:n
2026 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }
2027 < /dvipdfmx >

```

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

```

2028 < /dvipdfmx | xetex >

```

5.4 X_YTeX backend

```

2029 < *xetex >

```

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

```

\__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:VnNn
\__graphics_backend_getbb_auxv:nnNn
\__graphics_backend_getbb_auxvi:nnNn
\__graphics_backend_getbb_pagebox:w

```

```

2030 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2031 {
2032     \int_zero:N \l__graphics_page_int
2033     \tl_clear:N \l__graphics_pagebox_tl
2034     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2035 }
2036 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2037 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n

```

```

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

```

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

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

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

```

2087 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2088 {
2089   \tex_XeTeXpdffile:D #1 ~
2090   \int_compare:nNnT \l__graphics_page_int > 0
2091     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2092     \exp_after:wN \__graphics_backend_getbb_pagebox:w \l__graphics_pagebox_tl
2093 }

```

(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.

```

2094 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
2095 {
2096   \int_const:cn { c__graphics_#1_pages_int }
2097   {
2098     \int_max:nn
2099     { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2100     { 1 }
2101   }
2102 }

```

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

```
2103 </xetex>
```

5.5 dvisvgm backend

```
2104 <*dvisvgm>
```

`\l_graphics_search_ext_seq`

```

2105 \__graphics_backend_loaded:n
2106 {
2107   \seq_set_from_clist:Nn
2108   \l_graphics_search_ext_seq
2109   { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }
2110 }

```

(End of definition for `\l_graphics_search_ext_seq`.)

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

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

```

2111 \cs_new_protected:Npn \__graphics_backend_getbb_svg:n #1
2112 {
2113   \__graphics_bb_restore:nF {#1}
2114   {
2115     \ior_open:Nn \l__graphics_internal_ior {#1}
2116     \ior_if_eof:NTF \l__graphics_internal_ior
2117     { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2118     {
2119       \dim_zero:N \l__graphics_llx_dim
2120       \dim_zero:N \l__graphics_lly_dim

```

```

2121 \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2122 \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2123 \ior_str_map_inline:Nn \l__graphics_internal_ior
2124 {
2125   \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2126   {
2127     \__graphics_backend_getbb_svg_auxi:nNn
2128     { width } \l__graphics_urx_dim {##1}
2129   }
2130   \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2131   {
2132     \__graphics_backend_getbb_svg_auxi:nNn
2133     { height } \l__graphics_ury_dim {##1}
2134   }
2135   \bool_lazy_and:nnF
2136   { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2137   { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2138   { \ior_map_break: }
2139 }
2140 \__graphics_bb_save:n {#1}
2141 }
2142 \ior_close:N \l__graphics_internal_ior
2143 }
2144 }
2145 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2146 {
2147   \use:e
2148   {
2149     \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxii:w
2150     ##1 \tl_to_str:n {#1} = ##2 \tl_to_str:n {#1} = ##3
2151     \s__graphics_stop
2152   }
2153   {
2154     \tl_if_blank:nF {##2}
2155     {
2156       \peek_remove_spaces:n
2157       {
2158         \peek_meaning:NTF ' % '
2159         { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2160         {
2161           \peek_meaning:NTF " % "
2162           { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2163           { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2164         }
2165       }
2166       ##2 \s__graphics_stop
2167     }
2168   }
2169   \use:e
2170   {
2171     \__graphics_backend_getbb_svg_auxii:w #3
2172     \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2173     \s__graphics_stop
2174   }

```



```

2175 }
2176 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2177 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop
2178 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2179 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2180 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2181 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2182 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2183 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2184 {
2185   \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2186   \l__graphics_internal_dim #2 bp \scan_stop:
2187   \dim_set_eq:NN #1 \l__graphics_internal_dim
2188 }
2189 \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` Simply use the generic function.

```

\__graphics_backend_getbb_ps:n
2190 \__graphics_backend_loaded:n
2191 {
2192   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2193   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
2194 }

```

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

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

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_jpeg:n
2195 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2196 {
2197   \int_zero:N \l__graphics_page_int
2198   \tl_clear:N \l__graphics_pagebox_tl
2199   \__graphics_extract_bb:n {#1}
2200 }
2201 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2202 \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

```

2203 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2204 {
2205   \tl_clear:N \l__graphics_decodearray_str
2206   \bool_set_false:N \l__graphics_interpolate_bool
2207   \__graphics_extract_bb:n {#1}
2208 }

```

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

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

```

\__graphics_backend_include_ps:n
\__graphics_backend_include_pdf:n
\__graphics_backend_include:nn
2209 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2210 { \__graphics_backend_include:nn { PSfile } {#1} }
2211 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n

```

```

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

```

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

```

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

```

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

```

2225 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2226 {
2227   \box_move_up:nn { \l__graphics_ury_dim }
2228   {
2229     \hbox:n
2230     {
2231       \__kernel_backend_literal:e
2232       {
2233         dvisvgm:img~
2234         \dim_to_decimal:n { \l__graphics_urx_dim } ~
2235         \dim_to_decimal:n { \l__graphics_ury_dim } ~
2236         \__graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2237       }
2238     }
2239   }
2240 }
2241 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_svg:n
2242 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_svg:n
2243 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_svg:n
2244 \cs_new:Npn \__graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2245 {#2}

```

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

```

\__graphics_backend_get_pagecount:n

```

```

2246 \__graphics_backend_loaded:n
2247 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }

```

(End of definition for __graphics_backend_get_pagecount:n.)

```

2248 </dvisvgm>
2249 </package>

```

6 I3backend-pdf implementation

```
2250 ⟨*package⟩
2251 ⟨@@=pdf⟩
```

Setting up PDF resources is a complex area with only limited documentation in the engine manuals. The following code builds heavily on existing ideas from `hyperref` work by Sebastian Rahtz and Heiko Oberdiek, and significant contributions by Alexander Grahn, in addition to the specific code referenced a various points.

6.1 Shared code

A very small number of items that belong at the backend level but which are common to most backends.

```
2252 ⟨*!dvisvgm⟩

\l__pdf_internal_box
2253 \box_new:N \l__pdf_internal_box
(End of definition for \l__pdf_internal_box.)
2254 ⟨!/dvisvgm⟩
```

6.2 dvips backend

```
2255 ⟨*dvips⟩

\__pdf_backend_pdfmark:n Used often enough it should be a separate function.
\__pdf_backend_pdfmark:e
2256 \cs_new_protected:Npn \__pdf_backend_pdfmark:n #1
2257 { \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2258 \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
2259 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2260 { \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2261 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2262 { \__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
\__pdf_backend_object_id:n
2263 \cs_new_protected:Npn \__pdf_backend_object_new:
2264 { \int_gincr:N \g__pdf_backend_object_int }
2265 \cs_new:Npn \__pdf_backend_object_ref:n #1 { { pdf.obj #1 } }
2266 \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: some work to get things right. To allow code
`_pdf_backend_object_write:nne` sharing with the anonymous version, we use an auxiliary.
`_pdf_backend_object_write_aux:nnn` 2267 `\cs_new_protected:Npn _pdf_backend_object_write:nnn #1#2#3`
`_pdf_backend_object_write_array:nn` 2268 `{`
`_pdf_backend_object_write_dict:nn` 2269 `_pdf_backend_object_write_aux:nnn`
`_pdf_backend_object_write_fstream:nn` 2270 `{ _pdf_backend_object_ref:n {#1} }`
`_pdf_backend_object_write_stream:nn` 2271 `{#2} {#3}`
`_pdf_backend_object_write_stream:nnn` 2272 `}`
2273 `\cs_generate_variant:Nn _pdf_backend_object_write:nnn { nne }`
2274 `\cs_new_protected:Npn _pdf_backend_object_write_aux:nnn #1#2#3`
2275 `{`
2276 `_pdf_backend_pdfmark:e`
2277 `{`
2278 `/_objdef ~ #1`
2279 `/type`
2280 `\str_case:nn {#2}`
2281 `{`
2282 `{ array } { /array }`
2283 `{ dict } { /dict }`
2284 `{ fstream } { /stream }`
2285 `{ stream } { /stream }`
2286 `}`
2287 `/OBJ`
2288 `}`
2289 `\use:c { _pdf_backend_object_write_ #2 :nn } {#1} {#3}`
2290 `}`
2291 `\cs_new_protected:Npn _pdf_backend_object_write_array:nn #1#2`
2292 `{`
2293 `_pdf_backend_pdfmark:e`
2294 `{ #1 ~0~ [~ \exp_not:n {#2} ~] ~ /PUTINTERVAL }`
2295 `}`
2296 `\cs_new_protected:Npn _pdf_backend_object_write_dict:nn #1#2`
2297 `{`
2298 `_pdf_backend_pdfmark:e`
2299 `{ #1 << \exp_not:n {#2} >> /PUT }`
2300 `}`
2301 `\cs_new_protected:Npn _pdf_backend_object_write_fstream:nn #1#2`
2302 `{`
2303 `\exp_args:Ne`
2304 `_pdf_backend_object_write_fstream:nnn {#1} #2`
2305 `}`
2306 `\cs_new_protected:Npn _pdf_backend_object_write_fstream:nnn #1#2#3`
2307 `{`
2308 `_kernel_backend_postscript:n`
2309 `{`
2310 `SDict ~ begin ~`
2311 `mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~`
2312 `mark ~ #1 ~ (#3)~ (r)~ file ~ /PUT ~ pdfmark ~`
2313 `end`
2314 `}`
2315 `}`
2316 `\cs_new_protected:Npn _pdf_backend_object_write_stream:nn #1#2`
2317 `{`
2318 `\exp_args:Ne`

```

2319     \__pdf_backend_object_write_stream:nnn {#1} #2
2320   }
2321 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2322   {
2323     \__kernel_backend_postscript:n
2324     {
2325       mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2326       mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2327     }
2328   }

```

(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
2329 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2330   {
2331     \int_gincr:N \g__pdf_backend_object_int
2332     \__pdf_backend_object_write_aux:nnn
2333     { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2334     {#1} {#2}
2335   }
2336 \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.

```

2337 \cs_new:Npn \__pdf_backend_object_last:
2338   { { 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.

```

2339 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2340   { { 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.

```

2341 \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.

```

2342 \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.

```

2343 \int_new:N \g__pdf_backend_annotation_int

```

(End of definition for \g__pdf_backend_annotation_int.)

`_pdf_backend_annotation:nmmn`

Annotations are objects, but we track them separately. Notably, they are not in the object data lists. Here, to get the coordinates 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.

```
2344 \cs_new_protected:Npn \_pdf_backend_annotation:nmmn #1#2#3#4
2345 {
2346   \exp_args:Nf \_pdf_backend_annotation_aux:nmmn
2347   { \dim_eval:n {#1} } {#2} {#3} {#4}
2348 }
2349 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nmmn #1#2#3#4
2350 {
2351   \box_move_down:nn {#3}
2352   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2353   \box_move_up:nn {#2}
2354   {
2355     \hbox:n
2356     {
2357       \_kernel_kern:n {#1}
2358       \_kernel_backend_postscript:n { pdf.save.ur }
2359       \_kernel_kern:n { -#1 }
2360     }
2361   }
2362   \int_gincr:N \g__pdf_backend_object_int
2363   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2364   \_pdf_backend_pdfmark:e
2365   {
2366     /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2367     pdf.rect
2368     #4 ~
2369     /ANN
2370   }
2371 }
```

(End of definition for _pdf_backend_annotation:nmmn.)

`_pdf_backend_annotation_last:`

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

```
2372 \cs_new:Npn \_pdf_backend_annotation_last:
2373 { { 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.

```
2374 \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.

```
2375 \tl_new:N \g__pdf_backend_link_dict_tl
```

(End of definition for \g__pdf_backend_link_dict_tl.)

`\g__pdf_backend_link_sf_int`

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

```
2376 \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.
2377 `\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.
2378 `\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.
2379 `\tl_new:N \l_pdf_breaklink_pdfmark_tl`
2380 `\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.
2381 `\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.
2382 `\cs_new_eq:MN _pdf_breaklink_usebox:N \box_use:N`
(End of definition for `_pdf_breaklink_usebox:N`.)

