

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}{2026-08-02}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2026-08-02}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2026-08-02}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*hitex>
16   {l3backend-hitex.def}{2026-08-02}{ }
17   {L3 backend support: HiTeX}
18 </hitex>
19 <*luatex>
20   {l3backend-luatex.def}{2026-08-02}{ }
21   {L3 backend support: PDF output (LuaTeX)}
22 </luatex>
23 <*pdftex>
24   {l3backend-pdftex.def}{2026-08-02}{ }
25   {L3 backend support: PDF output (pdfTeX)}
26 </pdftex>
27 <*xetex>
28   {l3backend-xetex.def}{2026-08-02}{ }
29   {L3 backend support: XeTeX}
30 </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.

```
31 \cs_if_exist:NTF \_kernel\_dependency\_version\_check:nn
32 {
33   \_kernel\_dependency\_version\_check:nn {2023-10-10}
34 <dvipdfmx>   {l3backend-dvipdfmx.def}
35 <dvips>      {l3backend-dvips.def}
```

```

36 <dvips>      {l3backend-dvips.def}
37 <hitex>      {l3backend-hitex.def}
38 <luatex>     {l3backend-luatex.def}
39 <pdftex>     {l3backend-pdftex.def}
40 <xetex>      {l3backend-xetex.def}
41 }
42 {
43   \cs_if_exist_use:cF { @latex@error } { \errmessage }
44   {
45     Mismatched-LaTeX-support-files-detected. \MessageBreak
46     Loading-aborted!
47   }
48   { \use:c { @ehd } }
49   \tex_endinput:D
50 }

```

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 behavior so a wrapper is provided.
`__kernel_backend_literal:n`

```

51 \cs_new_eq:NN __kernel_backend_literal:e \tex_special:D
52 \cs_new_protected:Npn __kernel_backend_literal:n #1
53 { __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,

```

54 \cs_if_exist:NTF \ifl@t@r
55 {
56   \ifl@t@r \fmtversion { 2020-10-01 }
57   {
58     \cs_new_protected:Npn __kernel_backend_first_shipout:n #1
59     { \hook_gput_code:nnn { shipout / firstpage } { l3backend } {#1} }
60   }
61   { \cs_new_eq:NN __kernel_backend_first_shipout:n \AtBeginDvi }
62 }
63 { \cs_new_eq:NN __kernel_backend_first_shipout:n \use:n }

```

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

1.1 dvips backend

```

64 <*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.
`__kernel_backend_literal_postscript:e`

```

65 \cs_new_protected:Npn \__kernel_backend_literal_postscript:n #1
66 { \__kernel_backend_literal:n { ps:: #1 } }
67 \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 ").

```

68 \cs_new_protected:Npn \__kernel_backend_postscript:n #1
69 { \__kernel_backend_literal:n { ps: SDict ~ begin ~ #1 ~ end } }
70 \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.

```

71 \bool_if:NT \g__kernel_backend_header_bool
72 {
73   \__kernel_backend_first_shipout:n
74   { \__kernel_backend_literal:n { header = l3backend-dvips.pro } }
75 }

```

__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.

```

76 \cs_new_protected:Npn \__kernel_backend_align_begin:
77 {
78   \__kernel_backend_literal:n { ps::[begin] }
79   \__kernel_backend_literal_postscript:n { currentpoint }
80   \__kernel_backend_literal_postscript:n { currentpoint-translate }
81 }
82 \cs_new_protected:Npn \__kernel_backend_align_end:
83 {
84   \__kernel_backend_literal_postscript:n { neg-exch-neg-exch-translate }
85   \__kernel_backend_literal:n { ps::[end] }
86 }

```

(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.

```

87 \cs_new_protected:Npn \__kernel_backend_scope_begin:
88 { \__kernel_backend_literal:n { ps:gsave } }
89 \cs_new_protected:Npn \__kernel_backend_scope_end:
90 { \__kernel_backend_literal:n { ps:grestore } }

```

(End of definition for __kernel_backend_scope_begin: and __kernel_backend_scope_end:.)

```

91 </dvips>

```

1.2 LuaTeX and pdfTeX backends

92 `<*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.

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. The engines may optimize these transformations, so there is not a simple relationship between the number of `\__kernel_backend_literal_pdf:n` and the resulting pairs of cm operations in the PDF (cf. XeTeX).

```

93 \cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1
94 {
95   <*luatex>
96   \tex_pdfextension:D literal
97   </luatex>
98   <*pdftex>
99   \tex_pdfliteral:D
100  </pdftex>
101   { \exp_not:n {#1} }
102 }
103 \cs_new_protected:Npn \__kernel_backend_literal_pdf:e #1
104 {
105   <*luatex>
106   \tex_pdfextension:D literal
107   </luatex>
108   <*pdftex>
109   \tex_pdfliteral:D
110   </pdftex>
111   {#1}
112 }
```

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

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

```

113 \cs_new_protected:Npn \__kernel_backend_literal_page:n #1
114 {
115   <*luatex>
116   \tex_pdfextension:D literal ~
117   </luatex>
118   <*pdftex>
119   \tex_pdfliteral:D
120   </pdftex>
121   page { \exp_not:n {#1} }
122 }
123 \cs_new_protected:Npn \__kernel_backend_literal_page:e #1
124 {
125   <*luatex>
126   \tex_pdfextension:D literal ~
127   </luatex>
128   <*pdftex>
129   \tex_pdfliteral:D
130   </pdftex>
```

```

131         page {#1}
132     }

```

(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:
133 \cs_new_protected:Npn \_kernel_backend_scope_begin:
134 {
135   <*luatex>
136   \tex_pdfextension:D save \scan_stop:
137   </luatex>
138   <*pdftex>
139   \tex_pdfsave:D
140   </pdftex>
141 }
142 \cs_new_protected:Npn \_kernel_backend_scope_end:
143 {
144   <*luatex>
145   \tex_pdfextension:D restore \scan_stop:
146   </luatex>
147   <*pdftex>
148   \tex_pdfrestore:D
149   </pdftex>
150 }

```

(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.

`\_kernel_backend_matrix:e`

```

151 \cs_new_protected:Npn \_kernel_backend_matrix:n #1
152 {
153   <*luatex>
154   \tex_pdfextension:D setmatrix
155   </luatex>
156   <*pdftex>
157   \tex_pdfsetmatrix:D
158   </pdftex>
159   { \exp_not:n {#1} }
160 }
161 \cs_new_protected:Npn \_kernel_backend_matrix:e #1
162 {
163   <*luatex>
164   \tex_pdfextension:D setmatrix
165   </luatex>
166   <*pdftex>
167   \tex_pdfsetmatrix:D
168   </pdftex>
169   {#1}
170 }

```

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

```

171 </luatex | pdftex>

```

1.3 dvipdfmx backend

172 $\langle *dvipdfmx | xetex \rangle$

The `dvipdfmx` shares code with the PDF mode one (using the common section to this file) but also with `XYTeX`. 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 `XYTeX` as required. Undocumented but equivalent to pdfTeX’s `literal` keyword. It’s similar to be not the same as the documented `contents` keyword as that adds a `q/Q` pair. Like the pdfTeX version, the positioning here is kept in track by the engine; there is though no optimization of the insertion of `cm` pairs, so there will always be one pair for each `\__kernel_backend_literal_pdf:n` used.

173 `\cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1`
 174 `{ \__kernel_backend_literal:n { pdf:literal~ #1 } }`
 175 `\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!

176 `\cs_new_protected:Npn \__kernel_backend_literal_page:n #1`
 177 `{ \__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 `xdvipdmx` (`x:`) as these are well-tested “in the wild”.

178 `\cs_new_protected:Npn \__kernel_backend_scope_begin:`
 179 `{ \__kernel_backend_literal:n { x:gsave } }`
 180 `\cs_new_protected:Npn \__kernel_backend_scope_end:`
 181 `{ \__kernel_backend_literal:n { x:grestore } }`

(End of definition for `\__kernel_backend_scope_begin:` and `\__kernel_backend_scope_end:.`)

182 $\langle /dvipdfmx | xetex \rangle$

1.4 dvisvgm backend

183 $\langle *dvisvgm \rangle$

`\__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.

`\__kernel_backend_literal_svg:e`

184 `\cs_new_protected:Npn \__kernel_backend_literal_svg:n #1`
 185 `{ \__kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }`
 186 `\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.

`\l__kernel_backend_scope_int`

187 `\int_new:N \g__kernel_backend_scope_int`
 188 `\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:
__kernel_backend_scope_end:
__kernel_backend_scope_begin:n
__kernel_backend_scope_begin:e
__kernel_backend_scope:n
__kernel_backend_scope:e`

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.