`_pdf_backend_link_begin_goto:nmw` Links are created like annotations but with dedicated code to allow for adjusting the size
`_pdf_backend_link_begin_user:nmw` 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.

```
2383 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nmw #1#2
2384 {
2385   \_pdf_backend_link_begin:nw
2386   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2387 }
2388 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nmw #1#2
2389 { \_pdf_backend_link_begin:nw {#1#2} }
2390 \cs_new_protected:Npn \_pdf_backend_link_begin:nw #1
2391 {
```

```

2392     \bool_if:NF \g__pdf_backend_link_bool
2393     { \__pdf_backend_link_begin_aux:nw {#1} }
2394 }

```

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.

```

2395 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2396 {
2397     \bool_gset_true:N \g__pdf_backend_link_bool
2398     \__kernel_backend_postscript:n
2399     { /pdf.link.dict ( #1 ) def }
2400     \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2401     \__pdf_backend_link_sf_save:
2402     \mode_if_math:TF
2403     { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2404     { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2405     \hbox_set:Nw \l__pdf_backend_content_box
2406     \__pdf_backend_link_sf_restore:
2407     \bool_if:NT \g__pdf_backend_link_math_bool
2408     { \c_math_toggle_token }
2409 }
2410 \cs_new_protected:Npn \__pdf_backend_link_end:
2411 {
2412     \bool_if:NT \g__pdf_backend_link_bool
2413     { \__pdf_backend_link_end_aux: }
2414 }
2415 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2416 {
2417     \bool_if:NT \g__pdf_backend_link_math_bool
2418     { \c_math_toggle_token }
2419     \__pdf_backend_link_sf_save:
2420     \hbox_set_end:
2421     \__pdf_backend_link_minima:
2422     \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2423     \exp_args:Ne \__pdf_backend_link_outerbox:n
2424     {
2425         \int_if_odd:nTF { \value { page } }
2426         { \oddsidemargin }
2427         { \evensidemargin }
2428     }
2429     \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2430     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2431     \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2432     \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2433     \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2434     \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2435     {
2436         \hbox:n
2437         { \__kernel_backend_postscript:n { pdf.save.linkur } }
2438     }
2439     \int_gincr:N \g__pdf_backend_object_int
2440     \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2441     \__kernel_backend_postscript:e
2442     {

```



```

2443     mark
2444     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2445     \g__pdf_backend_link_dict_tl \c_space_tl
2446     pdf.rect
2447     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2448   }
2449   \__pdf_backend_link_sf_restore:
2450   \bool_gset_false:N \g__pdf_backend_link_bool
2451 }
2452 \cs_new_protected:Npn \__pdf_backend_link_minima:
2453 {
2454   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2455   \__kernel_backend_postscript:e
2456   {
2457     /pdf.linkdp.pad ~
2458     \dim_to_decimal:n
2459     {
2460       \dim_max:nn
2461       {
2462         \box_dp:N \l__pdf_backend_model_box
2463         - \box_dp:N \l__pdf_backend_content_box
2464       }
2465       { Opt }
2466     } ~
2467     pdf.pt.dvi ~ def
2468   /pdf.linkht.pad ~
2469   \dim_to_decimal:n
2470   {
2471     \dim_max:nn
2472     {
2473       \box_ht:N \l__pdf_backend_model_box
2474       - \box_ht:N \l__pdf_backend_content_box
2475     }
2476     { Opt }
2477   } ~
2478   pdf.pt.dvi ~ def
2479 }
2480 }
2481 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2482 {
2483   \__kernel_backend_postscript:e
2484   {
2485     /pdf.outerbox
2486     [
2487       \dim_to_decimal:n {#1} ~
2488       \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2489       \dim_to_decimal:n { #1 + \textwidth } ~
2490       \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2491     ]
2492     [ exch { pdf.pt.dvi } forall ] def
2493   /pdf.baselineskip ~
2494   \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2495   { pdf.pt.dvi ~ def }
2496   { pop ~ pop }

```

```

2497         ifelse
2498     }
2499 }
2500 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2501 {
2502     \int_gset:Nn \g__pdf_backend_link_sf_int
2503     {
2504         \mode_if_horizontal:TF
2505         { \tex_spacefactor:D }
2506         { 0 }
2507     }
2508 }
2509 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2510 {
2511     \mode_if_horizontal:T
2512     {
2513         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2514         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2515     }
2516 }

```

(End of definition for `__pdf_backend_link_begin_goto:nw` 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.

```

2517 \use_none:n
2518 {
2519     \cs_if_exist:NT \@makecol@hook
2520     {
2521         \tl_put_right:Nn \@makecol@hook
2522         {
2523             \box_if_empty:NF \l_shipout_box
2524             {
2525                 \vbox_set:Nn \l_shipout_box
2526                 {
2527                     \__kernel_backend_postscript:n
2528                     {
2529                         pdf.globaldict /pdf.brokenlink.rect ~ known
2530                         { pdf.bordertracking.continue }
2531                         if
2532                     }
2533                     \vbox_unpack_drop:N \l_shipout_box
2534                     \__kernel_backend_postscript:n
2535                     { pdf.bordertracking.endpage }
2536                 }
2537             }
2538         }
2539         \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2540         \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2541         \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2542     }
2543 }

```

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

```

2544 \cs_new:Npn \__pdf_backend_link_last:
2545   { { 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.

```

2546 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2547   {
2548     \__kernel_backend_postscript:e
2549     {
2550       /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2551     }
2552   }

```

(End of definition for __pdf_backend_link_margin:n.)

__pdf_backend_destination:mn 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.

```

2553 \cs_new_protected:Npn \__pdf_backend_destination:mn #1#2
2554   {
2555     \__kernel_backend_postscript:n { pdf.dest.anchor }
2556     \__pdf_backend_pdfmark:e
2557     {
2558       /View
2559       [
2560         \str_case:nnF {#2}
2561         {
2562           { xyz } { /XYZ ~ pdf.dest.point ~ null }
2563           { fit } { /Fit }
2564           { fitb } { /FitB }
2565           { fitbh } { /FitBH ~ pdf.dest.y }
2566           { fitbv } { /FitBV ~ pdf.dest.x }
2567           { fith } { /FitH ~ pdf.dest.y }
2568           { fitv } { /FitV ~ pdf.dest.x }
2569           { fitr } { /Fit }
2570         }
2571         {
2572           /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2573         }
2574       ]
2575       /Dest ( \exp_not:n {#1} ) cvn
2576       /DEST
2577     }
2578   }
2579 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2580   {
2581     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2582     { \dim_eval:n {#2} } {#1} {#3} {#4}
2583   }
2584 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2585   {
2586     \vbox_to_zero:n

```

```

2587     {
2588       \__kernel_kern:n {#4}
2589       \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2590       \tex_vss:D
2591     }
2592 \__kernel_kern:n {#1}
2593 \vbox_to_zero:n
2594 {
2595   \__kernel_kern:n { -#3 }
2596   \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2597   \tex_vss:D
2598 }
2599 \__kernel_kern:n { -#1 }
2600 \__pdf_backend_pdfmark:n
2601 {
2602   /View
2603   [
2604     /FitR ~
2605     pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2606     pdf.urx ~ pdf.ury ~ pdf.dest2device
2607   ]
2608   /Dest ( #2 ) cvn
2609   /DEST
2610 }
2611 }

```

(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 Doable for the usual ps2pdf method.
\__pdf_backend_compress_objects:n
2612 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2613 {
2614   \int_compare:nNnT {#1} = 0
2615   {
2616     \__kernel_backend_literal_postscript:n
2617     {
2618       /setdistillerparams ~ where
2619       { pop << /CompressPages ~ false >> setdistillerparams }
2620       if
2621     }
2622   }
2623 }
2624 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2625 {
2626   \bool_if:nF {#1}
2627   {
2628     \__kernel_backend_literal_postscript:n
2629     {
2630       /setdistillerparams ~ where
2631       { pop << /CompressStreams ~ false >> setdistillerparams }
2632       if
2633     }

```

```

2634     }
2635   }

```

(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
2636 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2637   {
2638     \cs_gset:Npe \_pdf_backend_version_major: { \int_eval:n {#1} }
2639   }
2640 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2641   {
2642     \cs_gset:Npe \_pdf_backend_version_minor: { \int_eval:n {#1} }
2643   }

```

(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:
2644 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2645 \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:
2646 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2647   { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2648 \cs_new_protected:Npn \_pdf_backend_emc:
2649   { \_pdf_backend_pdfmark:n { /EMC } }

```

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

```

2650 </dvips>

```

6.3 LuaTeX and pdfTeX backend

```

2651 <*luatex | pdftex>

```

6.3.1 Annotations

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

```

2652 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2653   {
2654     <*luatex>
2655     \tex_pdfextension:D annot ~
2656     </luatex>
2657     <*pdftex>
2658     \tex_pdfannot:D
2659     </pdftex>
2660     width ~ \dim_eval:n {#1} ~
2661     height ~ \dim_eval:n {#2} ~
2662     depth ~ \dim_eval:n {#3} ~
2663     {#4}
2664   }

```

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

`_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.

```
2665 \cs_new:Npe \_pdf_backend_annotation_last:
2666   {
2667     \exp_not:N \int_value:w
2668   }
2669   \exp_not:N \tex_pdffeedback:D lastannot ~
2670 \</luatex>
2671 \<*/pdftex>
2672   \exp_not:N \tex_pdflastannot:D
2673 \</pdftex>
2674   \c_space_tl 0 ~ R
2675 }
```

(End of definition for `_pdf_backend_annotation_last:`.)

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

```
\_pdf_backend_link_begin_user:nmw 2676 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nmw #1#2
\_pdf_backend_link_begin:nmw      2677   { \_pdf_backend_link_begin:nmw {#1} { goto~name } {#2} }
\_pdf_backend_link_end:           2678 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nmw #1#2
2679   { \_pdf_backend_link_begin:nmw {#1} { user } {#2} }
2680 \cs_new_protected:Npn \_pdf_backend_link_begin:nmw #1#2#3
2681   {
2682   }
2683   \tex_pdfextension:D startlink ~
2684 \</luatex>
2685 \<*/pdftex>
2686   \tex_pdfstartlink:D
2687 \</pdftex>
2688   attr {#1}
2689   #2 {#3}
2690 }
2691 \cs_new_protected:Npn \_pdf_backend_link_end:
2692   {
2693   }
2694   \tex_pdfextension:D endlink \scan_stop:
2695 \</luatex>
2696 \<*/pdftex>
2697   \tex_pdfendlink:D
2698 \</pdftex>
2699 }
```

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

`_pdf_backend_link_last:` Formatted for direct use.

```
2700 \cs_new:Npe \_pdf_backend_link_last:
2701   {
2702     \exp_not:N \int_value:w
2703   }
2704   \exp_not:N \tex_pdffeedback:D lastlink ~
2705 \</luatex>
```

```

2706 <*pdftex>
2707     \exp_not:N \tex_pdflastlink:D
2708 </pdftex>
2709     \c_space_tl 0 ~ R
2710 }

```

(End of definition for `_pdf_backend_link_last:.`)

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

```

2711 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2712 {
2713 <*luatex>
2714     \tex_pdfvariable:D linkmargin
2715 </luatex>
2716 <*pdftex>
2717     \tex_pdflinkmargin:D
2718 </pdftex>
2719     \dim_eval:n {#1} \scan_stop:
2720 }

```

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

`_pdf_backend_destination:mn`
`_pdf_backend_destination:nmnn` 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.

```

2721 \cs_new_protected:Npn \_pdf_backend_destination:mn #1#2
2722 {
2723 <*luatex>
2724     \tex_pdfextension:D dest ~
2725 </luatex>
2726 <*pdftex>
2727     \tex_pdfdest:D
2728 </pdftex>
2729     name {#1}
2730     \str_case:nnF {#2}
2731     {
2732         { xyz } { xyz }
2733         { fit } { fit }
2734         { fitb } { fitb }
2735         { fitbh } { fitbh }
2736         { fitbv } { fitbv }
2737         { fith } { fith }
2738         { fitv } { fitv }
2739         { fitr } { fitr }
2740     }
2741     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2742     \scan_stop:
2743 }
2744 \cs_new_protected:Npn \_pdf_backend_destination:nmnn #1#2#3#4
2745 {
2746 <*luatex>
2747     \tex_pdfextension:D dest ~
2748 </luatex>
2749 <*pdftex>

```

```

2750     \tex_pdfdest:D
2751 </pdfTeX>
2752     name {#1}
2753     fitr ~
2754     width \dim_eval:n {#2} ~
2755     height \dim_eval:n {#3} ~
2756     depth \dim_eval:n {#4} \scan_stop:
2757 }

```

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

6.3.2 Catalogue entries

```

\__pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
2758 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2759 {
2760 <*luatex>
2761     \tex_pdfextension:D catalog
2762 </luatex>
2763 <*pdfTeX>
2764     \tex_pdfcatalog:D
2765 </pdfTeX>
2766     { / #1 ~ #2 }
2767 }
2768 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2769 {
2770 <*luatex>
2771     \tex_pdfextension:D info
2772 </luatex>
2773 <*pdfTeX>
2774     \tex_pdfinfo:D
2775 </pdfTeX>
2776     { / #1 ~ #2 }
2777 }

```

(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.
2778 \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
\__pdf_backend_object_id:n
2779 \cs_new_protected:Npn \__pdf_backend_object_new:
2780 {
2781 <*luatex>
2782     \tex_pdfextension:D obj ~
2783 </luatex>
2784 <*pdfTeX>
2785     \tex_pdfobj:D
2786 </pdfTeX>
2787     reserveobjnum ~

```



```

2788 \int_gset:Nn \g__pdf_backend_object_int
2789 <*luatex>
2790 { \tex_pdffeedback:D lastobj }
2791 </luatex>
2792 <*pdfTeX>
2793 { \tex_pdflastobj:D }
2794 </pdfTeX>
2795 }
2796 \cs_new:Npn \__pdf_backend_object_ref:n #1 { #1 ~ 0 ~ R }
2797 \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 2798 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
\__pdf_backend_object_write:nn 2799 {
  \__pdf_exp_not_i:nn 2800 <*luatex>
  \__pdf_exp_not_ii:nn 2801 \tex_immediate:D \tex_pdfextension:D obj ~
2802 </luatex>
2803 <*pdfTeX>
2804 \tex_immediate:D \tex_pdfobj:D
2805 </pdfTeX>
2806 useobjnum ~ #1
2807 \__pdf_backend_object_write:nn {#2} {#3}
2808 }
2809 \cs_new:Npn \__pdf_backend_object_write:nn #1#2
2810 {
2811 \str_case:nn {#1}
2812 {
2813 { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2814 { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2815 { fstream }
2816 {
2817 stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2818 file ~ { \__pdf_exp_not_ii:nn #2 }
2819 }
2820 { stream }
2821 {
2822 stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2823 { \__pdf_exp_not_ii:nn #2 }
2824 }
2825 }
2826 }
2827 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2828 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2829 \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 2830 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2831 {
2832 <*luatex>
2833 \tex_immediate:D \tex_pdfextension:D obj ~

```

```

2834 </luatex>
2835 <*pdftex>
2836     \tex_immediate:D \tex_pdfobj:D
2837 </pdftex>
2838     \__pdf_backend_object_write:nn {#1} {#2}
2839 }
2840 \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.

```

2841 \cs_new:Npe \__pdf_backend_object_last:
2842 {
2843     \exp_not:N \int_value:w
2844 <*luatex>
2845     \exp_not:N \tex_pdffeedback:D lastobj ~
2846 </luatex>
2847 <*pdftex>
2848     \exp_not:N \tex_pdflastobj:D
2849 </pdftex>
2850     \c_space_tl 0 ~ R
2851 }

```

(End of definition for __pdf_backend_object_last:.)

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

```

2852 \cs_new:Npe \__pdf_backend_pageobject_ref:n #1
2853 {
2854     \exp_not:N \int_value:w
2855 <*luatex>
2856     \exp_not:N \tex_pdffeedback:D pageref
2857 </luatex>
2858 <*pdftex>
2859     \exp_not:N \tex_pdfpageref:D
2860 </pdftex>
2861     \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2862 }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.3.4 Structure

__pdf_backend_compresslevel:n Simply pass data to the engine.

```

\__pdf_backend_compress_objects:n
\__pdf_backend_objcompresslevel:n
2863 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2864 {
2865     \tex_global:D
2866 <*luatex>
2867     \tex_pdfvariable:D compresslevel
2868 </luatex>
2869 <*pdftex>
2870     \tex_pdfcompresslevel:D
2871 </pdftex>
2872     \int_value:w \int_eval:n {#1} \scan_stop:
2873 }
2874 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1

```

```

2875 {
2876   \bool_if:nTF {#1}
2877     { \__pdf_backend_objcompresslevel:n { 2 } }
2878     { \__pdf_backend_objcompresslevel:n { 0 } }
2879 }
2880 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2881 {
2882   \tex_global:D
2883   \*luatex
2884   \tex_pdfvariable:D objcompresslevel
2885   \*luatex
2886   \*pdftex
2887   \tex_pdfobjcompresslevel:D
2888   \*pdftex
2889   #1 \scan_stop:
2890 }

```

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

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

```

2891 \cs_new_protected:Npe \__pdf_backend_version_major_gset:n #1
2892 {
2893   \*luatex
2894   \int_compare:nNnT \tex_luatexversion:D > { 106 }
2895   {
2896     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2897     \exp_not:N \int_eval:n {#1} \scan_stop:
2898   }
2899   \*luatex
2900   \*pdftex
2901   \cs_if_exist:NT \tex_pdfmajorversion:D
2902   {
2903     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2904     \exp_not:N \int_eval:n {#1} \scan_stop:
2905   }
2906   \*pdftex
2907 }
2908 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2909 {
2910   \tex_global:D
2911   \*luatex
2912   \tex_pdfvariable:D minorversion
2913   \*luatex
2914   \*pdftex
2915   \tex_pdfminorversion:D
2916   \*pdftex
2917   \int_eval:n {#1} \scan_stop:
2918 }

```

(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:` 2919 \cs_new:Npe __pdf_backend_version_major:

```

2920 {
2921 <*luatex>
2922   \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2923     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2924     { 1 }
2925 </luatex>
2926 <*pdftex>
2927   \cs_if_exist:NTF \tex_pdfmajorversion:D
2928     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2929     { 1 }
2930 </pdftex>
2931 }
2932 \cs_new:Npn \__pdf_backend_version_minor:
2933 {
2934   \tex_the:D
2935 <*luatex>
2936   \tex_pdfvariable:D minorversion
2937 </luatex>
2938 <*pdftex>
2939   \tex_pdfminorversion:D
2940 </pdftex>
2941 }

```

(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>.

```

2942 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2943 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2944 \cs_new_protected:Npn \__pdf_backend_emc:
2945 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2946 </luatex | pdftex>

```

6.4 dvipdfmx backend

```

2947 <*dvipdfmx | xetex>

```

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

```

\__pdf_backend:e
2948 \cs_new_protected:Npe \__pdf_backend:n #1
2949 { \__kernel_backend_literal:n { pdf: #1 } }
2950 \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
2951 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2952 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2953 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2954 { \__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.

```
2955 \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 2956 \cs_new_protected:Npn \_pdf_backend_object_new:
\_pdf_backend_object_id:n 2957 { \int_gincr:N \g_pdf_backend_object_int }
2958 \cs_new:Npn \_pdf_backend_object_ref:n #1 { @pdf.obj #1 }
2959 \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 2960 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
\_pdf_backend_object_write_array:nn 2961 {
\_pdf_backend_object_write_dict:nn 2962   \use:c { \_pdf_backend_object_write_ #2 :nn }
\_pdf_backend_object_write_fstream:nn 2963   { \_pdf_backend_object_ref:n {#1} } {#3}
\_pdf_backend_object_write_stream:nn 2964 }
\_pdf_backend_object_write_stream:nnnn 2965 \cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nne }
2966 \cs_new_protected:Npn \_pdf_backend_object_write_array:nn #1#2
2967 {
2968   \_pdf_backend:e
2969   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2970 }
2971 \cs_new_protected:Npn \_pdf_backend_object_write_dict:nn #1#2
2972 {
2973   \_pdf_backend:e
2974   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2975 }
2976 \cs_new_protected:Npn \_pdf_backend_object_write_fstream:nn #1#2
2977 { \_pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2978 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nn #1#2
2979 { \_pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2980 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nnnn #1#2#3#4
2981 {
2982   \_pdf_backend:e
2983   {
2984     #1 stream ~ #2 ~
2985     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2986   }
2987 }
```

(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 2988 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2989 {
2990   \int_gincr:N \g_pdf_backend_object_int
2991   \exp_args:Nne \use:c { \_pdf_backend_object_write_ #1 :nn }
```

```

2992     { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2993     {#2}
2994   }
2995   \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2996   \cs_new:Npn \__pdf_backend_object_last:
2997     { @pdf.obj \int_use:N \g__pdf_backend_object_int }

```

(End of definition for __pdf_backend_object_last:.)

__pdf_backend_pageobject_ref:n Page references are easy in dvipdfmx/X_YTeX.

```

2998   \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2999     { @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.

```

3000   \int_new:N \g__pdf_backend_annotation_int

```

(End of definition for \g__pdf_backend_annotation_int.)

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

```

3001   \cs_new_protected:Npn \__pdf_backend_annotation:nmmn #1#2#3#4
3002     {
3003       \int_gincr:N \g__pdf_backend_object_int
3004       \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
3005       \__pdf_backend:e
3006       {
3007         ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
3008         width ~ \dim_eval:n {#1} ~
3009         height ~ \dim_eval:n {#2} ~
3010         depth ~ \dim_eval:n {#3} ~
3011         << /Type /Annot #4 >>
3012       }
3013     }

```

(End of definition for __pdf_backend_annotation:nmmn.)

__pdf_backend_annotation_last:

```

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

```

3016   \int_new:N \g__pdf_backend_link_int

```

(End of definition for \g__pdf_backend_link_int.)

`_pdf_backend_link_begin_goto:nw` All created using the same internals.

```
\_pdf_backend_link_begin_user:nw 3017 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nw #1#2
\_pdf_backend_link_begin:n 3018 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
\_pdf_backend_link_end: 3019 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nw #1#2
3020 { \_pdf_backend_link_begin:n {#1#2} }
3021 \cs_new_protected:Npe \_pdf_backend_link_begin:n #1
3022 {
3023 \exp_not:N \int_gincr:N \exp_not:N \g__pdf_backend_link_int
3024 \_pdf_backend:e
3025 {
3026 bann ~
3027 @pdf.lnk
3028 \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
3029 \c_space_tl
3030 <<
3031 /Type /Annot
3032 #1
3033 >>
3034 }
3035 }
3036 \cs_new_protected:Npn \_pdf_backend_link_end:
3037 { \_pdf_backend:n { eann } }
```

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

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

```
3038 \cs_new:Npn \_pdf_backend_link_last:
3039 { @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`.

```
3040 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
3041 { \_kernel_backend_literal:e { dvipdfmx:config~g~ \dim_eval:n {#1} } }
```

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

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

```
3042 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
3043 {
3044 \_pdf_backend:e
3045 {
3046 dest ~ ( \exp_not:n {#1} )
3047 [
3048 @thispage
3049 \str_case:nnF {#2}
3050 {
3051 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
3052 { fit } { /Fit }
3053 { fitb } { /FitB }
3054 { fitbh } { /FitBH }

```

```

3055         { fitbv } { /FitBV ~ @xpos }
3056         { fith } { /FitH ~ @ypos }
3057         { fitv } { /FitV ~ @xpos }
3058         { fitr } { /Fit }
3059     }
3060     { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
3061 ]
3062 }
3063 }
3064 \cs_new_protected:Npn \__pdf_backend_destination:nmmm #1#2#3#4
3065 {
3066     \exp_args:Ne \__pdf_backend_destination_aux:nmmm
3067     { \dim_eval:n {#2} } {#1} {#3} {#4}
3068 }
3069 \cs_new_protected:Npn \__pdf_backend_destination_aux:nmmm #1#2#3#4
3070 {
3071     \vbox_to_zero:n
3072     {
3073         \__kernel_kern:n {#4}
3074         \hbox:n
3075         {
3076             \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
3077             \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
3078         }
3079         \tex_vss:D
3080     }
3081     \__kernel_kern:n {#1}
3082     \vbox_to_zero:n
3083     {
3084         \__kernel_kern:n { -#3 }
3085         \hbox:n
3086         {
3087             \__pdf_backend:n
3088             {
3089                 dest ~ (#2)
3090                 [
3091                 @thispage
3092                 /FitR ~
3093                 @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
3094                 @xpos ~ @ypos
3095                 ]
3096             }
3097         }
3098         \tex_vss:D
3099     }
3100     \__kernel_kern:n { -#1 }
3101 }

```

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

6.4.4 Structure

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


```

3102 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
3103   { \__kernel_backend_literal:e { dvipdfmx:config-z~ \int_eval:n {#1} } }
3104 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
3105   {
3106     \bool_if:nF {#1}
3107     { \__kernel_backend_literal:n { dvipdfmx:config-C~0x40 } }
3108   }

```

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

__pdf_backend_version_major_gset:n
 __pdf_backend_version_minor_gset:n

We start with the assumption that the default is active.

```

3109 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
3110   {
3111     \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
3112     \__kernel_backend_literal:e { pdf:majorversion~ \__pdf_backend_version_major: }
3113   }
3114 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
3115   {
3116     \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
3117     \__kernel_backend_literal:e { pdf:minorversion~ \__pdf_backend_version_minor: }
3118   }

```

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

__pdf_backend_version_major:
 __pdf_backend_version_minor:

We start with the assumption that the default is active.

```

3119 \cs_new:Npn \__pdf_backend_version_major: { 1 }
3120 \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
 __pdf_backend_emc:

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

```

3121 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
3122   { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
3123 \cs_new_protected:Npn \__pdf_backend_emc:
3124   { \__kernel_backend_literal_page:n { EMC } }

```

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

```
3125 </dvipdfmx | xetex>
```

6.5 dvisvgm backend

```
3126 <*dvisvgm>
```

6.5.1 Annotations

__pdf_backend_annotation:nnnn

```
3127 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4 { }
```

(End of definition for __pdf_backend_annotation:nnnn.)

__pdf_backend_annotation_last:

```
3128 \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 3129 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2 { }
\_pdf_backend_link_begin:nnnw 3130 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2 { }
\_pdf_backend_link_end: 3131 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3 { }
3132 \cs_new_protected:Npn \_pdf_backend_link_end: { }
```

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

```
\_pdf_backend_link_last:
3133 \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.
3134 \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 3135 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2 { }
3136 \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 3137 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2 { }
3138 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2 { }
(End of definition for \_pdf_backend_catalog_gput:nn and \_pdf_backend_info_gput:nn.)
```

6.5.3 Objects

```
\_pdf_backend_object_new: All no-ops here.
\_pdf_backend_object_ref:n 3139 \cs_new_protected:Npn \_pdf_backend_object_new: { }
\_pdf_backend_object_id:n 3140 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
\_pdf_backend_object_write:nn 3141 \cs_new:Npn \_pdf_backend_object_id:n #1 { }
\_pdf_backend_object_write:ne 3142 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3 { }
\_pdf_backend_object_now:nn 3143 \cs_new_protected:Npn \_pdf_backend_object_write:ne #1#2#3 { }
\_pdf_backend_object_now:ne 3144 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
\_pdf_backend_object_last: 3145 \cs_new_protected:Npn \_pdf_backend_object_now:ne #1#2 { }
\_pdf_backend_pageobject_ref:n 3146 \cs_new:Npn \_pdf_backend_object_last: { }
3147 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1 { }
(End of definition for \_pdf_backend_object_new: and others.)
```

6.5.4 Structure

```

\__pdf_backend_compresslevel:n These are all no-ops.
\__pdf_backend_compress_objects:n 3148 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
3149 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1 { }

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

\__pdf_backend_version_major_gset:n Data not available!
\__pdf_backend_version_minor_gset:n 3150 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
3151 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1 { }

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

\__pdf_backend_version_major: Data not available!
\__pdf_backend_version_minor: 3152 \cs_new:Npn \__pdf_backend_version_major: { -1 }
3153 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

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

\__pdf_backend_bdc:nn More no-ops.
\__pdf_backend_emc: 3154 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
3155 \cs_new_protected:Npn \__pdf_backend_emc: { }

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

3156 </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.

```

3157 <*dviPDFmx | dvips>

\__pdf_backend_pagesize_gset:nn This is done as a backend literal, so we deal with it using the shipout hook.
3158 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2
3159 {
3160   \__kernel_backend_first_shipout:n
3161   {
3162     \__kernel_backend_literal:e
3163     {
3164       <*dviPDFmx>
3165         pdf:pagesize ~
3166         width ~ \dim_eval:n {#1} ~
3167         height ~ \dim_eval:n {#2}
3168       </dviPDFmx>
3169       <*dvips>
3170         papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
3171       </dvips>
3172     }
3173   }
3174 }

```

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

```
3175 </dviptfm | dvips>
3176 <*luatex | pdftex | xetex>
```

`_pdf_backend_pagesize_gset:nn` Pass to the primitives.

```
3177 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
3178 {
3179   \dim_gset:Nn \tex_pagewidth:D {#1}
3180   \dim_gset:Nn \tex_pageheight:D {#2}
3181 }
```

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

```
3182 </luatex | pdftex | xetex>
3183 <*dvisvgm>
```

`_pdf_backend_pagesize_gset:nn` A no-op.

```
3184 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2 { }
```

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

```
3185 </dvisvgm>
3186 </package>
```

7 l3backend-opacity implementation

```
3187 <*package>
3188 <@@=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.