```

189 \cs_new_protected:Npn \__kernel_backend_scope_begin:
190 {
191   \__kernel_backend_literal_svg:n { <g> }
192   \int_set_eq:NN
193     \l__kernel_backend_scope_int
194     \g__kernel_backend_scope_int
195   \group_begin:
196     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
197 }
198 \cs_new_protected:Npn \__kernel_backend_scope_end:
199 {
200   \prg_replicate:nn
201     { \g__kernel_backend_scope_int }
202     { \__kernel_backend_literal_svg:n { </g> } }
203   \group_end:
204   \int_gset_eq:NN
205     \g__kernel_backend_scope_int
206     \l__kernel_backend_scope_int
207 }
208 \cs_new_protected:Npn \__kernel_backend_scope_begin:n #1
209 {
210   \__kernel_backend_literal_svg:n { <g ~ #1 > }
211   \int_set_eq:NN
212     \l__kernel_backend_scope_int
213     \g__kernel_backend_scope_int
214   \group_begin:
215     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
216 }
217 \cs_generate_variant:Nn \__kernel_backend_scope_begin:n { e }
218 \cs_new_protected:Npn \__kernel_backend_scope:n #1
219 {
220   \__kernel_backend_literal_svg:n { <g ~ #1 > }
221   \int_gincr:N \g__kernel_backend_scope_int
222 }
223 \cs_generate_variant:Nn \__kernel_backend_scope:n { e }

```

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

```

224 </dvisvgm>
225 </package>

```

2 l3backend-box implementation

```

226 *package>
227 @@=box>

```

2.1 dvips backend

```

228 *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.

```

229 \cs_new_protected:Npn \_box_backend_clip:N #1
230 {
231   \_kernel_backend_scope_begin:
232   \_kernel_backend_align_begin:
233   \_kernel_backend_literal_postscript:n { matrix~currentmatrix }
234   \_kernel_backend_literal_postscript:n
235     { Resolution~72~div~VResolution~72~div~scale }
236   \_kernel_backend_literal_postscript:n { DVImag~dup~scale }
237   \_kernel_backend_literal_postscript:e
238     {
239       0 ~
240       \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
241       \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
242       \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
243       rectclip
244     }
245   \_kernel_backend_literal_postscript:n { setmatrix }
246   \_kernel_backend_align_end:
247   \hbox_overlap_right:n { \box_use:N #1 }
248   \_kernel_backend_scope_end:
249   \skip_horizontal:n { \box_wd:N #1 }
250 }

```

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

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

`\_box_backend_rotate_aux:Nn`

```

251 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
252 { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
253 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
254 {
255   \_kernel_backend_scope_begin:
256   \_kernel_backend_align_begin:
257   \_kernel_backend_literal_postscript:e
258     {
259       \fp_compare:nNnTF {#2} = \c_zero_fp
260       { 0 }
261       { \fp_eval:n { round ( -(#2) , 5 ) } } ~
262       rotate
263     }
264   \_kernel_backend_align_end:
265   \box_use:N #1
266   \_kernel_backend_scope_end:
267 }

```

(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.

```

268 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
269 {
270   \__kernel_backend_scope_begin:
271   \__kernel_backend_align_begin:
272   \__kernel_backend_literal_postscript:e
273   {
274     \fp_eval:n { round ( #2 , 5 ) } ~
275     \fp_eval:n { round ( #3 , 5 ) } ~
276     scale
277   }
278   \__kernel_backend_align_end:
279   \hbox_overlap_right:n { \box_use:N #1 }
280   \__kernel_backend_scope_end:
281 }

```

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

```

282 </dvips>

```

2.2 LuaTeX and pdfTeX backends

```

283 <*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.

```

284 \cs_new_protected:Npn \__box_backend_clip:N #1
285 {
286   \__kernel_backend_scope_begin:
287   \__kernel_backend_literal_pdf:e
288   {
289     0~
290     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
291     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
292     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
293     re~W~n
294   }
295   \hbox_overlap_right:n { \box_use:N #1 }
296   \__kernel_backend_scope_end:
297   \skip_horizontal:n { \box_wd:N #1 }
298 }

```

(End of definition for `\__box_backend_clip:N`.)

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

```

\__box_backend_rotate_aux:Nn
  \l__box_backend_cos_fp
  \l__box_backend_sin_fp

```

```

299 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2

```

```

300 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
301 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
302 {
303   \__kernel_backend_scope_begin:
304   \box_set_wd:Nn #1 { Opt }
305   \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
306   \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
307     { \fp_zero:N \l__box_backend_cos_fp }
308   \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
309   \__kernel_backend_matrix:e
310   {
311     \fp_use:N \l__box_backend_cos_fp \c_space_tl
312     \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp
313       { 0~0 }
314       {
315         \fp_use:N \l__box_backend_sin_fp
316         \c_space_tl
317         \fp_eval:n { -\l__box_backend_sin_fp }
318       }
319     \c_space_tl
320     \fp_use:N \l__box_backend_cos_fp
321   }
322   \box_use:N #1
323   \__kernel_backend_scope_end:
324 }
325 \fp_new:N \l__box_backend_cos_fp
326 \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.

```

327 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
328 {
329   \__kernel_backend_scope_begin:
330   \__kernel_backend_matrix:e
331   {
332     \fp_eval:n { round ( #2 , 5 ) } ~
333     0~0~
334     \fp_eval:n { round ( #3 , 5 ) }
335   }
336   \hbox_overlap_right:n { \box_use:N #1 }
337   \__kernel_backend_scope_end:
338 }

```

(End of definition for __box_backend_scale:Nnn.)

339 </luatex | pdftex>

2.3 dvipdfmx/X_YTeX backend

340 <*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.

```

341 \cs_new_protected:Npn \__box_backend_clip:N #1

```

```

342 {
343   \__kernel_backend_scope_begin:
344   \__kernel_backend_literal_pdf:e
345   {
346     0~
347     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
348     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
349     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
350     re~W~n
351   }
352   \hbox_overlap_right:n { \box_use:N #1 }
353   \__kernel_backend_scope_end:
354   \skip_horizontal:n { \box_wd:N #1 }
355 }

```

(End of definition for __box_backend_clip:N.)

__box_backend_rotate:Nn
__box_backend_rotate_aux:Nn

Rotating in dvipdmtx/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.

```

356 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
357 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
358 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
359 {
360   \__kernel_backend_scope_begin:
361   \__kernel_backend_literal:e
362   {
363     x:rotate~
364     \fp_compare:nNnTF {#2} = \c_zero_fp
365     { 0 }
366     { \fp_eval:n { round ( #2 , 5 ) } }
367   }
368   \box_use:N #1
369   \__kernel_backend_scope_end:
370 }

```

(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.

```

371 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
372 {
373   \__kernel_backend_scope_begin:
374   \__kernel_backend_literal:e
375   {
376     x:scale~
377     \fp_eval:n { round ( #2 , 5 ) } ~
378     \fp_eval:n { round ( #3 , 5 ) }
379   }
380   \hbox_overlap_right:n { \box_use:N #1 }
381   \__kernel_backend_scope_end:
382 }

```

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

383 `</dviptfm | xetex>`

2.4 dvisvgm backend

384 `<*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 $\text{\TeX}$ box and keep the reference point the same!

```

385 \cs_new_protected:Npn \__box_backend_clip:N #1
386 {
387   \int_gincr:N \g__kernel_clip_path_int
388   \__kernel_backend_literal_svg:e
389   { < clipPath-id = " l3cp \int_use:N \g__kernel_clip_path_int " > }
390   \__kernel_backend_literal_svg:e
391   {
392     <
393       path ~ d =
394         "
395           M ~ 0 ~
396             \dim_to_decimal:n { -\box_dp:N #1 } ~
397             L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
398             \dim_to_decimal:n { -\box_dp:N #1 } ~
399             L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
400             \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
401             L ~ 0 ~
402             \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
403             Z
404           "
405         />
406     }
407   \__kernel_backend_literal_svg:n
408   { < /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 $\text{\TeX}$ 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 $\text{\TeX}$ box.

```

409 \__kernel_backend_scope_begin:n
410 {
411   transform =
412     "
413       translate ( { ?x } , { ?y } ) ~
414       scale ( 1 , -1 )
415     "
416   }
417 \__kernel_backend_scope:e

```

```

418     {
419         clip-path =
420             "url ( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int ) "
421     }
422     \__kernel_backend_scope:n
423     {
424         transform =
425             "
426             scale ( -1 , 1 ) ~
427             translate ( { ?x } , { ?y } ) ~
428             scale ( -1 , -1 )
429             "
430     }
431     \box_use:N #1
432     \__kernel_backend_scope_end:
433 }
434 \int_new:N \g__kernel_clip_path_int

```

(End of definition for `\__box_backend_clip:N` and `\g__kernel_clip_path_int`.)

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

```

435 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
436 {
437     \__kernel_backend_scope_begin:e
438     {
439         transform =
440             "
441             rotate
442             ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
443             "
444     }
445     \box_use:N #1
446     \__kernel_backend_scope_end:
447 }

```

(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).

```

448 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
449 {
450     \__kernel_backend_scope_begin:e
451     {
452         transform =
453             "
454             translate ( { ?x } , { ?y } ) ~
455             scale
456             (
457                 \fp_eval:n { round ( -#2 , 5 ) } ,
458                 \fp_eval:n { round ( -#3 , 5 ) }
459             ) ~

```

```

460         translate ( { ?x } , { ?y } ) ~
461         scale ( -1 )
462     "
463 }
464 \hbox_overlap_right:n { \box_use:N #1 }
465 \__kernel_backend_scope_end:
466 }

```

(End of definition for __box_backend_scale:Nnn.)