```
3189 <*dvips>
```

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

```
3190 \cs_new_protected:Npn \_opacity_backend_select:n #1
3191 {
3192   \_opacity_backend:nnn {#1} { fill } { ca }
3193   \_opacity_backend:nnn {#1} { stroke } { CA }
3194 }
3195 \cs_new_protected:Npn \_opacity_backend_fill:n #1
3196 {
3197   \_opacity_backend:nnn
3198   { #1 }
3199   { fill }
3200   { ca }
3201 }
```

```

3202 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3203 {
3204   \__opacity_backend:nnn
3205     { #1 }
3206     { stroke }
3207     { CA }
3208 }
3209 \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3210 {
3211   \__kernel_backend_postscript:n
3212   {
3213     product ~ (Ghostscript) ~ search
3214     {
3215       pop ~ pop ~ pop ~
3216       #1 ~ .set #2 constantalpha
3217     }
3218     {
3219       pop ~
3220       mark ~
3221       /#3 ~ #1
3222       /SetTransparency ~
3223       pdfmark
3224     }
3225     ifelse
3226   }
3227 }

```

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

```
3228 </dvips>
```

```
3229 <*dviPDFmx | luatex | pdftex | xetex>
```

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

```

3230 \bool_lazy_and:nnT
3231   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3232   { \pdfmanagement_if_active_p: }
3233   {
3234     <*luatex | pdftex>
3235     \__kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3236     { page ~ direct } { /opacity 1 ~ gs }
3237     </luatex | pdftex>
3238     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3239     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3240   }

```

(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.

```

3241 \tl_new:N \l__opacity_backend_fill_tl
3242 \tl_new:N \l__opacity_backend_stroke_tl
3243 \tl_set:Nn \l__opacity_backend_fill_tl { 1 }
3244 \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: 3245 \cs_new_protected:Npn \__opacity_backend_select:n #1
                          3246 {
                          3247   \tl_set:Nn \l__opacity_backend_fill_tl {#1}
                          3248   \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
                          3249   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
                          3250   { opacity #1 }
                          3251   { << /ca ~ #1 /CA ~ #1 >> }
                          3252   <*dvipdfmx | xetex>
                          3253   \__kernel_backend_literal_pdf:n
                          3254   </dvipdfmx | xetex>
                          3255   <*luatex | pdftex>
                          3256   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
                          3257   </luatex | pdftex>
                          3258   { /opacity #1 ~ gs }
                          3259   \group_insert_after:N \__opacity_backend_reset:
                          3260   }
                          3261 \cs_new_protected:Npn \__opacity_backend_reset:
                          3262 {
                          3263 <*dvipdfmx | xetex>
                          3264 \__kernel_backend_literal_pdf:n
                          3265 { /opacity1 ~ gs }
                          3266 </dvipdfmx | xetex>
                          3267 <*luatex | pdftex>
                          3268 \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int
                          3269 </luatex | pdftex>
                          3270 }

```

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

```

\__opacity_backend_fill:n For separate fill and stroke, we need to work out if we need to do more work or if we can
\__opacity_backend_stroke:n stick to a single setting.
\__opacity_backend_fill_stroke:nn 3271 \cs_new_protected:Npn \__opacity_backend_fill:n #1
                                   3272 {
                                   3273   \exp_args:Nno \__opacity_backend_fill_stroke:nn
                                   3274   { #1 }
                                   3275   { \l__opacity_backend_stroke_tl }
                                   3276   }
                                   3277 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
                                   3278 {
                                   3279   \exp_args:No \__opacity_backend_fill_stroke:nn
                                   3280   { \l__opacity_backend_fill_tl }
                                   3281   { #1 }
                                   3282   }
                                   3283 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
                                   3284 {
                                   3285   \str_if_eq:nnTF {#1} {#2}
                                   3286   { \__opacity_backend_select:n {#1} }
                                   3287   {
                                   3288     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
                                   3289     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
                                   3290     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
                                   3291     { opacity.fill #1 }
                                   3292     { << /ca ~ #1 >> }

```

```

3293     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3294     { opacity.stroke #2 }
3295     { << /CA ~ #2 >> }
3296 < *dvipdfmx | xetex >
3297     \__kernel_backend_literal_pdf:n
3298 < /dvipdfmx | xetex >
3299 < *luatex | pdftex >
3300     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3301 < /luatex | pdftex >
3302     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3303     \group_insert_after:N \__opacity_backend_reset:
3304 }
3305 }

```

(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
3306 \bool_lazy_and:nnF
3307 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3308 { \pdfmanagement_if_active_p: }
3309 {
3310     \cs_gset_protected:Npn \__opacity_backend_select:n #1 { }
3311     \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2 { }
3312 }

```

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

```

3313 < /dvipdfmx | luatex | pdftex | xetex >
3314 < *dvisvgm >

```

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

```

\__opacity_backend_fill:n
\__opacity_backend_stroke:n
\__opacity_backend:nn
3315 \cs_new_protected:Npn \__opacity_backend_select:n #1
3316 { \__opacity_backend:nn {#1} { } }
3317 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3318 { \__opacity_backend:nn {#1} { fill- } }
3319 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3320 { \__opacity_backend:nn {#1} { stroke- } }
3321 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3322 { \__kernel_backend_scope:e { #2 opacity = " #1 " } }

```

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

```

3323 < /dvisvgm >
3324 < /package >

```

7.1 Font handling integration

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

```

3325 < *lua >

```

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

```

3326 local pdfmanagement_active do
3327   local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3328   local cmd = pdfmanagement_if_active_p.cmdname
3329   if cmd == 'undefined_cs' then
3330     pdfmanagement_active = false
3331   else
3332     token.put_next(pdfmanagement_if_active_p)
3333     pdfmanagement_active = token.scan_int() ~= 0
3334   end
3335 end
3336
3337 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3338   luaotfload.set_transparent_colorstack(function() return token.create'c__opacity_backend_st
3339
3340   local transparent_register = {
3341     token.create'pdfmanagement_add:nm',
3342     token.new(0, 1),
3343     'Page/Resources/ExtGState',
3344     token.new(0, 2),
3345     token.new(0, 1),
3346     '',
3347     token.new(0, 2),
3348     token.new(0, 1),
3349     '<</ca ',
3350     '',
3351     '/CA ',
3352     '',
3353     '>>',
3354     token.new(0, 2),
3355   }
3356   luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)
3357     value = (octet * -1):match(value)
3358     if not value then
3359       tex.error'Invalid transparency value'
3360       return
3361     end
3362     value = value:sub(1, -2)
3363     local result = 'opacity' .. value
3364     tex.runtoks(function()
3365       transparent_register[6], transparent_register[10], transparent_register[12] = result,
3366       tex.sprint(-2, transparent_register)
3367     end)
3368     return '/' .. result .. ' gs'
3369   end, 'l3opacity')
3370 end
3371 </lua>

```

8 l3backend-header implementation

```

3372 <*dvips & header>

```

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


```
3373 /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.

```
3374 TeXDict begin
3375 /TeXcolorseparation { setcolor } def
3376 end
```

(End of definition for TeXcolorseparation and separation.)

pdf.globaldict A small global dictionary for backend use.

```
3377 true setglobal
3378 /pdf.globaldict 4 dict def
3379 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 3380 /pdf.cvs { 65534 string cvs } def
3381 /pdf.dvi.pt { 72.27 mul Resolution div } def
3382 /pdf.pt.dvi { 72.27 div Resolution mul } def
3383 /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 3384 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad 3385 /pdf.linkdp.pad { 0 } def
3386 /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 3387 /pdf.rect
pdf.save.linkur 3388 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.llx 3389 /pdf.save.ll
pdf.lly 3390 {
pdf.urx 3391 currentpoint
pdf.ury 3392 /pdf.lly exch def
3393 /pdf.llx exch def
3394 }
3395 def
3396 /pdf.save.ur
3397 {
3398 currentpoint
3399 /pdf.ury exch def
3400 /pdf.urx exch def
3401 }
3402 def
```

```

3403 /pdf.save.linkll
3404 {
3405     currentpoint
3406     pdf.linkmargin add
3407     pdf.linkdp.pad add
3408     /pdf.lly exch def
3409     pdf.linkmargin sub
3410     /pdf.llx exch def
3411 }
3412 def
3413 /pdf.save.linkur
3414 {
3415     currentpoint
3416     pdf.linkmargin sub
3417     pdf.linkht.pad sub
3418     /pdf.ury exch def
3419     pdf.linkmargin add
3420     /pdf.urx exch def
3421 }
3422 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 coordinate 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 3423 /pdf.dest.anchor
pdf.dev.y 3424 {
pdf.tmpa 3425     currentpoint exch
pdf.tmpb 3426     pdf.dvi.pt 72 add
pdf.tmpc 3427     /pdf.dest.x exch def
pdf.tmpd 3428     pdf.dvi.pt
3429     vsize 72 sub exch sub
3430     /pdf.dest.y exch def
3431 }
3432 def
3433 /pdf.dest.point
3434 { pdf.dest.x pdf.dest.y } def
3435 /pdf.dest2device
3436 {
3437     /pdf.dest.y exch def
3438     /pdf.dest.x exch def
3439     matrix currentmatrix
3440     matrix defaultmatrix
3441     matrix invertmatrix
3442     matrix concatmatrix
3443     cvx exec
3444     /pdf.dev.y exch def
3445     /pdf.dev.x exch def
3446     /pdf.tmpd exch def
3447     /pdf.tmpc exch def
3448     /pdf.tmpb exch def

```

```

3449 /pdf.tmpa exch def
3450 pdf.dest.x pdf.tmpa mul
3451 pdf.dest.y pdf.tmpc mul add
3452 pdf.dev.x add
3453 pdf.dest.x pdf.tmpb mul
3454 pdf.dest.y pdf.tmpd mul add
3455 pdf.dev.y add
3456 }
3457 def

```

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

```

pdf.bordertracking To know where a breakable link can go, we need to track the boundary rectangle. That
pdf.bordertracking.begin can be done by hooking into a and x operations: those names have to be retained. The
pdf.bordertracking.end boundary is stored at the end of the operation. Special effort is needed at the start and
pdf.leftboundary end of pages (or rather galleys), such that everything works properly.
pdf.rightboundary
pdf.brokenlink.rect 3458 /pdf.bordertracking false def
pdf.brokenlink.skip 3459 /pdf.bordertracking.begin
pdf.brokenlink.dict 3460 {
pdf.bordertracking.endpage 3461 SDict /pdf.bordertracking true put
pdf.bordertracking.continue 3462 SDict /pdf.leftboundary undef
pdf.originx 3463 SDict /pdf.rightboundary undef
pdf.originy 3464 /a where
3465 {
3466 /a
3467 {
3468 currentpoint pop
3469 SDict /pdf.rightboundary known dup
3470 {
3471 SDict /pdf.rightboundary get 2 index lt
3472 { not }
3473 if
3474 }
3475 if
3476 { pop }
3477 { SDict exch /pdf.rightboundary exch put }
3478 ifelse
3479 moveto
3480 currentpoint pop
3481 SDict /pdf.leftboundary known dup
3482 {
3483 SDict /pdf.leftboundary get 2 index gt
3484 { not }
3485 if
3486 }
3487 if
3488 { pop }
3489 { SDict exch /pdf.leftboundary exch put }
3490 ifelse
3491 }
3492 put
3493 }
3494 if
3495 }

```

```

3496     def
3497 /pdf.bordertracking.end
3498     {
3499     /a where { /a { moveto } put } if
3500     /x where { /x { 0 exch rmoveto } put } if
3501     SDict /pdf.leftboundary known
3502     { pdf.outerbox 0 pdf.leftboundary put }
3503     if
3504     SDict /pdf.rightboundary known
3505     { pdf.outerbox 2 pdf.rightboundary put }
3506     if
3507     SDict /pdf.bordertracking false put
3508     }
3509     def
3510 /pdf.bordertracking.endpage
3511 {
3512 pdf.bordertracking
3513     {
3514     pdf.bordertracking.end
3515     true setglobal
3516     pdf.globaldict
3517     /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3518     pdf.globaldict
3519     /pdf.brokenlink.skip pdf.baselineskip put
3520     pdf.globaldict
3521     /pdf.brokenlink.dict
3522     pdf.link.dict pdf.cvs put
3523     false setglobal
3524     mark pdf.link.dict cvx exec /Rect
3525     [
3526     pdf.llx
3527     pdf.lly
3528     pdf.outerbox 2 get pdf.linkmargin add
3529     currentpoint exch pop
3530     pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3531     ]
3532     /ANN pdf.pdfmark
3533     }
3534 if
3535 }
3536 def
3537 /pdf.bordertracking.continue
3538 {
3539 /pdf.link.dict pdf.globaldict
3540 /pdf.brokenlink.dict get def
3541 /pdf.outerbox pdf.globaldict
3542 /pdf.brokenlink.rect get def
3543 /pdf.baselineskip pdf.globaldict
3544 /pdf.brokenlink.skip get def
3545 pdf.globaldict dup dup
3546 /pdf.brokenlink.dict undef
3547 /pdf.brokenlink.skip undef
3548 /pdf.brokenlink.rect undef
3549 currentpoint

```

```

3550 /pdf.originy exch def
3551 /pdf.originx exch def
3552 /a where
3553 {
3554 /a
3555 {
3556 moveto
3557 SDict
3558 begin
3559 currentpoint pdf.originy ne exch
3560 pdf.originx ne or
3561 {
3562 pdf.save.linkll
3563 /pdf.lly
3564 pdf.lly pdf.outerbox 1 get sub def
3565 pdf.bordertracking.begin
3566 }
3567 if
3568 end
3569 }
3570 put
3571 }
3572 if
3573 /x where
3574 {
3575 /x
3576 {
3577 0 exch rmoveto
3578 SDict
3579 begin
3580 currentpoint
3581 pdf.originy ne exch pdf.originx ne or
3582 {
3583 pdf.save.linkll
3584 /pdf.lly
3585 pdf.lly pdf.outerbox 1 get sub def
3586 pdf.bordertracking.begin
3587 }
3588 if
3589 end
3590 }
3591 put
3592 }
3593 if
3594 }
3595 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.

```

3596 /pdf.breaklink
3597 {
3598   pop
3599   counttomark 2 mod 0 eq
3600   {
3601     counttomark /pdf.count exch def
3602     {
3603       pdf.count 0 eq { exit } if
3604       counttomark 2 roll
3605       1 index /Rect eq
3606       {
3607         dup 4 array copy
3608         dup dup
3609         1 get
3610         pdf.outerbox pdf.rect.ht
3611         pdf.linkmargin 2 mul add sub
3612         3 exch put
3613         dup
3614         pdf.outerbox 2 get
3615         pdf.linkmargin add
3616         2 exch put
3617         dup dup
3618         3 get
3619         pdf.outerbox pdf.rect.ht
3620         pdf.linkmargin 2 mul add add
3621         1 exch put
3622         /pdf.currentrect exch def
3623         pdf.breaklink.write
3624         {
3625           pdf.currentrect
3626           dup
3627           pdf.outerbox 0 get
3628           pdf.linkmargin sub
3629           0 exch put
3630           dup
3631           pdf.outerbox 2 get
3632           pdf.linkmargin add
3633           2 exch put
3634           dup dup
3635           1 get
3636           pdf.baselineskip add
3637           1 exch put
3638           dup dup
3639           3 get
3640           pdf.baselineskip add
3641           3 exch put
3642           /pdf.currentrect exch def
3643           pdf.breaklink.write
3644         }
3645         1 index 3 get
3646         pdf.linkmargin 2 mul add
3647         pdf.outerbox pdf.rect.ht add
3648         2 index 1 get sub
3649         pdf.baselineskip div round cvi 1 sub

```

```

3650         exch
3651     repeat
3652     pdf.currentrect
3653     dup
3654         pdf.outerbox 0 get
3655         pdf.linkmargin sub
3656         0 exch put
3657     dup dup
3658         1 get
3659         pdf.baselineskip add
3660         1 exch put
3661     dup dup
3662         3 get
3663         pdf.baselineskip add
3664         3 exch put
3665     dup 2 index 2 get 2 exch put
3666     /pdf.currentrect exch def
3667     pdf.breaklink.write
3668     SDict /pdf.pdfmark.good false put
3669     exit
3670     }
3671     { pdf.count 2 sub /pdf.count exch def }
3672     ifelse
3673     }
3674     loop
3675     }
3676     if
3677     /ANN
3678     }
3679     def
3680 /pdf.breaklink.write
3681     {
3682     counttomark 1 sub
3683     index /_objdef eq
3684     {
3685         counttomark -2 roll
3686         dup wcheck
3687         {
3688             readonly
3689             counttomark 2 roll
3690         }
3691         { pop pop }
3692     } ifelse
3693     }
3694     if
3695     counttomark 1 add copy
3696     pop pdf.currentrect
3697     /ANN pdfmark
3698     }
3699     def

```

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

pdf.pdfmark The business end of breaking links starts by hooking into pdfmarks. Unlike hypdvips,
pdf.pdfmark.good we avoid altering any links we have not created by using a copy of the core pdfmarks
pdf.outerbox
pdf.baselineskip
pdf.pdfmark.dict

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.

```

3700 /pdf.pdfmark
3701 {
3702   SDict /pdf.pdfmark.good true put
3703   dup /ANN eq
3704   {
3705     pdf.pdfmark.store
3706     pdf.pdfmark.dict
3707     begin
3708       Subtype /Link eq
3709       currentdict /Rect known and
3710       SDict /pdf.outerbox known and
3711       SDict /pdf.baselineskip known and
3712       {
3713         Rect 3 get
3714         pdf.linkmargin 2 mul add
3715         pdf.outerbox pdf.rect.ht add
3716         Rect 1 get sub
3717         pdf.baselineskip div round cvi 0 gt
3718         { pdf.breaklink }
3719         if
3720       }
3721       if
3722     end
3723     SDict /pdf.outerbox undef
3724     SDict /pdf.baselineskip undef
3725     currentdict /pdf.pdfmark.dict undef
3726   }
3727   if
3728   pdf.pdfmark.good
3729   { pdfmark }
3730   { cleartomark }
3731   ifelse
3732 }
3733 def
3734 /pdf.pdfmark.store
3735 {
3736   /pdf.pdfmark.dict 65534 dict def
3737   counttomark 1 add copy
3738   pop
3739   {
3740     dup mark eq
3741     {
3742       pop
3743       exit
3744     }
3745     {
3746       pdf.pdfmark.dict
3747       begin def end
3748     }
3749     ifelse
3750   }