```
467 </dvisvgm>
```

```
468 </package>
```

3 I3backend-color implementation

```
469 <*package>
```

```
470 <@@=color>
```

Color support is split into parts: collecting data from L^AT_EX 2_ε, the color stack, general color, separations, and color for drawings. We have different approaches in each backend, and have some choices to make about dvipdfmx/X_YL_AT_EX in particular. Whilst it is in some ways convenient to use the same approach in multiple backends, the fact that dvipdfmx/X_YL_AT_EX is PDF-based means it (largely) sticks closer to direct PDF output.

3.1 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/X_YL_AT_EX 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

```
471 <*luatex | pdftex>
```

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

```
472 \int_new:N \l__color_backend_stack_int
```

(End of definition for \l__color_backend_stack_int.)

```
473 </luatex | pdftex>
```

3.1.2 Lua_TE_X and pdf_TE_X

```
474 <*luatex | pdftex>
```

__kernel_color_backend_stack_init:Nnn

```

475 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
476 {
477     \int_const:Nn #1
478     {
479 <*luatex>
480     \tex_pdffeedback:D colorstackinit ~
481 </luatex>

```

```

482 <*pdftex>
483     \tex_pdfcolorstackinit:D
484 </pdftex>
485     \tl_if_blank:nF {#2} { #2 ~ }
486     {#3}
487 }
488 }

```

(End of definition for __kernel_color_backend_stack_init:Nnn.)

```

\__kernel_color_backend_stack_push:nn
\__kernel_color_backend_stack_pop:n
489 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
490 {
491 <*luatex>
492     \tex_pdfextension:D colorstack ~
493 </luatex>
494 <*pdftex>
495     \tex_pdfcolorstack:D
496 </pdftex>
497     \int_eval:n {#1} ~ push ~ {#2}
498 }
499 \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
500 {
501 <*luatex>
502     \tex_pdfextension:D colorstack ~
503 </luatex>
504 <*pdftex>
505     \tex_pdfcolorstack:D
506 </pdftex>
507     \int_eval:n {#1} ~ pop \scan_stop:
508 }

```

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

```

509 </luatex | pdftex>

```

3.2 General color

3.2.1 dvips-style

```

510 <*dvips | dvisvgm>
511 <*dvips>
512 \cs_new:Npn \__color_backend_raw_cmyk:n #1 { #1 ~ setcmykcolor }
513 \cs_new:Npn \__color_backend_raw_gray:n #1 { #1 setgraycolor }
514 \cs_new:Npn \__color_backend_raw_rgb:n #1 { #1 ~ setrgbcolor }
515 </dvips>
516 <*dvisvgm>
517 \cs_new:Npn \__color_backend_raw_cmyk:n #1 { cmyk ~ #1 }
518 \cs_new:Npn \__color_backend_raw_gray:n #1 { gray ~ #1 }
519 \cs_new:Npn \__color_backend_raw_rgb:n #1 { rgb ~ #1 }
520 </dvisvgm>

```

(End of definition for __color_backend_raw_cmyk:n, __color_backend_raw_gray:n, and __color_backend_raw_rgb:n.)

```

\__color_backend_current_cmyk:n
\__color_backend_current_gray:n
\__color_backend_current_rgb:n
521 \cs_new:Npn \__color_backend_current_cmyk:n #1 { cmyk ~ #1 }
522 \cs_new:Npn \__color_backend_current_gray:n #1 { gray ~ #1 }
523 \cs_new:Npn \__color_backend_current_rgb:n #1 { rgb ~ #1 }

(End of definition for \__color_backend_current_cmyk:n, \__color_backend_current_gray:n, and \_
_color_backend_current_rgb:n.)

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_select:nn
\__color_backend_reset:
524 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
525 { \__color_backend_select:nn { cmyk } {#1} }
526 \cs_new_protected:Npn \__color_backend_select_gray:n #1
527 { \__color_backend_select:nn { gray } {#1} }
528 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
529 { \__color_backend_select:nn { rgb } {#1} }
530 \cs_new_protected:Npn \__color_backend_select:nn #1#2
531 {
532   \__kernel_backend_literal:e
533   { color~push~ \use:c { __color_backend_current_ #1 :n } {#2} }
534   <*dvips>
535   \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
536   </dvips>
537   }
538 \cs_new_protected:Npn \__color_backend_reset:
539 { \__kernel_backend_literal:n { color-pop } }

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

540 </dvips | dvisvgm>

```

3.2.2 LuaTeX and pdfTeX

```

541 <*luatex | pdftex>

\l__color_backend_fill_tl
\l__color_backend_stroke_tl
542 \tl_new:N \l__color_backend_fill_tl
543 \tl_new:N \l__color_backend_stroke_tl
544 \tl_set:Nn \l__color_backend_fill_tl { 0 ~ g }
545 \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_raw_cmyk:n
\__color_backend_raw_gray:n
\__color_backend_raw_rgb:n
546 \cs_new:Npn \__color_backend_raw_cmyk:n #1 { #1 ~ k \c_space_tl #1 ~ K }
547 \cs_new:Npn \__color_backend_raw_gray:n #1 { #1 ~ g \c_space_tl #1 ~ G }
548 \cs_new:Npn \__color_backend_raw_rgb:n #1 { #1 ~ rg \c_space_tl #1 ~ RG }

(End of definition for \__color_backend_raw_cmyk:n, \__color_backend_raw_gray:n, and \__color_
backend_raw_rgb:n.)

```


`\_color_backend_select_cmyk:n`
`\_color_backend_select_gray:n`
`\_color_backend_select_rgb:n`
`\_color_backend_select:nn`
`\_color_backend_select:w`
`\_color_backend_reset:`

Store the values then pass to the stack; just a bit of expansion to get them in the correct form.

```

549 \cs_new_protected:Npn \_color_backend_select_cmyk:n #1
550 { \_color_backend_select:nn { cmyk } {#1} }
551 \cs_new_protected:Npn \_color_backend_select_gray:n #1
552 { \_color_backend_select:nn { gray } {#1} }
553 \cs_new_protected:Npn \_color_backend_select_rgb:n #1
554 { \_color_backend_select:nn { rgb } {#1} }
555 \cs_new_protected:Npn \_color_backend_select:nn #1#2
556 {
557   \exp_after:wN \exp_after:wN \exp_after:wN \_color_backend_select:w
558   \cs:w \_color_backend_raw_ #1 :n \cs_end: {#2} \q_stop
559 }
560 \cs_new_protected:Npn \_color_backend_select:w #1 \c_space_tl #2 \q_stop
561 {
562   \tl_set:Nn \l__color_backend_fill_tl {#1}
563   \tl_set:Nn \l__color_backend_stroke_tl {#2}
564   \_kernel_color_backend_stack_push:nn \l__color_backend_stack_int { #1 ~ #2 }
565 }
566 \cs_new_protected:Npn \_color_backend_reset:
567 { \_kernel_color_backend_stack_pop:n \l__color_backend_stack_int }

```

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

568 `</luatex | pdftex>`

3.2.3 dvipmfix/X_YTeX

These backends have the most possible approaches: it recognizes 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.

569 `<*dvipdfmx | xetex>`

`\_color_backend_raw_cmyk:n` PDF literals: not used for selection due to the difference in stack approach.

```

570 \cs_new:Npn \_color_backend_raw_cmyk:n #1 { #1 ~ k ~ #1 ~ K }
571 \cs_new:Npn \_color_backend_raw_gray:n #1 { #1 ~ g ~ #1 ~ G }
572 \cs_new:Npn \_color_backend_raw_rgb:n #1 { #1 ~ rg ~ #1 ~ RG }

```

(End of definition for `\_color_backend_raw_cmyk:n`, `\_color_backend_raw_gray:n`, and `\_color_backend_raw_rgb:n`.)

`\_color_backend_current_cmyk:n` The current color has to match how the classical method works.

```

573 \cs_new:Npn \_color_backend_current_cmyk:n #1 { cmyk ~ #1 }
574 \cs_new:Npn \_color_backend_current_gray:n #1 { gray ~ #1 }
575 \cs_new:Npn \_color_backend_current_rgb:n #1 { rgb ~ #1 }

```

(End of definition for `\_color_backend_current_cmyk:n`, `\_color_backend_current_gray:n`, and `\_color_backend_current_rgb:n`.)

```

\__color_backend_select:n
  \_color_backend_select_cmyk:n
  \_color_backend_select_gray:n
  \_color_backend_select_rgb:n
\__color_backend_reset:
576 \cs_new_protected:Npn \__color_backend_select:n #1
577 { \__kernel_backend_literal:n { pdf : bc ~ [ #1 ] } }
578 \cs_new_eq:NN \__color_backend_select_cmyk:n \__color_backend_select:n
579 \cs_new_eq:NN \__color_backend_select_gray:n \__color_backend_select:n
580 \cs_new_eq:NN \__color_backend_select_rgb:n \__color_backend_select:n
581 \cs_new_protected:Npn \__color_backend_reset:
582 { \__kernel_backend_literal:n { pdf : ec } }

```

Using the single stack is relatively easy as there is only one route.

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

```
583 </dvipdfmx | xetex>
```

3.3 Separations

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

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

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

```
\g_color_backend_colorant_prop
```

```
585 \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
```

```

586 \cs_new:Npe \__color_backend_devicen_colorants:n #1
587 {
588   \exp_not:N \tl_if_blank:nF {#1}
589   {
590     \c_space_tl
591     << ~
592     /Colorants ~
593     << ~
594     \exp_not:N \__color_backend_devicen_colorants:w #1 ~
595     \exp_not:N \q_recursion_tail \c_space_tl
596     \exp_not:N \q_recursion_stop
597     >> ~
598     >>
599   }
600 }
601 \cs_new:Npn \__color_backend_devicen_colorants:w #1 ~
602 {
603   \quark_if_recursion_tail_stop:n {#1}
604   \prop_if_in:NnT \g_color_backend_colorant_prop {#1}
605   {
606     #1 ~
607     \prop_item:Nn \g_color_backend_colorant_prop {#1} ~
608   }
609   \__color_backend_devicen_colorants:w
610 }

```

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

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

```

612 < *dvips>

\__color_backend_raw_separation:nn
\__color_backend_raw_devicen:nn
\__color_backend_raw_iccbased:nn
Simple for Separation and DeviceN, no support for ICCbased.
613 \cs_new:Npn \__color_backend_raw_separation:nn #1#2 { separation ~ #1 ~ #2 }
614 \cs_new_eq:NN \__color_backend_raw_devicen:nn \__color_backend_raw_separation:nn
615 \cs_new:Npn \__color_backend_raw_iccbased:nn #1#2 { }

(End of definition for \__color_backend_raw_separation:nn, \__color_backend_raw_devicen:nn, and
\__color_backend_raw_iccbased:nn.)