```



```
3751     loop
3752   }
3753   def
```

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

```
3754 </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>	1119
A	
<code>\AtBeginDvi</code>	56
B	
bool commands:	
<code>\bool_gset_false:N</code>	1205, 1224, 1247, 1269, 1285, 1389, 1628, 1664, 2404, 2450
<code>\bool_gset_true:N</code>	1203, 1272, 1387, 1643, 2397, 2403
<code>\bool_if:NTF</code>	66, 578, 1215, 1219, 1235, 1238, 1242, 1253, 1260, 1264, 1276, 1280, 1400, 1405, 1410, 1602, 1647, 1786, 1836, 1976, 2018, 2392, 2407, 2412, 2417
<code>\bool_if:nTF</code>	2626, 2876, 3106
<code>\bool_lazy_and:nnTF</code>	791, 2135, 3230, 3306
<code>\bool_lazy_any:nTF</code>	1825
<code>\bool_lazy_or:nnTF</code>	2011
<code>\bool_new:N</code>	1206, 1273, 1390, 1644, 2377, 2378
<code>\bool_set_false:N</code>	1798, 1940, 2042, 2206
box commands:	
<code>\box_dp:N</code>	217, 219, 267, 269, 324, 326, 373, 375, 377, 379, 2429, 2462, 2463, 2488
<code>\box_ht:N</code>	219, 269, 326, 377, 379, 1849, 2083, 2434, 2473, 2474, 2490
<code>\box_if_empty:N</code>	2523
<code>\box_move_down:nn</code>	2351, 2429
<code>\box_move_up:nn</code>	2227, 2353, 2434
<code>\box_new:N</code>	2253, 2341, 2342
<code>\box_set_dp:Nn</code>	1727
<code>\box_set_ht:Nn</code>	1726
<code>\box_set_wd:Nn</code>	281, 1725
<code>\box_use:N</code>	224, 242, 256, 272, 299, 313, 329, 345, 357, 408, 422, 441, 1340, 1535, 1728, 2382
<code>\box_wd:N</code>	218, 226, 268, 274, 325, 331, 374, 376, 1848, 2082
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>	1293, 1420, 1671
color internal commands:	
<code>__color_backend:nnn</code>	1027, 1042, 1050, 1056
<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, 1077, 1077
<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, 1029
<code>__color_backend_fill_devicen:nn</code>	941, 951, 981, 985, 1007, 1011, 1071, 1073
<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, 1075, 1075
<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, 1071, 1071, 1073
<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 , 1077 , 1078	<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 , 1031 , 1075	<code>init_count:w</code> ... 576 , 639 , 640 , 644
<code>_color_backend_rgb:w</code>	<code>_color_backend_separation_-</code>
..... 1044	<code>init_Device:Nn</code>
<code>_color_backend_select:n</code> 576 , 620 , 622 , 624 , 625
..... 488 , 489 , 491 , 493 , 495 , 496 , 527 , 527 , 529 , 530 , 531 , 573	<code>\l_color_backend_stack_int</code>
<code>_color_backend_select:mn</code> 449 , 521 , 524 , 966 , 978
..... 511 , 512 , 514 , 516 , 517 , 784	<code>_color_backend_stroke:n</code>
<code>_color_backend_select_cmyk:n</code> 931 , 936 , 938 , 939 , 940 , 957 , 970 , 972 , 974 , 975 , 984
..... 488 , 488 , 511 , 511 , 527 , 529	<code>_color_backend_stroke_cmyk:n</code> ..
<code>_color_backend_select_devicen:nn</code> 931 , 938 , 957 , 969 , 991 , 1001 , 1027 , 1027
..... 572 , 574 , 756 , 757 , 778 , 786	<code>_color_backend_stroke_devicen:nn</code>
<code>_color_backend_select_gray:n</code> 941 , 952 , 981 , 986 , 1007 , 1012 , 1071 , 1074
..... 488 , 490 , 511 , 513 , 527 , 530 , 537	<code>_color_backend_stroke_gray:n</code> ..
<code>_color_backend_select_iccbased:nn</code> 931 , 939 , 957 , 971 , 991 , 1003 , 1027 , 1033
..... 575 , 575 , 760 , 760 , 778 , 787	<code>_color_backend_stroke_gray_-</code>
<code>_color_backend_select_named:n</code> .	<code>aux:n</code> 1027 , 1037 , 1041
..... 488 , 492 , 534 , 534	<code>_color_backend_stroke_reset:</code> ..
<code>_color_backend_select_rgb:n</code> 953 , 954 , 987 , 988 , 1013 , 1014 , 1075 , 1076
..... 488 , 494 , 511 , 515 , 527 , 531	<code>_color_backend_stroke_rgb:n</code> ...
<code>_color_backend_select_separation:nn</code> 931 , 940 , 957 , 973 , 991 , 1005 , 1027 , 1043
..... 572 , 572 , 574 , 756 , 756 , 757 , 778 , 779 , 783 , 786 , 787	<code>_color_backend_stroke_rgb:w</code> ...
<code>_color_backend_separation_-</code> 1027 , 1045
<code>init:n</code> 576 , 657 , 670	<code>_color_backend_stroke_separation:nn</code>
<code>_color_backend_separation_-</code> 941 , 946 , 952 , 981 , 983 , 986 , 1007 , 1009 , 1012 , 1071 , 1072 , 1074
<code>init:nn</code> 805 , 815 , 819	<code>\l_color_backend_stroke_tl</code>
<code>_color_backend_separation_-</code> 507 , 520 , 967 , 977
<code>init:nnn</code> 576 , 611 , 632	<code>\g_color_model_int</code> 583 , 592 , 740 , 768 , 817 , 823 , 824 , 878 , 879 , 888 , 912
<code>_color_backend_separation_-</code>	<code>\c_color_model_range_CIELAB_tl</code> .
<code>init:nnnn</code> 576 , 634 , 646 695 , 730 , 841 , 848
<code>_color_backend_separation_-</code>	<code>color.sc</code> 3373
<code>init:nnnnn</code> 576 , 576 , 597 , 690 , 758 , 758 , 805 , 805 , 845	<code>cs commands:</code>
<code>_color_backend_separation_-</code>	<code>\cs_generate_variant:Nn</code>
<code>init:nw</code> 576 , 661 , 672 , 686 62 , 65 , 98 , 147 , 152 , 163 , 194 , 200 , 597 , 1151 , 1350 , 1544 , 1990 , 2053 , 2073 , 2258 , 2273 , 2336 , 2827 , 2840 , 2950 , 2965 , 2995
<code>_color_backend_separation_-</code>	<code>\cs_gset:Npe</code> .. 2638 , 2642 , 3111 , 3116
<code>init:w</code> 576 , 648 , 663 , 668	<code>\cs_gset_protected:Npn</code> ... 3310 , 3311
<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_if_exist:NTF</code>	1494, 1506, 1518, 1525, 1547, 1553,
..... 27, 49, 1738, 2519, 2901, 2927	1558, 1563, 1574, 1584, 1594, 1596,
<code>\cs_if_exist_p:N</code>	1598, 1600, 1631, 1633, 1638, 1640,
<code>\cs_if_exist_use:NTF</code>	1642, 1645, 1666, 1677, 1690, 1692,
<code>\cs_new:Npe</code>	1694, 1696, 1698, 1700, 1702, 1704,
545, 2665, 2700, 2841, 2852, 2919, 3133	1706, 1714, 1736, 1755, 1778, 1795,
<code>\cs_new:Npn</code>	1809, 1814, 1822, 1852, 1865, 1883,
623, 625, 632, 638, 640, 646, 663,	1893, 1909, 1928, 1937, 1945, 1957,
670, 672, 890, 1298, 1425, 1675,	1963, 1966, 1981, 1991, 2030, 2039,
1851, 2086, 2244, 2265, 2337, 2339,	2045, 2051, 2054, 2061, 2074, 2079,
2372, 2544, 2644, 2645, 2796, 2797,	2087, 2094, 2111, 2145, 2176, 2177,
2809, 2828, 2829, 2932, 2958, 2996,	2179, 2181, 2183, 2189, 2195, 2203,
2998, 3014, 3038, 3119, 3120, 3128,	2209, 2212, 2214, 2225, 2256, 2259,
3140, 3141, 3146, 3147, 3152, 3153	2261, 2263, 2267, 2274, 2291, 2296,
<code>\cs_new_eq:NN</code> 46, 56, 58, 529, 530,	2301, 2306, 2316, 2321, 2329, 2344,
531, 574, 757, 786, 787, 933, 934,	2349, 2381, 2383, 2388, 2390, 2395,
935, 938, 939, 940, 951, 952, 953,	2410, 2415, 2452, 2481, 2500, 2509,
954, 985, 986, 987, 988, 1011, 1012,	2546, 2553, 2579, 2584, 2612, 2624,
1013, 1073, 1074, 1075, 1150, 1349,	2636, 2640, 2646, 2648, 2652, 2676,
1355, 1356, 1543, 1545, 1546, 1552,	2678, 2680, 2691, 2711, 2721, 2744,
1752, 1753, 1766, 1768, 1793, 1794,	2758, 2768, 2779, 2798, 2830, 2863,
1857, 1858, 1859, 1882, 1907, 1924,	2874, 2880, 2908, 2942, 2944, 2951,
1925, 1934, 1935, 1936, 1956, 1959,	2953, 2956, 2960, 2966, 2971, 2976,
1960, 1961, 2026, 2036, 2037, 2038,	2978, 2980, 2988, 3001, 3017, 3019,
2192, 2193, 2201, 2202, 2211, 2241,	3036, 3040, 3042, 3064, 3069, 3102,
2242, 2243, 2247, 2266, 2382, 2959	3104, 3109, 3114, 3121, 3123, 3127,
<code>\cs_new_protected:Npe</code>	3129, 3130, 3131, 3132, 3134, 3135,
..... 576, 1056, 2891, 2948, 3021	3136, 3137, 3138, 3139, 3142, 3143,
<code>\cs_new_protected:Npn</code> . 47, 53, 60,	3144, 3145, 3148, 3149, 3150, 3151,
63, 71, 77, 82, 84, 88, 99, 109, 119,	3154, 3155, 3158, 3177, 3184, 3190,
128, 137, 150, 153, 155, 157, 161,	3195, 3202, 3209, 3245, 3261, 3271,
166, 175, 185, 195, 206, 228, 230,	3277, 3283, 3315, 3317, 3319, 3321
245, 261, 276, 278, 304, 318, 333,	<code>\cs_set_eq:NN</code>
335, 348, 362, 412, 425, 452, 466,	<code>\cs_set_protected:Npn</code>
476, 488, 490, 492, 494, 496, 503,	2149
511, 513, 515, 517, 523, 527, 532,	
534, 572, 575, 598, 688, 734, 753,	
756, 758, 759, 760, 779, 783, 788,	
805, 819, 830, 852, 896, 914, 931,	
936, 941, 946, 957, 959, 961, 963,	
969, 971, 973, 975, 981, 983, 991,	
993, 995, 997, 1001, 1003, 1005,	
1007, 1009, 1014, 1017, 1019, 1021,	
1023, 1027, 1033, 1041, 1043, 1045,	
1071, 1072, 1076, 1077, 1078, 1152,	
1158, 1163, 1165, 1167, 1175, 1183,	
1192, 1202, 1204, 1207, 1209, 1226,	
1231, 1249, 1271, 1274, 1287, 1300,	
1305, 1307, 1309, 1311, 1313, 1315,	
1317, 1319, 1324, 1351, 1353, 1357,	
1362, 1367, 1377, 1386, 1388, 1391,	
1393, 1395, 1397, 1402, 1407, 1412,	
1414, 1427, 1432, 1434, 1436, 1438,	
1440, 1442, 1444, 1446, 1457, 1482,	

D	
dim commands:	
<code>\dim_compare:nNnTF</code>	2125, 2130
<code>\dim_compare_p:nNn</code>	2136, 2137
<code>\dim_eval:n</code>	
... 2347, 2582, 2660, 2661, 2662,	
2719, 2754, 2755, 2756, 3008, 3009,	
3010, 3041, 3067, 3166, 3167, 3170	
<code>\dim_gset:Nn</code>	3179, 3180
<code>\dim_max:nn</code>	2460, 2471
<code>\dim_set:Nn</code>	
.. 1848, 1849, 2082, 2083, 2121, 2122	
<code>\dim_set_eq:NN</code>	2187
<code>\dim_to_decimal:n</code> ..	373, 374, 375,
376, 377, 379, 1556, 1561, 1567,	
1568, 1569, 1570, 1579, 1580, 1581,	
1672, 1691, 2234, 2235, 2458, 2469,	
2487, 2488, 2489, 2490, 2494, 2550	
<code>\dim_to_decimal_in_bp:n</code>	

. 217, 218, 219, 267, 268, 269,
 324, 325, 326, 1171, 1172, 1179,
 1180, 1187, 1188, 1196, 1197, 1198,
 1295, 1299, 1303, 1360, 1365, 1371,
 1372, 1373, 1381, 1382, 1422, 1426,
 1430, 1676, 1760, 1761, 1762, 1763,
 1950, 1951, 1952, 1953, 2005, 2006,
 2007, 2008, 2219, 2220, 2221, 2222
 \dim_zero:N 2119, 2120
 \c_max_dim
 2121, 2122, 2125, 2130, 2136, 2137
 draw internal commands:
 __draw_backend_add_to_path:n
 1553,
 1555, 1560, 1565, 1576, 1584, 1599
 __draw_backend_begin:
 1152, 1152, 1351, 1351, 1547, 1547
 __draw_backend_box_use:Nnnnn
 1324, 1324, 1525, 1525, 1714, 1714
 __draw_backend_cap_but:
 1287, 1307, 1414, 1434, 1666, 1694
 __draw_backend_cap_rectangle:
 1287, 1311, 1414, 1438, 1666, 1698
 __draw_backend_cap_round:
 1287, 1309, 1414, 1436, 1666, 1696
 __draw_backend_clip:
 1207, 1271, 1391, 1407, 1598, 1642
 __draw_backend_closepath:
 1207, 1207,
 1228, 1391, 1391, 1598, 1598, 1635
 __draw_backend_closestroke:
 1207, 1226, 1391, 1395, 1598, 1633
 __draw_backend_cm:nnnn
 1319, 1319, 1335, 1336, 1337,
 1446, 1446, 1529, 1706, 1706, 1717
 __draw_backend_cm_aux:nnnn
 1446, 1453, 1457
 __draw_backend_cm_decompose:nnnnN
 1452, 1481, 1482
 __draw_backend_cm_decompose_
 auxi:nnnnN 1481, 1486, 1494
 __draw_backend_cm_decompose_
 auxii:nnnnN 1481, 1498, 1506
 __draw_backend_cm_decompose_
 auxiii:nnnnN 1481, 1510, 1518
 __draw_backend_curveto:nnnnnn
 1167, 1192, 1357, 1367, 1553, 1574
 __draw_backend_dash:n
 1287, 1293, 1298,
 1414, 1420, 1425, 1666, 1671, 1675
 __draw_backend_dash_aux:nn
 1666, 1670, 1677
 __draw_backend_dash_pattern:nn
 1287, 1287, 1414, 1414, 1666, 1666
 __draw_backend_discardpath:
 1207, 1274, 1391, 1412, 1598, 1645
 __draw_backend_end:
 1152, 1158, 1351, 1353, 1547, 1552
 __draw_backend_evenodd_rule:
 1202, 1202, 1386, 1386, 1594, 1594
 __draw_backend_fill:
 1207, 1231, 1391, 1397, 1598, 1638
 __draw_backend_fillstroke:
 1207, 1249, 1391, 1402, 1598, 1640
 __draw_backend_join_bevel:
 1287, 1317, 1414, 1444, 1666, 1704