\__color_backend_current_separation:nn
\__color_backend_current_devicen:nn
\__color_backend_current_iccbased:nn
616 \cs_new:Npn \__color_backend_current_separation:nn #1#2 { ~ #1 ~ #2 }
617 \cs_new_eq:NN \__color_backend_current_devicen:nn \__color_backend_current_separation:nn
618 \cs_new_eq:NN \__color_backend_current_iccbased:nn \__color_backend_current_separation:nn

(End of definition for \__color_backend_current_separation:nn, \__color_backend_current_devicen:nn,
and \__color_backend_current_iccbased:nn.)

\__color_backend_select_separation:nn
\__color_backend_select_devicen:nn
\__color_backend_select_iccbased:nn
619 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
620 { \__color_backend_select:nn { separation } {#1} }
621 \cs_new_eq:NN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn
622 \cs_new_protected:Npn \__color_backend_select_iccbased:nn #1#2 { }

(End of definition for \__color_backend_select_separation:nn, \__color_backend_select_devicen:nn,
and \__color_backend_select_iccbased:nn.)

\__color_backend_separation_init:nnnnn
\__color_backend_separation_init:neemn
\__color_backend_separation_init_aux:nnnnnn
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:nnnn
\__color_backend_separation_init:w
\__color_backend_separation_init:n
\__color_backend_separation_init:nw
\__color_backend_separation_init_CIELAB:nnn
Initializing 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.
623 \cs_new_protected:Npe \__color_backend_separation_init:nnnnn #1#2#3#4#5
624 {
625   \bool_if:NT \g__kernel_backend_header_bool
626   {
627     \exp_not:N \exp_args:Ne \__kernel_backend_first_shipout:n
628     {
629       \exp_not:N \__color_backend_separation_init_aux:nnnnnn
630       { \exp_not:N \int_use:N \g__color_model_int }
631       {#1} {#2} {#3} {#4} {#5}
632     }
633     \prop_gput:Nee \exp_not:N \g__color_backend_colorant_prop
634     { / \exp_not:N \str_convert_pdfname:n {#1} }
635     {
636       << ~
637       /setcolorspace ~ {} ~
638       >> ~ begin ~
639       color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
640       end
641     }
642   }
643 }
644 \cs_generate_variant:Nn \__color_backend_separation_init:nnnnn { nee }
645 \cs_new_protected:Npn \__color_backend_separation_init_aux:nnnnnn #1#2#3#4#5#6
646 {

```

```

647   \__kernel_backend_literal:e
648   {
649     !
650     TeXDict ~ begin ~
651     /color #1
652     {
653       [ ~
654       /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
655       [ ~ #3 ~ ] ~
656       {
657         \cs_if_exist_use:cF { __color_backend_separation_init_ #3 :nnn }
658         { \__color_backend_separation_init:nnn }
659         {#4} {#5} {#6}
660       }
661       ] ~ setcolorspace
662     } ~ def ~
663   end
664 }
665 }
666 \cs_new:cpn { __color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
667 { \__color_backend_separation_init_Device:Nn 4 {#3} }
668 \cs_new:cpn { __color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
669 { \__color_backend_separation_init_Device:Nn 1 {#3} }
670 \cs_new:cpn { __color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
671 { \__color_backend_separation_init_Device:Nn 2 {#3} }
672 \cs_new:Npn \__color_backend_separation_init_Device:Nn #1#2
673 {
674   #2 ~
675   \prg_replicate:nn {#1}
676   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
677   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
678 }

```

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.

```

679 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3
680 {
681   \exp_args:Ne \__color_backend_separation_init:nnnn
682   { \__color_backend_separation_init_count:n {#2} }
683   {#1} {#2} {#3}
684 }
685 \cs_new:Npn \__color_backend_separation_init_count:n #1
686 { \int_eval:n { 0 \__color_backend_separation_init_count:w #1 ~ \s_color_stop } }
687 \cs_new:Npn \__color_backend_separation_init_count:w #1 ~ #2 \s_color_stop
688 {
689   +1
690   \tl_if_blank:nF {#2}
691   { \__color_backend_separation_init_count:w #2 \s_color_stop }
692 }

```

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 **C0** and **C1** arrays to be interleaved, and add a 0 to each pair: this is used to keep the stack of constant length while we are doing the first pass of mathematics. We then working through that list, calculating from the last to the first value before tidying up by removing all of the input values. We do that by first copying all of the final *y* values to the end of the stack, then rolling everything so we can pop the now-unneeded material.

```

693 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
694 {
695   \__color_backend_separation_init:w #3 ~ \s__color_stop #4 ~ \s__color_stop
696   \prg_replicate:nn {#1}
697   {
698     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
699     \int_eval:n { 3 * #1 } ~ index ~ mul ~
700     2 ~ index ~ add ~
701     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
702   }
703   \int_step_function:nnnN {#1} { -1 } { 1 }
704   \__color_backend_separation_init:n
705   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
706   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
707   \tl_if_blank:nF {#2}
708   { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
709 }
710 \cs_new:Npn \__color_backend_separation_init:w
711 #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
712 {
713   #1 ~ #3 ~ 0 ~
714   \tl_if_blank:nF {#2}
715   { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }
716 }
717 \cs_new:Npn \__color_backend_separation_init:n #1
718 { \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.

```

719 \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s__color_stop
720 {
721   #2 ~ #3 ~
722   2 ~ index ~ 2 ~ index ~ lt ~
723   { ~ pop ~ exch ~ pop ~ } ~
724   { ~
725     2 ~ index ~ 1 ~ index ~ gt ~
726     { ~ exch ~ pop ~ exch ~ pop ~ } ~
727     { ~ pop ~ pop ~ } ~
728     ifelse ~
729   }
730   ifelse ~
731   #1 ~ 1 ~ roll ~
732   \tl_if_blank:nF {#4}
733   { \__color_backend_separation_init:nw {#1} #4 \s__color_stop }

```

734 }

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.

```

735 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
736 {
737   \__color_backend_separation_init:neenn
738   {#2}
739   {
740     /CIEBasedABC ~
741     << ~
742     /RangeABC ~ [ ~ \c__color_model_range_CIELAB_tl \c_space_tl ] ~
743     /DecodeABC ~
744     [ ~
745     { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
746     { ~ 500 ~ div ~ } ~ bind ~
747     { ~ 200 ~ div ~ } ~ bind ~
748     ] ~
749     /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
750     /DecodeLMN ~
751     [ ~
752     { ~
753       dup ~ 6 ~ 29 ~ div ~ ge ~
754       { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
755       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
756       ifelse ~
757       0.9505 ~ mul ~
758     } ~ bind ~
759     { ~
760       dup ~ 6 ~ 29 ~ div ~ ge ~
761       { ~ dup ~ dup ~ mul ~ mul ~ } ~
762       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
763       ifelse ~
764     } ~ bind ~
765     { ~
766       dup ~ 6 ~ 29 ~ div ~ ge ~
767       { ~ dup ~ dup ~ mul ~ mul ~ } ~
768       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
769       ifelse ~
770       1.0890 ~ mul ~
771     } ~ bind
772     ] ~
773     /WhitePoint ~
774     [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
775     >>
776   }
777   { \c__color_model_range_CIELAB_tl }
778   { 100 ~ 0 ~ 0 }
779   {#3}
780 }

```

(End of definition for __color_backend_separation_init:nnnnn and others.)

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

```

781 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3

```

```

782 {
783   \_kernel_backend_literal:e
784   {
785     !
786     TeXDict ~ begin ~
787     /color \int_use:N \g\_color_model_int
788     {
789       [ ~
790         /DeviceN ~
791         [ ~ #1 ] ~
792         #2 ~
793         { ~ #3 ~ } ~
794         \_color_backend_devicen_colorants:n {#1}
795       ] ~ setcolorspace
796     } ~ def ~
797   end
798 }
799 }

```

(End of definition for _color_backend_devicen_init:nnn.)

_color_backend_iccbased_init:nnn No support at present.

```

800 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3 { }

```

(End of definition for _color_backend_iccbased_init:nnn.)

```

801 </dvips>

```

```

802 <*dvisvgm>

```

_color_backend_raw_separation:nn No support at present.