 __draw_backend_join_miter:
 1287, 1313, 1414, 1440, 1666, 1700
 __draw_backend_join_round:
 1287, 1315, 1414, 1442, 1666, 1702
 __draw_backend_lineto:mn
 1167, 1175, 1357, 1362, 1553, 1558
 __draw_backend_linewidth:n
 1287, 1300, 1414, 1427, 1666, 1690
 __draw_backend_literal:n
 1150, 1150, 1151, 1151, 1154,
 1155, 1156, 1160, 1161, 1164, 1166,
 1169, 1177, 1185, 1194, 1208, 1211,
 1212, 1213, 1214, 1217, 1223, 1233,
 1240, 1246, 1251, 1256, 1257, 1258,
 1259, 1262, 1268, 1278, 1284, 1289,
 1302, 1306, 1308, 1310, 1312, 1314,
 1316, 1318, 1321, 1326, 1327, 1328,
 1329, 1330, 1331, 1332, 1333, 1334,
 1338, 1339, 1341, 1342, 1343, 1344,
 1345, 1349, 1349, 1350, 1359, 1364,
 1369, 1379, 1392, 1394, 1396, 1399,
 1404, 1409, 1413, 1416, 1429, 1433,
 1435, 1437, 1439, 1441, 1443, 1445,
 1543, 1543, 1544, 1605, 1624, 1650
 __draw_backend_miterlimit:n
 1287, 1305, 1414, 1432, 1666, 1692
 __draw_backend_moveto:nn
 1167, 1167, 1357, 1357, 1553, 1553
 __draw_backend_nonzero_rule:
 1202, 1204, 1386, 1388, 1594, 1596
 __draw_backend_path:n
 1598, 1600, 1632, 1639, 1641
 \g__draw_backend_path_int 1613, 1630
 \g__draw_backend_path_tl
 1553, 1609, 1625, 1627, 1654, 1663
 __draw_backend_rectangle:nnnn
 1167, 1183, 1357, 1377, 1553, 1563
 __draw_backend_scope_begin: 1163,
 1163, 1352, 1355, 1355, 1545, 1545
 __draw_backend_scope_end: 1163,
 1165, 1354, 1355, 1356, 1545, 1546

`__draw_backend_stroke`: [1207](#), [1209](#),
[1229](#), [1391](#), [1393](#), [1598](#), [1631](#), [1636](#)
`\g__draw_draw_clip_bool` . . . [1207](#), [1598](#)
`\g__draw_draw_eor_bool`
. . . [1202](#), [1219](#), [1235](#), [1242](#), [1253](#),
[1264](#), [1280](#), [1386](#), [1400](#), [1405](#), [1410](#)
`\g__draw_draw_path_int` [1598](#)

E

`\errmessage` [38](#)
`\evensidemargin` [2427](#)
exp commands:
`\exp_after:wN` [2092](#)
`\exp_args:Ne` [580](#),
[634](#), [815](#), [1816](#), [1871](#), [1873](#), [1897](#),
[1899](#), [2303](#), [2318](#), [2423](#), [2581](#), [3066](#)
`\exp_args:Nf` [1292](#), [1419](#), [2346](#)
`\exp_args:Nne` [2991](#)
`\exp_args:NNf` [229](#), [277](#), [334](#)
`\exp_args:Nno` [3273](#)
`\exp_args:No` [3279](#)
`\exp_not:N` . [547](#), [553](#), [554](#), [555](#), [580](#),
[582](#), [583](#), [586](#), [587](#), [592](#), [2667](#), [2669](#),
[2672](#), [2702](#), [2704](#), [2707](#), [2843](#), [2845](#),
[2848](#), [2854](#), [2856](#), [2859](#), [2896](#), [2897](#),
[2903](#), [2904](#), [2923](#), [2928](#), [3023](#), [3028](#)
`\exp_not:n` [48](#), [96](#), [107](#), [145](#),
[904](#), [2294](#), [2299](#), [2575](#), [2813](#), [2814](#),
[2828](#), [2829](#), [2969](#), [2974](#), [2985](#), [3046](#)
`\ExplBackendFileDate` [1](#)

F

file commands:
`\file_compare_timestamp:nNnTF` . [1885](#)
`\file_parse_full_name:nNNN` [1867](#), [1895](#)
`\fmtversion` [51](#)
fp commands:
`\fp_compare:nNnTF`
. [236](#), [283](#), [289](#), [341](#), [1462](#), [1475](#), [1520](#)
`\fp_eval:n`
. [229](#), [238](#), [251](#), [252](#), [277](#), [294](#), [309](#),
[311](#), [334](#), [343](#), [354](#), [355](#), [419](#), [434](#),
[435](#), [1038](#), [1051](#), [1052](#), [1053](#), [1464](#),
[1469](#), [1470](#), [1477](#), [1487](#), [1488](#), [1489](#),
[1490](#), [1499](#), [1500](#), [1501](#), [1502](#), [1511](#),
[1512](#), [1513](#), [1514](#), [2572](#), [2741](#), [3060](#)
`\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](#), [1462](#), [1475](#)

G

graphics commands:
`\l_graphics_search_ext_seq`
. [1748](#), [1771](#), [1917](#), [2105](#)
graphics internal commands:
`\l_graphics_attr_tl` [1777](#),
[1782](#), [1799](#), [1811](#), [1818](#), [1820](#), [1855](#)
`__graphics_backend_dequote:w`
. [1778](#), [1817](#), [1851](#)
`\l_graphics_backend_dir_str` . [1860](#)
`\l_graphics_backend_ext_str` . [1860](#)
`__graphics_backend_get_pagecount:n`
. [1767](#), [1768](#), [1909](#), [1909](#),
[2024](#), [2026](#), [2094](#), [2094](#), [2246](#), [2247](#)
`__graphics_backend_getbb_auxi:n`
. [1778](#), [1791](#), [1807](#), [1809](#)
`__graphics_backend_getbb_-
auxi:nN` [2030](#), [2034](#), [2043](#), [2045](#)
`__graphics_backend_getbb_-
auxii:n` [1778](#), [1812](#), [1814](#)
`__graphics_backend_getbb_-
auxiii:nnN` . . [2030](#), [2048](#), [2051](#), [2053](#)
`__graphics_backend_getbb_-
auxiiii:n` [1778](#), [1816](#), [1822](#)
`__graphics_backend_getbb_-
auxiii:nNnn` . [2030](#), [2049](#), [2052](#), [2054](#)
`__graphics_backend_getbb_-
auxiv:nnNnn` . [2030](#), [2057](#), [2061](#), [2073](#)
`__graphics_backend_getbb_-
auxv:nNnn` . . [2030](#), [2058](#), [2065](#), [2074](#)
`__graphics_backend_getbb_-
auxvi:nNnn` [2077](#), [2079](#)
`__graphics_backend_getbb_bmp:n` .
. [1922](#), [1936](#), [2030](#), [2038](#)
`__graphics_backend_getbb_eps:n` .
. [1750](#), [1752](#), [1860](#),
[1865](#), [1882](#), [1922](#), [1924](#), [2190](#), [2192](#)
`__graphics_backend_getbb_eps:nm`
. [1860](#)
`__graphics_backend_getbb_eps:nn`
. [1871](#), [1883](#)
`__graphics_backend_getbb_jpeg:n`
. [1778](#), [1793](#),
[1922](#), [1934](#), [2030](#), [2036](#), [2195](#), [2201](#)
`__graphics_backend_getbb_jpg:n` .
[1778](#), [1778](#), [1793](#), [1794](#), [1922](#), [1928](#),
[1934](#), [1935](#), [1936](#), [2030](#), [2030](#), [2036](#),
[2037](#), [2038](#), [2195](#), [2195](#), [2201](#), [2202](#)
`__graphics_backend_getbb_-
pagebox:w` . . [2030](#), [2069](#), [2086](#), [2092](#)
`__graphics_backend_getbb_pdf:n` .
. [1778](#), [1795](#), [1891](#),
[1922](#), [1937](#), [2030](#), [2039](#), [2203](#), [2203](#)

<code>__graphics_backend_getbb_png:n</code> .	1778 , 1794 ,
	1922 , 1935 , 2030 , 2037 , 2195 , 2202
<code>__graphics_backend_getbb_ps:n</code> . .	1750 , 1753 ,
	1860 , 1882 , 1922 , 1925 , 2190 , 2193
<code>__graphics_backend_getbb_svg:n</code> .	2111 , 2111
<code>__graphics_backend_getbb_svg_-auxi:nNn</code> . . .	2111 , 2127 , 2132 , 2145
<code>__graphics_backend_getbb_svg_-auxii:w</code>	2111 , 2149 , 2171 , 2176
<code>__graphics_backend_getbb_svg_-auxiii:Nw</code>	2111 , 2159 , 2177
<code>__graphics_backend_getbb_svg_-auxiv:Nw</code>	2111 , 2162 , 2179
<code>__graphics_backend_getbb_svg_-auxv:Nw</code>	2111 , 2163 , 2181
<code>__graphics_backend_getbb_svg_-auxvi:Nn</code> 2111 , 2178 , 2180 , 2182 , 2183	
<code>__graphics_backend_getbb_svg_-auxvii:w</code>	2111 , 2185 , 2189
<code>__graphics_backend_include:nn</code> . .	2209 , 2210 , 2213 , 2214
<code>__graphics_backend_include_-auxi:nn</code>	1945 , 1958 , 1964 , 1966
<code>__graphics_backend_include_-auxii:nnn</code> . .	1945 , 1968 , 1981 , 1990
<code>__graphics_backend_include_-auxiii:nnn</code>	1945 , 1988 , 1991
<code>__graphics_backend_include_-bmp:n</code>	1945 , 1961
<code>__graphics_backend_include_-dequote:w</code>	2225 , 2236 , 2244
<code>__graphics_backend_include_-eps:n</code>	1755 ,
	1755 , 1766 , 1860 , 1893 , 1907 ,
	1945 , 1945 , 1956 , 2209 , 2209 , 2211
<code>__graphics_backend_include_-jpeg:n</code> .	1852 , 1857 , 1959 , 2225 , 2242
<code>__graphics_backend_include_-jpg:n</code>	1852 ,
	1852 , 1857 , 1858 , 1859 , 1945 ,
	1957 , 1959 , 1960 , 1961 , 2225 , 2243
<code>__graphics_backend_include_-jpsseg:n</code>	1945
<code>__graphics_backend_include_-pdf:n</code>	1852 , 1858 , 1897 ,
	1945 , 1963 , 2087 , 2087 , 2209 , 2212
<code>__graphics_backend_include_-png:n</code>	1852 , 1859 , 1945 , 1960 , 2225 , 2241
<code>__graphics_backend_include_ps:n</code>	1755 , 1766 ,
	1860 , 1907 , 1945 , 1956 , 2209 , 2211
<code>__graphics_backend_include_-svg:n</code> . .	2225 , 2225 , 2241 , 2242 , 2243
<code>__graphics_backend_loaded:n</code> . . .	1736 , 1736 , 1748 , 1750 , 1767 , 1771 ,
	1917 , 1922 , 2025 , 2105 , 2190 , 2246
<code>\l__graphics_backend_name_str</code> .	1860
<code>__graphics_bb_restore:nTF</code>	1811 , 2076 , 2113
<code>__graphics_bb_save:n</code> 1820 , 2084 , 2140	
<code>\l__graphics_decodearray_str</code> . . .	1784 , 1785 ,
	1797 , 1828 , 1834 , 1835 , 1939 , 1974 ,
	1975 , 2013 , 2016 , 2017 , 2041 , 2205
<code>__graphics_extract_bb:n</code>	1932 , 1941 , 2199 , 2207
<code>\l__graphics_final_name_str</code> . .	1890
<code>__graphics_get_pagecount:n</code>	1768 , 2026 , 2247
<code>\l__graphics_internal_box</code>	1846 , 1848 , 1849 , 2081 , 2082 , 2083
<code>\l__graphics_internal_dim</code> 2186 , 2187	
<code>\l__graphics_internal_ior</code>	2115 , 2116 , 2123 , 2142
<code>\l__graphics_interpolate_bool</code> . . .	1786 , 1798 , 1827 , 1836 ,
	1940 , 1976 , 2012 , 2018 , 2042 , 2206
<code>\l__graphics_llx_dim</code>	1760 , 1950 , 2005 , 2119 , 2219
<code>\l__graphics_lly_dim</code>	1761 , 1951 , 2006 , 2120 , 2220
<code>\l__graphics_page_int</code>	1780 , 1802 , 1803 , 1841 ,
	1842 , 1930 , 1972 , 1973 , 1999 , 2000 ,
	2032 , 2047 , 2048 , 2090 , 2091 , 2197
<code>\l__graphics_pagebox_tl</code>	55 , 1781 , 1801 ,
	1843 , 1844 , 1931 , 1970 , 1971 , 2001 ,
	2003 , 2033 , 2056 , 2057 , 2092 , 2198
<code>\l__graphics_pdf_str</code>	1788 , 1789 , 1804 , 1805 , 1829 , 1838
<code>__graphics_read_bb:n</code>	1752 , 1753 , 1924 , 1925 , 2192 , 2193
<code>\g__graphics_track_int</code>	1944 , 1993 , 1994
<code>\l__graphics_urx_dim</code>	1762 , 1848 , 1952 , 2007 , 2082 ,
	2121 , 2125 , 2128 , 2136 , 2221 , 2234
<code>\l__graphics_ury_dim</code>	1763 , 1849 , 1953 , 2008 , 2083 , 2122 ,
	2130 , 2133 , 2137 , 2222 , 2227 , 2235
group commands:	
<code>\group_begin:</code>	172 , 191
<code>\group_end:</code>	180

\group_insert_after:N ... 3259, 3303

H

hbox commands:

\hbox:n 2229, 2352, 2355,
2430, 2436, 2589, 2596, 3074, 3085
\hbox_overlap_right:n 224,
256, 272, 313, 329, 357, 441, 1340, 1535
\hbox_set:Nn .. 1846, 2081, 2422, 2454
\hbox_set:Nw 2405
\hbox_set_end: 2420
\hbox_unpack:N 2541

hook commands:

\hook_gput_code:nnn .. 54, 1738, 1740

I

int commands:

\int_compare:nNnTF
..... 1802, 1841, 1972, 1999,
2047, 2090, 2513, 2614, 2894, 2922
\int_const:Nn
..... 454, 1818, 1912, 1994, 2096
\int_eval:n 474, 484, 630, 639, 652,
654, 658, 671, 2638, 2642, 2872,
2897, 2904, 2917, 3103, 3111, 3116
\int_gincr:N 198,
364, 1604, 1649, 1993, 2264, 2331,
2362, 2439, 2957, 2990, 3003, 3023
\int_gset:Nn 173, 192, 2502, 2788
\int_gset_eq:NN 181, 2363, 2440, 3004
\int_if_exist:NTF 1983
\int_if_odd:nTF 2425
\int_max:nn 2098
\int_new:N 164, 165, 411, 449, 1630,
1944, 2343, 2374, 2376, 3000, 3016
\int_set_eq:NN 169, 188, 2514
\int_step_function:nnnN 656
\int_use:N 366, 397, 583,
592, 740, 768, 817, 823, 824, 878,
879, 888, 912, 1607, 1613, 1620,
1652, 1660, 1803, 1842, 1855, 1913,
1973, 1986, 1998, 2000, 2091, 2099,
2333, 2338, 2366, 2373, 2444, 2545,
2992, 2997, 3007, 3015, 3028, 3039
\int_value:w
..... 2667, 2702, 2843, 2854, 2872
\int_zero:N ... 1780, 1930, 2032, 2197

ior commands:

\ior_close:N 2142
\ior_if_eof:NTF 2116
\ior_map_break: 2138
\ior_open:Nn 2115
\ior_str_map_inline:Nn 2123

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, 3160
\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,
1459, 1466, 1472, 1532, 1537, 1757,
1947, 1985, 1995, 2216, 2231, 2949,
3041, 3103, 3107, 3112, 3117, 3162