_color_backend_raw_devicen:nn

```

803 \cs_new:Npn \_color_backend_raw_separation:nn #1#2 { }

```

```

804 \cs_new_eq:NN \_color_backend_raw_devicen:nn \_color_backend_raw_separation:nn

```

(End of definition for _color_backend_raw_separation:nn and _color_backend_raw_devicen:nn.)

_color_backend_current_separation:nn

_color_backend_current_devicen:nn

```

805 \cs_new:Npn \_color_backend_current_separation:nn #1#2 { ~ #1 ~ #2 }

```

_color_backend_current_iccbased:nn

```

806 \cs_new_eq:NN \_color_backend_current_devicen:nn \_color_backend_current_separation:nn

```

```

807 \cs_new_eq:NN \_color_backend_current_iccbased:nn \_color_backend_current_separation:nn

```

(End of definition for _color_backend_current_separation:nn, _color_backend_current_devicen:nn, and _color_backend_current_iccbased:nn.)

_color_backend_select_separation:nn

No support at present.

_color_backend_select_devicen:nn

```

808 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2 { }

```

```

809 \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

```

810 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }

```

```

811 \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.)

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.

```

812 \cs_new:Npn \__color_backend_raw_iccbased:nn #1#2
813 {
814   <style>
815     @color-profile ~
816     \str_if_eq:nnTF {#2} { cmyk }
817     { device-cmyk }
818     { --color \int_use:N \g__color_model_int }
819     \c_space_tl
820     { src:("#1") }
821   </style>
822 }
823 \cs_new_protected:Npn \__color_backend_select_iccbased:nn #1#2
824 {
825   \__kernel_backend_literal_svg:e
826   { \__color_backend_raw_iccbased:nn {#1} {#2} }
827 }

```

(End of definition for __color_backend_raw_iccbased:nn and __color_backend_select_iccbased:nn.)

```

828 </dvisvgm>
829 <*dvipdfmx | luatex | pdftex | xetex>

```

```

\__color_backend_raw_separation:nn
\__color_backend_raw_devicen:nn
\__color_backend_raw_iccbased:nn
830 \cs_new:Npn \__color_backend_raw_separation:nn #1#2
831 { /#1 ~ cs ~ #2 ~ scn \c_space_tl /#1 ~ CS ~ #2 ~ SCN }
832 \cs_new_eq:NN \__color_backend_raw_devicen:nn \__color_backend_raw_separation:nn
833 \cs_new_eq:NN \__color_backend_raw_iccbased:nn \__color_backend_raw_separation:nn

```

(End of definition for __color_backend_raw_separation:nn, __color_backend_raw_devicen:nn, and __color_backend_raw_iccbased:nn.)

Only needed for dvipdfmx-based workflows.

```

\__color_backend_current_separation:nn
\__color_backend_current_devicen:nn
\__color_backend_current_iccbased:nn
834 <*dvipdfmx | xetex>
835 \cs_new:Npn \__color_backend_current_separation:nn #1#2 { ~ #1 ~ #2 }
836 \cs_new_eq:NN \__color_backend_current_devicen:nn \__color_backend_current_separation:nn
837 \cs_new_eq:NN \__color_backend_current_iccbased:nn \__color_backend_current_separation:nn
838 </dvipdfmx | xetex>

```

(End of definition for __color_backend_current_separation:nn, __color_backend_current_devicen:nn, and __color_backend_current_iccbased:nn.)

```

\__color_backend_select_separation:nn
\__color_backend_select_devicen:nn
\__color_backend_select_iccbased:nn
839 <*dvipdfmx | xetex>
840 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
841 { \__kernel_backend_literal:e { pdf : bc ~ \pdf_object_ref:n {#1} ~ [ #2 ] } }
842 </dvipdfmx | xetex>
843 <*luatex | pdftex>
844 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
845 {
846   \exp_after:wN \__color_backend_select:w
847   \__color_backend_raw_separation:nn {#1} {#2} \q_stop
848 }
849 </luatex | pdftex>

```



```

850 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
851 \cs_new_eq:NN \_color_backend_select_iccbased:nn \_color_backend_select_separation:nn

```

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

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

```

852 \cs_new_protected:Npn \_color_backend_init_resource:n #1
853 {
854   < *luatex | pdftex >
855   \bool_lazy_and:nnT
856     { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
857     { \pdfmanagement_if_active_p: }
858   {
859     \use:e
860     {
861       \pdfmanagement_add:nnn
862         { Page / Resources / ColorSpace }
863         { #1 }
864         { \pdf_object_ref_last: }
865     }
866   }
867   < /luatex | pdftex >
868 }

```

(End of definition for _color_backend_init_resource:n.)

_color_backend_separation_init:nnnnn
 _color_backend_separation_init:nn
 _color_backend_separation_init_CIELAB:nnn

Initializing 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/X_YTeX.

```

869 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5
870 {
871   \pdf_object_unnamed_write:ne { dict }
872   {
873     /FunctionType ~ 2
874     /Domain ~ [0 ~ 1]
875     \tl_if_blank:nF {#3} { /Range ~ [#3] }
876     /C0 ~ [#4] ~
877     /C1 ~ [#5] /N ~ 1
878   }
879   \exp_args:Ne \_color_backend_separation_init:nn
880   { \str_convert_pdfname:n {#1} } {#2}
881   \_color_backend_init_resource:n { color \int_use:N \g__color_model_int }
882 }
883 \cs_new_protected:Npn \_color_backend_separation_init:nn #1#2
884 {
885   \use:e
886   {
887     \pdf_object_new:n { color \int_use:N \g__color_model_int }
888     \pdf_object_write:nnn { color \int_use:N \g__color_model_int } { array }
889     { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
890   }

```

```

891     \prop_gput:Nne \g__color_backend_colorant_prop { /#1 }
892     { \pdf_object_ref_last: }
893 }

```

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

```

894 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
895 {
896     \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
897     {
898         \pdf_object_new:n { __color_illuminant_CIELAB_ #1 }
899         \pdf_object_write:nne { __color_illuminant_CIELAB_ #1 } { array }
900         {
901             /Lab ~
902             <<
903             /WhitePoint ~
904             [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
905             /Range ~ [ \c__color_model_range_CIELAB_tl ]
906             >>
907         }
908     }
909     \__color_backend_separation_init:nnnnn
910     {#2}
911     { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
912     { \c__color_model_range_CIELAB_tl }
913     { 100 ~ 0 ~ 0 }
914     {#3}
915 }

```

(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
\__color_backend_devicen_init:w

```

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

```

916 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
917 {
918     \pdf_object_unnamed_write:ne { stream }
919     {
920         {
921             /FunctionType ~ 4 ~
922             /Domain ~
923             [ ~
924                 \prg_replicate:nn
925                 { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
926                 { 0 ~ 1 ~ }
927             ] ~
928             /Range ~
929             [ ~
930                 \str_case:nn {#2}
931                 {
932                     { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
933                     { /DeviceGray } { 0 ~ 1 }
934                     { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
935                 } ~
936             ]

```

```

937     }
938     { {#3} }
939 }
940 \use:e
941 {
942     \pdf_object_new:n { color \int_use:N \g__color_model_int }
943     \pdf_object_write:nnn { color \int_use:N \g__color_model_int } { array }
944     {
945         /DeviceN ~
946         [ ~ #1 ] ~
947         #2 ~
948         \pdf_object_ref_last:
949         \__color_backend_devicen_colorants:n {#1}
950     }
951 }
952 \__color_backend_init_resource:n { color \int_use:N \g__color_model_int }
953 }
954 \cs_new:Npn \__color_backend_devicen_init:w #1 ~ #2 \s__color_stop
955 {
956     + 1
957     \tl_if_blank:nF {#2}
958     { \__color_backend_devicen_init:w #2 \s__color_stop }
959 }

```

(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.

```

960 \cs_new_protected:Npn \__color_backend_iccbased_init:nnn #1#2#3
961 {
962     \pdf_object_if_exist:nF { __color_icc_ #1 }
963     {
964         \pdf_object_new:n { __color_icc_ #1 }
965         \pdf_object_write:nne { __color_icc_ #1 } { fstream }
966         {
967             {
968                 /N ~ \exp_not:n { #2 } ~
969                 \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
970             }
971             {#1}
972         }
973     }
974     \pdf_object_unnamed_write:ne { array }
975     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
976     \__color_backend_init_resource:n { color \int_use:N \g__color_model_int }
977 }

```

(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.

```

978 \cs_new_protected:Npn \__color_backend_iccbased_device:nnn #1#2#3
979 {
980     \pdf_object_if_exist:nF { __color_icc_ #1 }
981     {

```

```

982     \pdf_object_new:n { __color_icc_ #1 }
983     \pdf_object_write:nnn { __color_icc_ #1 } { fstream }
984     {
985         { /N ~ #3 }
986         {#1}
987     }
988 }
989 \pdf_object_unnamed_write:ne { array }
990 { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
991 \__color_backend_init_resource:n { Default #2 }
992 }

```

(End of definition for __color_backend_iccbased_device:nnn.)

```

993 </dvipdfmx | luatex | pdftex | xetex>

```

3.4 Fill and stroke color

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

```

994 < *dvipdfmx | xetex>

```

__color_backend_fill:n Separate stroke and fill colors don't really propagate to the classical approach, so we skip \current@color here.

```

\__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
995 \cs_new_protected:Npn \__color_backend_fill:n #1
996 { \__kernel_backend_literal:n { pdf : bc ~ fill ~ [ #1 ] } }
997 \cs_new_eq:NN \__color_backend_fill_cmyk:n \__color_backend_fill:n
998 \cs_new_eq:NN \__color_backend_fill_gray:n \__color_backend_fill:n
999 \cs_new_eq:NN \__color_backend_fill_rgb:n \__color_backend_fill:n
1000 \cs_new_protected:Npn \__color_backend_stroke:n #1
1001 { \__kernel_backend_literal:n { pdf : bc ~ stroke ~ [ #1 ] } }
1002 \cs_new_eq:NN \__color_backend_stroke_cmyk:n \__color_backend_stroke:n
1003 \cs_new_eq:NN \__color_backend_stroke_gray:n \__color_backend_stroke:n
1004 \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
1005 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
1006 {
1007     \__kernel_backend_literal:e
1008     { pdf : bc ~ fill ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
1009 }
1010 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1011 {
1012     \__kernel_backend_literal:e
1013     { pdf : bc ~ stroke ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
1014 }
1015 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1016 \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: 1017 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
                                1018 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:

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

1019 </dvipdfmx | xetex>
1020 <*luatex | pdftex>
```

`\_color_backend_fill_cmyk:n` Drawing (fill/stroke) color is handled in `dvipdfmx/XqTeX` 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. Here, we can set the classical data at the same time.

```
\_color_backend_stroke_cmyk:n 1021 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
\_color_backend_stroke_gray:n 1022 { \_color_backend_fill:n { #1 ~ k } }
\_color_backend_stroke_rgb:n 1023 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
\_color_backend_stroke:n 1024 { \_color_backend_fill:n { #1 ~ g } }
1025 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1026 { \_color_backend_fill:n { #1 ~ rg } }
1027 \cs_new_protected:Npn \_color_backend_fill:n #1
1028 {
1029   \tl_set:Nn \l__color_backend_fill_tl {#1}
1030   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1031   { #1 ~ \l__color_backend_stroke_tl }
1032 }
1033 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1034 { \_color_backend_stroke:n { #1 ~ K } }
1035 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1036 { \_color_backend_stroke:n { #1 ~ G } }
1037 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1038 { \_color_backend_stroke:n { #1 ~ RG } }
1039 \cs_new_protected:Npn \_color_backend_stroke:n #1
1040 {
1041   \tl_set:Nn \l__color_backend_stroke_tl {#1}
1042   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1043   { \l__color_backend_fill_tl \c_space_tl #1 }
1044 }
```

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

```
\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn 1045 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
  \_color_backend_fill_devicen:nn 1046 { \_color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
  \_color_backend_stroke_devicen:nn 1047 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1048 { \_color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
1049 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1050 \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: 1051 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
                                1052 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:
```

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

1053 $\langle$ /luatex | pdftex $\rangle$

1054 $\langle$ *dvips $\rangle$

`\_color_backend_fill_cmyk:n` Fill color here is the same as general color *except* we skip the stroke part. Here, the fill color is more or less the same as the current color, so we just set that.

```

\__color_backend_fill_gray:n
\__color_backend_fill_rgb:n
  \__color_backend_fill:n
    \_color_backend_stroke_cmyk:n
    \_color_backend_stroke_gray:n
    \_color_backend_stroke_rgb:n.
  \__color_backend_stroke:n
1055 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1056   { \__color_backend_fill:n { cmyk ~ #1 } }
1057 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1058   { \__color_backend_fill:n { gray ~ #1 } }
1059 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1060   { \__color_backend_fill:n { rgb ~ #1 } }
1061 \cs_new_protected:Npn \__color_backend_fill:n #1
1062   { \__kernel_backend_literal:n { color~push~ #1 } }
1063 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1064   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmycolor } def } }
1065 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1066   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1067 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1068   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }
1069 \cs_new_protected:Npn \__color_backend_stroke:n #1
1070   { \__kernel_backend_postscript:n { /color.sc {#1} def } }
```

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

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
1071 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
1072   { \__color_backend_fill:n { separation ~ #1 ~ #2 } }
1073 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1074   { \__kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1075 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1076 \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:
1077 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1078 \cs_new_protected:Npn \__color_backend_stroke_reset: { }
```

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

1079 $\langle$ /dvips $\rangle$

1080 $\langle$ *dvisvgm $\rangle$

`\_color_backend_fill_cmyk:n` Fill color here is the same as general color.

```

\__color_backend_fill_gray:n
\__color_backend_fill_rgb:n
  \__color_backend_fill:n
1081 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1082   { \__color_backend_fill:n { cmyk ~ #1 } }
1083 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1084   { \__color_backend_fill:n { gray ~ #1 } }
1085 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1086   { \__color_backend_fill:n { rgb ~ #1 } }
1087 \cs_new_protected:Npn \__color_backend_fill:n #1
1088   { \__kernel_backend_literal:n { color~push~ #1 } }
```

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

`\_color_backend_stroke_cmyk:n`
`\_color_backend_stroke_gray:n`
`\_color_backend_stroke_gray_aux:n`
`\_color_backend_stroke_rgb:n`
`\_color_backend_stroke_rgb:w`
`\_color_backend:nnn`

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.

```

1089 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1090 {
1091   \_color_backend_fill_cmyk:n {#1}
1092   \_kernel_backend_scope:n { stroke = "{?color}" }
1093   \_color_backend_reset:
1094 }
1095 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1096 {
1097   \use:e
1098   {
1099     \_color_backend_stroke_gray_aux:n
1100     { \fp_eval:n { 100 * (#1) } }
1101   }
1102 }
1103 \cs_new_protected:Npn \_color_backend_stroke_gray_aux:n #1
1104 { \_color_backend:nnn {#1} {#1} {#1} }
1105 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1106 { \_color_backend_stroke_rgb:w #1 \_color_stop }
1107 \cs_new_protected:Npn \_color_backend_stroke_rgb:w
1108 #1 ~ #2 ~ #3 \_color_stop
1109 {
1110   \use:e
1111   {
1112     \_color_backend:nnn
1113     { \fp_eval:n { 100 * (#1) } }
1114     { \fp_eval:n { 100 * (#2) } }
1115     { \fp_eval:n { 100 * (#3) } }
1116   }
1117 }
1118 \cs_new_protected:Npe \_color_backend:nnn #1#2#3
1119 {
1120   \_kernel_backend_scope:n
1121   {
1122     stroke =
1123     "
1124     rgb
1125     (
1126       #1 \c_percent_str ,
1127       #2 \c_percent_str ,
1128       #3 \c_percent_str
1129     )
1130     "
1131   }
1132 }

```

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

`\_color_backend_fill_separation:nn`
`\_color_backend_stroke_separation:nn`
`\_color_backend_fill_devicen:nn`
`\_color_backend_stroke_devicen:nn`

At present, these are no-ops.

```

1133 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }

```

```

1134 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2 { }
1135 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1136 \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:
1137 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1138 \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
1139 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3 { }
1140 \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.)

```

1141 $\langle$ /dvisvgm $\rangle$

1142 $\langle$ /package $\rangle$

3.5 Font handling integration

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

```

1143  $\langle$ *lua $\rangle$ 

1144 local l = lpeg
1145 local spaces = l.P' '^0
1146 local digit16 = l.R('09', 'af', 'AF')
1147
1148 local octet = digit16 * digit16 / function(s)
1149   return string.format('%.3g ', tonumber(s, 16) / 255)
1150 end
1151
1152 if luaotfload and luaotfload.set_transparent_colorstack then
1153   local htmlcolor = l.Cs(octet * octet * octet * -1 * l.Cc'rg')
1154   local color_export = {
1155     token.create'tex_endlocalcontrol:D',
1156     token.create'tex_hpack:D',
1157     token.new(0, 1),
1158     token.create'color_export:nnN',
1159     token.new(0, 1),
1160     '',
1161     token.new(0, 2),
1162     token.new(0, 1),
1163     'backend',
1164     token.new(0, 2),
1165     token.create'l__color_tmp_tl',
1166     token.create'exp_after:wN',
1167     token.create'__color_select_auxi:nn',
1168     token.create'l__color_tmp_tl',
1169     token.new(0, 2),
1170   }
1171   local group_end = token.create'group_end:'

```



```

1172 local value = (1 - l.P')^0
1173 luatexbase.add_to_callback('luaotfload.parse_color', function (value)
1174 % Also allow HTML colors to preserve compatibility
1175     local html = htmlcolor:match(value)
1176     if html then return html end
1177
1178 % If no l3color named color with this name is known, check for defined xcolor colors
1179     local l3color_prop = token.get_macro(string.format('l_color_named_%s_prop', value))
1180     if l3color_prop == nil or l3color_prop == '' then
1181         local legacy_color_macro = token.create(string.format('\\color@%s', value))
1182         if legacy_color_macro.cmdname ~= 'undefined_cs' then
1183             token.put_next(legacy_color_macro)
1184             return token.scan_argument()
1185         end
1186     end
1187
1188     tex.runtoks(function()
1189         token.get_next()
1190         color_export[6] = value
1191         tex.sprint(-2, color_export)
1192     end)
1193     local list = token.scan_list()
1194     if not list.head or list.head.next
1195         or list.head.subtype ~= node.subtype'pdf_colorstack' then
1196         error'Unexpected backend behavior'
1197     end
1198     local cmd = list.head.data
1199     node.free(list)
1200     return cmd
1201 end, 'l3color')
1202 end
1203 </lua>
1204 <*luatex>
1205 <*package>
1206 \lua_load_module:n {l3backend-luatex}
1207 </package>
1208 </luatex>