__kernel_backend_literal_page:n
..... 99, 99,
109, 153, 153, 2943, 2945, 3122, 3124
__kernel_backend_literal_pdf:n .
..... 88, 88, 98, 150, 150,
152, 264, 321, 1349, 3253, 3264, 3297
__kernel_backend_literal_-
postscript:n 60,
60, 62, 74, 75, 79, 210, 211, 213,
214, 222, 234, 249, 1150, 2616, 2628
__kernel_backend_literal_svg:n .
. 161, 161, 163, 168, 179, 187, 197,
365, 367, 384, 762, 1543, 1718, 1729
__kernel_backend_matrix:n
..... 137, 137, 147, 286, 307, 1449
__kernel_backend_postscript:n ..
..... 63, 63, 65,
500, 1002, 1004, 1006, 1010, 2257,
2308, 2323, 2352, 2358, 2398, 2430,
2437, 2441, 2455, 2483, 2527, 2534,
2540, 2548, 2555, 2589, 2596, 3211
__kernel_backend_scope:n
..... 166, 195, 200, 394, 399,
1030, 1058, 1550, 1595, 1597, 1617,
1657, 1679, 1691, 1693, 1695, 1697,
1699, 1701, 1703, 1705, 1708, 3322
__kernel_backend_scope_begin: ..
82, 82, 119, 119, 155, 155, 166, 166,
208, 232, 247, 263, 280, 306, 320,
337, 350, 1355, 1527, 1545, 1549, 1716
__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,

<code>__pdf_backend_destination:nnnn</code> .	<code>__pdf_backend_link_sf_restore:</code> .
..... 2553 , 2579 , 2383 , 2406 , 2449 , 2509
2721 , 2744 , 3042 , 3064 , 3135 , 3136	<code>__pdf_backend_link_sf_save:</code> . . .
<code>__pdf_backend_destination_-</code> 2383 , 2401 , 2419 , 2500
aux:nnnn	<code>\l__pdf_backend_model_box</code> . 2342 ,
.. 2553 , 2581 , 2584 , 3042 , 3066 , 3069	2422 , 2454 , 2462 , 2473 , 2488 , 2490
<code>__pdf_backend_emc:</code> . . 2646 , 2648 ,	<code>__pdf_backend_objcompresslevel:n</code>
2942 , 2944 , 3121 , 3123 , 3154 , 3155 2863 , 2877 , 2878 , 2880
<code>__pdf_backend_info_gput:nn</code>	<code>__pdf_backend_object_id:n</code>
..... 2259 , 2261 , 2263 , 2266 ,
2758 , 2768 , 2951 , 2953 , 3137 , 3138	2779 , 2797 , 2956 , 2959 , 3139 , 3141
<code>__pdf_backend_link:nw</code> 2383	<code>\g__pdf_backend_object_int</code>
<code>__pdf_backend_link_aux:nw</code> 2383	. . . 2264 , 2331 , 2333 , 2338 , 2362 ,
<code>__pdf_backend_link_begin:n</code>	2363 , 2366 , 2439 , 2440 , 2788 , 2957 ,
..... 3017 , 3018 , 3020 , 3021	2990 , 2992 , 2997 , 3003 , 3004 , 3007
<code>__pdf_backend_link_begin:nnnw</code>	<code>__pdf_backend_object_last:</code>
.. 2676 , 2677 , 2679 , 2680 , 3129 , 3131 2337 , 2337 ,
<code>__pdf_backend_link_begin:nw</code>	2841 , 2841 , 2996 , 2996 , 3139 , 3146
..... 2385 , 2389 , 2390	<code>__pdf_backend_object_new:</code>
<code>__pdf_backend_link_begin_aux:nw</code> 2263 , 2263 ,
..... 2393 , 2395	2779 , 2779 , 2956 , 2956 , 3139 , 3139
<code>__pdf_backend_link_begin_-</code>	<code>__pdf_backend_object_now:nn</code>
goto:nnw 2383 , 2383 ,	2329 , 2329 , 2336 , 2830 , 2830 , 2840 ,
2676 , 2676 , 3017 , 3017 , 3129 , 3129	2988 , 2988 , 2995 , 3139 , 3144 , 3145
<code>__pdf_backend_link_begin_-</code>	<code>\g__pdf_backend_object_prop</code>
user:nnw 2383 , 2388 , 2778 , 2955
2676 , 2678 , 3017 , 3019 , 3129 , 3130	<code>__pdf_backend_object_ref:n</code>
<code>\g__pdf_backend_link_bool</code> 2263 , 2265 , 2266 , 2270 , 2779 , 2796 ,
..... 2378 , 2392 , 2397 , 2412 , 2450	2956 , 2958 , 2959 , 2963 , 3139 , 3140
<code>\g__pdf_backend_link_dict_tl</code>	<code>__pdf_backend_object_write:nn</code>
..... 2375 , 2400 , 2445 2798 , 2807 , 2809 , 2838 , 3139
<code>__pdf_backend_link_end:</code>	<code>__pdf_backend_object_write:nnn</code>
..... 2383 , 2410 , 2267 , 2267 , 2273 , 2798 , 2798 , 2827 ,
2676 , 2691 , 3017 , 3036 , 3129 , 3132	2960 , 2960 , 2965 , 3139 , 3142 , 3143
<code>__pdf_backend_link_end_aux:</code>	<code>__pdf_backend_object_write_-</code>
..... 2383 , 2413 , 2415	array:nn . . . 2267 , 2291 , 2960 , 2966
<code>\g__pdf_backend_link_int</code>	<code>__pdf_backend_object_write_-</code>
..... 2374 , 2440 ,	aux:nnn 2267 , 2269 , 2274 , 2332
2444 , 2545 , 3016 , 3023 , 3028 , 3039	<code>__pdf_backend_object_write_-</code>
<code>__pdf_backend_link_last:</code>	dict:nn 2267 , 2296 , 2960 , 2971
..... 2544 , 2544 ,	<code>__pdf_backend_object_write_-</code>
2700 , 2700 , 3038 , 3038 , 3133 , 3133	fstream:nn . 2267 , 2301 , 2960 , 2976
<code>__pdf_backend_link_margin:n</code>	<code>__pdf_backend_object_write_-</code>
..... 2546 , 2546 ,	fstream:nnn 2304 , 2306
2711 , 2711 , 3040 , 3040 , 3134 , 3134	<code>__pdf_backend_object_write_-</code>
<code>\g__pdf_backend_link_math_bool</code>	stream:nn . . 2267 , 2316 , 2960 , 2978
..... 2377 , 2403 , 2404 , 2407 , 2417	<code>__pdf_backend_object_write_-</code>
<code>__pdf_backend_link_minima:</code>	stream:nnn 2267 , 2319 , 2321
..... 2383 , 2421 , 2452	<code>__pdf_backend_object_write_-</code>
<code>__pdf_backend_link_outerbox:n</code>	stream:nnnn . 2960 , 2977 , 2979 , 2980
..... 2383 , 2423 , 2481	<code>__pdf_backend_pageobject_ref:n</code>
<code>\g__pdf_backend_link_sf_int</code> 2339 , 2339 ,
..... 2376 , 2502 , 2513 , 2514	2852 , 2852 , 2998 , 2998 , 3139 , 3147

_pdf_backend_pagesize_gset:nn . . .	3158, 3158, 3177, 3177, 3184, 3184	pdf.linkmargin	3384
_pdf_backend_pdfmark:n . . .	2256, 2258, 2260, 2262, 2276, 2293, 2298, 2364, 2556, 2600, 2647, 2649	pdf.llx	3387
_pdf_backend_version_major: . . .	2638, 2644, 2644, 2919, 2919, 3111, 3112, 3119, 3119, 3152, 3152	pdf.lly	3387
_pdf_backend_version_major_gset:n	2636, 2636, 2891, 2891, 3109, 3109, 3150, 3150	pdf.originx	3458
_pdf_backend_version_minor: . . .	2642, 2644, 2645, 2919, 2932, 3116, 3117, 3119, 3120, 3152, 3153	pdf.originy	3458
_pdf_backend_version_minor_gset:n	2636, 2640, 2891, 2908, 3109, 3114, 3150, 3151	pdf.outerbox	3700
\l_pdf_breaklink_pdfmark_tl . . .	2379, 2447, 2539	pdf.pdfmark	3700
_pdf_breaklink_postscript:n . . .	2381, 2381, 2431, 2433, 2540	pdf.pdfmark.dict	3700
_pdf_breaklink_usebox:N	2382, 2382, 2432, 2541	pdf.pdfmark.good	3700
_pdf_exp_not_i:nn	2798, 2817, 2822, 2828	pdf.pt.dvi	3380
_pdf_exp_not_ii:nn	2798, 2818, 2823, 2829	pdf.rect	3387
\l_pdf_internal_box	2253	pdf.rect.ht	3380
pdf.baselineskip	3700	pdf.rightboundary	3458
pdf.bordertracking	3458	pdf.save.linkll	3387
pdf.bordertracking.begin	3458	pdf.save.linkur	3387
pdf.bordertracking.continue	3458	pdf.save.ll	3387
pdf.bordertracking.end	3458	pdf.save.ur	3387
pdf.bordertracking.endpage	3458	pdf.tmpa	3423
pdf.breaklink	3596	pdf.tmpb	3423
pdf.breaklink.write	3596	pdf.tmpc	3423
pdf.brokenlink.dict	3458	pdf.tmpd	3423
pdf.brokenlink.rect	3458	pdf.urx	3387
pdf.brokenlink.skip	3458	pdf.ury	3387
pdf.count	3596	pdfmanagement commands:	
pdf.currentrect	3596	_pdfmanagement_add:nnn	797, 3238, 3249, 3290, 3293
pdf.cvs	3380	_pdfmanagement_if_active_p:	792, 793, 3231, 3232, 3307, 3308
pdf.dest.anchor	3423	peek commands:	
pdf.dest.point	3423	_peek_meaning:NnTF	2158, 2161
pdf.dest.x	3423	_peek_remove_spaces:n	2156
pdf.dest.y	3423	prg commands:	
pdf.dest2device	3423	_prg_replicate:nn	177, 628, 649, 659, 860
pdf.dev.x	3423	prop commands:	
pdf.dev.y	3423	_prop_gput:Nnn	586, 827
pdf.dvi.pt	3380	_prop_if_in:NnTF	563
pdf.globaldict	3377	_prop_item:Nn	566
pdf.leftboundary	3458	_prop_new:N	544, 2778, 2955
pdf.linkdp.pad	3384	\ProvidesExplFile	2
pdf.linkht.pad	3384		
		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, 2186, 2189, 2694, 2719, 2742, 2756, 2872, 2889, 2897, 2904, 2917

scan internal commands:	
\ s _ c o l o r _ s t o p	639, 640, 644, 648, 661, 664, 668, 672, 686, 861, 890, 894, 1044, 1046
\ s _ g r a p h i c s _ s t o p	1817, 1851, 2151, 2166, 2173, 2177, 2179, 2181, 2236, 2244
separation	3374
seq commands:	
\ s e q _ s e t _ f r o m _ c l i s t : N n	1749, 1773, 1919, 2107
shipout commands:	
\ l _ s h i p o u t _ b o x	2523, 2525, 2533
skip commands:	
\ s k i p _ h o r i z o n t a l : n	226, 274, 331
str commands:	
\ c _ h a s h _ s t r	397, 1613, 1620, 1660
\ c _ p e r c e n t _ s t r	1064, 1065, 1066
\ s t r _ c a s e : n n	866, 2280, 2811
\ s t r _ c a s e : n n T F	2560, 2730, 3049
\ s t r _ c o n v e r t _ p d f n a m e : n	587, 607, 816
\ s t r _ i f _ e m p t y : N T F	1788, 1804
\ s t r _ i f _ e m p t y _ p : N	1829
\ s t r _ i f _ e q : n n T F	536, 766, 3285
\ s t r _ n e w : N	1862, 1863, 1864
\ s t r _ t a i l : N	1876, 1902
sys commands:	
\ s y s _ i f _ s h e l l : T F	1860
\ s y s _ s h e l l _ n o w : n	1887
T	
T _E X and L ^A T _E X 2 _ε commands:	
\ @ i f l @ t @ r	49, 51
\ @ m a k e c o l @ h o o k	2519, 2521
\ s p e c i a l	2
tex commands:	
\ t e x _ a f t e r a s s i g n m e n t : D	2185
\ t e x _ b a s e l i n e s k i p : D	2494
\ t e x _ e n d i n p u t : D	44
\ t e x _ g l o b a l : D	2865, 2882, 2896, 2903, 2910
\ t e x _ i m m e d i a t e : D	1824, 2801, 2804, 2833, 2836
\ t e x _ l u a t e x v e r s i o n : D	2894, 2922
\ t e x _ p a g e h e i g h t : D	3180
\ t e x _ p a g e w i d t h : D	3179
\ t e x _ p d f a n n o t : D	2658
\ t e x _ p d f c a t a l o g : D	2764
\ t e x _ p d f c o l o r s t a c k : D	472, 482
\ t e x _ p d f c o l o r s t a c k i n i t : D	460
\ t e x _ p d f c o m p r e s s l e v e l : D	2870
\ t e x _ p d f d e s t : D	2727, 2750
\ t e x _ p d f e n d l i n k : D	2697
\ t e x _ p d f e x t e n s i o n : D	91, 102, 112, 122, 131, 140, 469, 479, 2655, 2683, 2694, 2724, 2747, 2761, 2771, 2782, 2801, 2833
\ t e x _ p d f f e e d b a c k : D	457, 2669, 2704, 2790, 2845, 2856
\ t e x _ p d f i n f o : D	2774
\ t e x _ p d f l a s t a n n o t : D	2672
\ t e x _ p d f l a s t l i n k : D	2707
\ t e x _ p d f l a s t o b j : D	2793, 2848
\ t e x _ p d f l a s t x i m a g e : D	1819, 1847
\ t e x _ p d f l a s t x i m a g e p a g e s : D	1913
\ t e x _ p d f l i n k m a r g i n : D	2717
\ t e x _ p d f l i t e r a l : D	94, 105, 115
\ t e x _ p d f m a j o r v e r s i o n : D	2901, 2903, 2927, 2928
\ t e x _ p d f m i n o r v e r s i o n : D	2915, 2939
\ t e x _ p d f o b j : D	2785, 2804, 2836
\ t e x _ p d f o b j c o m p r e s s l e v e l : D	2887
\ t e x _ p d f p a g e r e f : D	2859
\ t e x _ p d f r e f x i m a g e : D	1847, 1854
\ t e x _ p d f r e s t o r e : D	134
\ t e x _ p d f s a v e : D	125
\ t e x _ p d f s e t m a t r i x : D	143
\ t e x _ p d f s t a r t l i n k : D	2686
\ t e x _ p d f v a r i a b l e : D	2714, 2867, 2884, 2896, 2912, 2923, 2936
\ t e x _ p d f x i m a g e : D	1824, 1911
\ t e x _ s p a c e f a c t o r : D	2505, 2514
\ t e x _ s p e c i a l : D	46
\ t e x _ t h e : D	1819, 2923, 2928, 2934
\ t e x _ v s s : D	2590, 2597, 3079, 3098
\ t e x _ X e T e X p d f f i l e : D	2043, 2089
\ t e x _ X e T e X p d f p a g e c o u n t : D	2099
\ t e x _ X e T e X p i c f i l e : D	2034
TeXcolorseparation	3374
\ t e x t w i d t h	2489
tl commands:	
\ c _ s p a c e _ t l	288, 293, 296, 549, 554, 592, 695, 769, 979, 1589, 1759, 1760, 1761, 1762, 1949, 1950, 1951, 1952, 2000, 2003, 2005, 2006, 2007, 2008, 2069, 2091, 2218, 2219, 2220, 2221, 2445, 2674, 2709, 2850, 2861, 3007, 3029
\ t l _ c l e a r : N	1781, 1797, 1931, 1939, 2033, 2041, 2198, 2205
\ t l _ g c l e a r : N	1627, 1663
\ t l _ g s e t : N n	1586, 2400
\ t l _ i f _ b l a n k : n T F	462, 547, 643, 660, 667, 685, 811, 893, 2068, 2154
\ t l _ i f _ e m p t y : N T F	1589, 1784, 1834, 1843, 1970, 1974, 2001, 2016, 2056
\ t l _ i f _ e m p t y : n T F	905, 1683