```

4 l3backend-draw implementation

```

1209 <*package>
1210 <@@=draw>

```

4.1 dvips backend

```

1211 <*dvips>

```

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

```

\__draw_backend_literal:e 1212 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_postscript:n
1213 \cs_generate_variant:Nn \__draw_backend_literal:n { e }

```

(End of definition for `\__draw_backend_literal:n`)

`\__draw_backend_begin:` The `ps::[begin]` special here deals with positioning but allows us to continue on to a matching `ps::[end]`: contrast with `ps:`, which positions but where we can't split material between separate calls. The `@beginspecial/@endspecial` pair are from `special.pro` and correct the scale and y -axis direction. As for `pgf`, we need to save the current point as this is required for box placement. (Note that `@beginspecial/@endspecial` forms a backend scope.)

```

1214 \cs_new_protected:Npn \__draw_backend_begin:
1215 {
1216   \__draw_backend_literal:n { [begin] }
1217   \__draw_backend_literal:n { /draw.x~currentpoint~/draw.y~exch~def~def }
1218   \__draw_backend_literal:n { @beginspecial }
1219 }
1220 \cs_new_protected:Npn \__draw_backend_end:
1221 {
1222   \__draw_backend_literal:n { @endspecial }
1223   \__draw_backend_literal:n { [end] }
1224 }
```

(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:`

```

1225 \cs_new_protected:Npn \__draw_backend_scope_begin:
1226 { \__draw_backend_literal:n { save } }
1227 \cs_new_protected:Npn \__draw_backend_scope_end:
1228 { \__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 `e`-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.

```

1229 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1230 {
1231   \__draw_backend_literal:e
1232   {
1233     \dim_to_decimal_in_bp:n {#1} ~
1234     \dim_to_decimal_in_bp:n {#2} ~ moveto
1235   }
1236 }
1237 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1238 {
1239   \__draw_backend_literal:e
1240   {
1241     \dim_to_decimal_in_bp:n {#1} ~
1242     \dim_to_decimal_in_bp:n {#2} ~ lineto
1243   }
1244 }
1245 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1246 {
1247   \__draw_backend_literal:e
```

```

1248     {
1249       \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1250       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1251       moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1252     }
1253   }
1254   \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1255   {
1256     \__draw_backend_literal:e
1257     {
1258       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1259       \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1260       \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1261       curveto
1262     }
1263   }

```

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

```

\__draw_backend_evenodd_rule: The even-odd rule here can be implemented as a simply switch.
\__draw_backend_nonzero_rule:
\g__draw_draw_eor_bool
1264 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1265   { \bool_gset_true:N \g__draw_draw_eor_bool }
1266 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1267   { \bool_gset_false:N \g__draw_draw_eor_bool }
1268 \bool_new:N \g__draw_draw_eor_bool

```

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

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

```

\__draw_backend_closepath:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
1269 \cs_new_protected:Npn \__draw_backend_closepath:
1270   { \__draw_backend_literal:n { closepath } }
1271 \cs_new_protected:Npn \__draw_backend_stroke:
1272   {
1273     \__draw_backend_literal:n { gsave }
1274     \__draw_backend_literal:n { color.sc }
1275     \__draw_backend_literal:n { stroke }
1276     \__draw_backend_literal:n { grestore }
1277     \bool_if:NT \g__draw_draw_clip_bool
1278     {
1279       \__draw_backend_literal:e
1280       {
1281         \bool_if:NT \g__draw_draw_eor_bool { eo }
1282         clip
1283       }
1284     }
1285     \__draw_backend_literal:n { newpath }
1286     \bool_gset_false:N \g__draw_draw_clip_bool
1287   }

```

```

1288 \cs_new_protected:Npn \__draw_backend_closestroke:
1289 {
1290   \__draw_backend_closepath:
1291   \__draw_backend_stroke:
1292 }
1293 \cs_new_protected:Npn \__draw_backend_fill:
1294 {
1295   \__draw_backend_literal:e
1296   {
1297     \bool_if:NT \g__draw_draw_eor_bool { eo }
1298     fill
1299   }
1300   \bool_if:NT \g__draw_draw_clip_bool
1301   {
1302     \__draw_backend_literal:e
1303     {
1304       \bool_if:NT \g__draw_draw_eor_bool { eo }
1305       clip
1306     }
1307   }
1308   \__draw_backend_literal:n { newpath }
1309   \bool_gset_false:N \g__draw_draw_clip_bool
1310 }
1311 \cs_new_protected:Npn \__draw_backend_fillstroke:
1312 {
1313   \__draw_backend_literal:e
1314   {
1315     \bool_if:NT \g__draw_draw_eor_bool { eo }
1316     fill
1317   }
1318   \__draw_backend_literal:n { gsave }
1319   \__draw_backend_literal:n { color.sc }
1320   \__draw_backend_literal:n { stroke }
1321   \__draw_backend_literal:n { grestore }
1322   \bool_if:NT \g__draw_draw_clip_bool
1323   {
1324     \__draw_backend_literal:e
1325     {
1326       \bool_if:NT \g__draw_draw_eor_bool { eo }
1327       clip
1328     }
1329   }
1330   \__draw_backend_literal:n { newpath }
1331   \bool_gset_false:N \g__draw_draw_clip_bool
1332 }
1333 \cs_new_protected:Npn \__draw_backend_clip:
1334 { \bool_gset_true:N \g__draw_draw_clip_bool }
1335 \bool_new:N \g__draw_draw_clip_bool
1336 \cs_new_protected:Npn \__draw_backend_discardpath:
1337 {
1338   \bool_if:NT \g__draw_draw_clip_bool
1339   {
1340     \__draw_backend_literal:e
1341     {

```

```

1342         \bool_if:NT \g__draw_draw_eor_bool { eo }
1343         clip
1344     }
1345 }
1346 \__draw_backend_literal:n { newpath }
1347 \bool_gset_false:N \g__draw_draw_clip_bool
1348 }

```

(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:
1349 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1350 {
1351     \__draw_backend_literal:e
1352     {
1353         [
1354             \exp_args:Nf \use:n
1355             { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1356         ] ~
1357         \dim_to_decimal_in_bp:n {#2} ~ setdash
1358     }
1359 }
1360 \cs_new:Npn \__draw_backend_dash:n #1
1361 { ~ \dim_to_decimal_in_bp:n {#1} }
1362 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1363 {
1364     \__draw_backend_literal:e
1365     { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1366 }
1367 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1368 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1369 \cs_new_protected:Npn \__draw_backend_cap_but:
1370 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1371 \cs_new_protected:Npn \__draw_backend_cap_round:
1372 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1373 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1374 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1375 \cs_new_protected:Npn \__draw_backend_join_miter:
1376 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1377 \cs_new_protected:Npn \__draw_backend_join_round:
1378 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1379 \cs_new_protected:Npn \__draw_backend_join_bevel:
1380 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

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

```

\__draw_backend_transform:nnnn
\__draw_backend_shift:nn

```

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

```

1381 \cs_new_protected:Npn \__draw_backend_transform:nnnn #1#2#3#4
1382 {
1383     \__draw_backend_literal:n
1384     { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }

```

```

1385 }
1386 \cs_new_protected:Npn \__draw_backend_shift:nn #1#2
1387 {
1388   \__draw_backend_literal:n
1389   { [ 1 ~ 0 ~ 0 ~ 1 ~ #1 ~ #2 ] ~ concat }
1390 }

```

(End of definition for __draw_backend_transform:nnnn and __draw_backend_shift:nn.)

__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`.

```

1391 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1392 {
1393   \__draw_backend_literal:n { save }
1394   \__draw_backend_literal:n { 72~Resolution~div~72~VResolution~div~neg~scale }
1395   \__draw_backend_literal:n { magscale { 1~DVImag~div~dup~scale } if }
1396   \__draw_backend_literal:n { draw.x~neg~draw.y~neg~translate }
1397   \__draw_backend_literal:n { [end] }
1398   \__draw_backend_literal:n { [begin] }
1399   \__draw_backend_literal:n { save }
1400   \__draw_backend_literal:n { currentpoint }
1401   \__draw_backend_literal:n { currentpoint~translate }
1402   \__draw_backend_transform:nnnn { 1 } { 0 } { 0 } { -1 }
1403   \__draw_backend_transform:nnnn { #2 } { #3 } { #4 } { #5 }
1404   \__draw_backend_transform:nnnn { 1 } { 0 } { 0 } { -1 }
1405   \__draw_backend_literal:n { neg~exch~neg~exch~translate }
1406   \__draw_backend_literal:n { [end] }
1407   \hbox_overlap_right:n { \box_use:N #1 }
1408   \__draw_backend_literal:n { [begin] }
1409   \__draw_backend_literal:n { restore }
1410   \__draw_backend_literal:n { [end] }
1411   \__draw_backend_literal:n { [begin] }
1412   \__draw_backend_literal:n { restore }
1413 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1414 </dvips>

```

4.2 LuaTeX, pdfTeX, dvipdfmx and XeTeX

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

```

1415 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

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

`\_draw_backend_literal:e` 1416 `\cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_pdf:n`
1417 `\cs_new_eq:NN \_draw_backend_literal:e \_kernel_backend_literal_pdf: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:` 1418 `\cs_new_protected:Npn \_draw_backend_begin:`
1419 `{ \_draw_backend_scope_begin: }`
1420 `\cs_new_protected:Npn \_draw_backend_end:`
1421 `{ \_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:` 1422 `\cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:`
1423 `\cs_new_eq:NN \_draw_backend_scope_end: \_kernel_backend_scope_end:`
(End of definition for _draw_backend_scope_begin: and _draw_backend_scope_end:.)

`\_draw_backend_moveto:nn` Path creation operations all resolve directly to PDF primitive steps, with only the need
`\_draw_backend_lineto:nn` to convert to bp.
`\_draw_backend_curveto:nnnnnn` 1424 `\cs_new_protected:Npn \_draw_backend_moveto:nn #1#2`
`\_draw_backend_rectangle:nnnn` 1425 `{`
1426 `\_draw_backend_literal:e`
1427 `{ \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }`
1428 `}`
1429 `\cs_new_protected:Npn \_draw_backend_lineto:nn #1#2`
1430 `{`
1431 `\_draw_backend_literal:e`
1432 `{ \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }`
1433 `}`
1434 `\cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6`
1435 `{`
1436 `\_draw_backend_literal:e`
1437 `{`
1438 `\dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~`
1439 `\dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~`
1440 `\dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~`
1441 `c`
1442 `}`
1443 `}`
1444 `\cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4`
1445 `{`
1446 `\_draw_backend_literal:e`
1447 `{`
1448 `\dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~`
1449 `\dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~`
1450 `re`
1451 `}`
1452 `}`
(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: 1453 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
\g__draw_draw_eor_bool 1454 { \bool_gset_true:N \g__draw_draw_eor_bool }
1455 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1456 { \bool_gset_false:N \g__draw_draw_eor_bool }
1457 \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: 1458 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_closestroke: 1459 { \__draw_backend_literal:n { h } }
\__draw_backend_fill: 1460 \cs_new_protected:Npn \__draw_backend_stroke:
\__draw_backend_fillstroke: 1461 { \__draw_backend_literal:n { S } }
\__draw_backend_clip: 1462 \cs_new_protected:Npn \__draw_backend_closestroke:
\__draw_backend_discardpath: 1463 { \__draw_backend_literal:n { s } }
1464 \cs_new_protected:Npn \__draw_backend_fill:
1465 {
1466   \__draw_backend_literal:e
1467   { f \bool_if:NT \g__draw_draw_eor_bool * }
1468 }
1469 \cs_new_protected:Npn \__draw_backend_fillstroke:
1470 {
1471   \__draw_backend_literal:e
1472   { B \bool_if:NT \g__draw_draw_eor_bool * }
1473 }
1474 \cs_new_protected:Npn \__draw_backend_clip:
1475 {
1476   \__draw_backend_literal:e
1477   { W \bool_if:NT \g__draw_draw_eor_bool * }
1478 }
1479 \cs_new_protected:Npn \__draw_backend_discardpath:
1480 { \__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 1481 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
\__draw_backend_linewidth:n 1482 {
\__draw_backend_miterlimit:n 1483   \__draw_backend_literal:e
\__draw_backend_cap_butt: 1484   {
\__draw_backend_cap_round: 1485     [
\__draw_backend_cap_rectangle: 1486       \exp_args:Nf \use:n
\__draw_backend_join_miter: 1487       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
\__draw_backend_join_round: 1488     ] ~
\__draw_backend_join_bevel: 1489     \dim_to_decimal_in_bp:n {#2} ~ d
1490   }
1491 }
1492 \cs_new:Npn \__draw_backend_dash:n #1
1493 { ~ \dim_to_decimal_in_bp:n {#1} }
1494 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1495 {
1496   \__draw_backend_literal:e

```



```

1497     { \dim_to_decimal_in_bp:n {#1} ~ w }
1498   }
1499   \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1500     { \__draw_backend_literal:e { #1 ~ M } }
1501   \cs_new_protected:Npn \__draw_backend_cap_but:
1502     { \__draw_backend_literal:n { 0 ~ J } }
1503   \cs_new_protected:Npn \__draw_backend_cap_round:
1504     { \__draw_backend_literal:n { 1 ~ J } }
1505   \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1506     { \__draw_backend_literal:n { 2 ~ J } }
1507   \cs_new_protected:Npn \__draw_backend_join_miter:
1508     { \__draw_backend_literal:n { 0 ~ j } }
1509   \cs_new_protected:Npn \__draw_backend_join_round:
1510     { \__draw_backend_literal:n { 1 ~ j } }
1511   \cs_new_protected:Npn \__draw_backend_join_bevel:
1512     { \__draw_backend_literal:n { 2 ~ j } }

```

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

```

\__draw_backend_transform:nnnn
\__draw_backend_transform_aux:nnnn
\__draw_backend_shift:nn

```

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

```

1513   \cs_new_protected:Npn \__draw_backend_transform:nnnn #1#2#3#4
1514     {
1515       <*luatex | pdftex>
1516       \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1517       </luatex | pdftex>
1518       <*dvipdfmx | xetex>
1519       \str_if_eq:nnTF { #2 ~ #3 } { 0 ~ 0 }
1520       {
1521         \__kernel_backend_literal:n { x:rotate~0 }
1522         \__kernel_backend_literal:n { x:scale~#1~#4 }
1523         \__kernel_backend_literal:n { x:rotate~0 }
1524       }
1525       {
1526         \__draw_backend_transform_decompose:nnnnN {#1} {#2} {#3} {#4}
1527         \__draw_backend_transform_aux:nnnn
1528       }
1529       </dvipdfmx | xetex>
1530     }
1531   <*dvipdfmx | xetex>
1532   \cs_new_protected:Npn \__draw_backend_transform_aux:nnnn #1#2#3#4
1533     {
1534       \__kernel_backend_literal:e
1535       {
1536         x:rotate~
1537         \fp_compare:nNnTF {#1} = \c_zero_fp
1538         { 0 }

```

```

1539         { \fp_eval:n { round ( -#1 , 5 ) } }
1540     }
1541     \__kernel_backend_literal:e
1542     {
1543         x:scale~
1544         \fp_eval:n { round ( #2 , 5 ) } ~
1545         \fp_eval:n { round ( #3 , 5 ) }
1546     }
1547     \__kernel_backend_literal:e
1548     {
1549         x:rotate~
1550         \fp_compare:nNnTF {#4} = \c_zero_fp
1551         { 0 }
1552         { \fp_eval:n { round ( -#4 , 5 ) } }
1553     }
1554 }
1555 </dvipdfmx | xetex>

```

Much less complex for a shift: this is deliberately not tracked by the engine (we would otherwise do stuff in T_EX), so use the same approach for all PDF-based routes.

```

1556 \cs_new_protected:Npn \__draw_backend_shift:nn #1#2
1557 {
1558     \__draw_backend_literal:n
1559     { 1 ~ 0 ~ 0 ~ 1 ~ #1 ~ #2 ~ cm }
1560 }

```

(End of definition for `\__draw_backend_transform:nnnn`, `\__draw_backend_transform_aux:nnnn`, and `\__draw_backend_shift:nn`.)

```

\__draw_backend_transform_decompose:nnnnN
draw_backend_transform_decompose_auxi:nnnnN
draw_backend_transform_decompose_auxii:nnnnN
draw_backend_transform_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.

```

1561 <*dvipdfmx|xetex>
1562 \cs_new_protected:Npn \__draw_backend_transform_decompose:nnnnN #1#2#3#4#5
1563 {
1564   \use:e
1565   {
1566     \__draw_backend_transform_decompose_auxi:nnnnN
1567     { \fp_eval:n { (#1 + #4) / 2 } }
1568     { \fp_eval:n { (#1 - #4) / 2 } }
1569     { \fp_eval:n { (#3 + #2) / 2 } }
1570     { \fp_eval:n { (#3 - #2) / 2 } }
1571   }
1572   #5
1573 }
1574 \cs_new_protected:Npn \__draw_backend_transform_decompose_auxi:nnnnN #1#2#3#4#5
1575 {
1576   \use:e
1577   {
1578     \__draw_backend_transform_decompose_auxii:nnnnN
1579     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1580     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1581     { \fp_eval:n { atand ( #3 , #2 ) } }
1582     { \fp_eval:n { atand ( #4 , #1 ) } }
1583   }
1584   #5
1585 }
1586 \cs_new_protected:Npn \__draw_backend_transform_decompose_auxii:nnnnN #1#2#3#4#5
1587 {
1588   \use:e
1589   {
1590     \__draw_backend_transform_decompose_auxiii:nnnnN
1591     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1592     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1593     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1594     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1595   }
1596   #5
1597 }
1598 \cs_new_protected:Npn \__draw_backend_transform_decompose_auxiii:nnnnN #1#2#3#4#5
1599 {
1600   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1601   { #5 {#1} {#2} {#3} {#4} }
1602   { #5 {#1} {#3} {#2} {#4} }
1603 }
1604 </dvipdfmx|xetex>

```

(End of definition for `\__draw_backend_transform_decompose:nnnnN` and others.)

`\__draw_backend_box_use:Nnnnn` Inserting a $\text{\TeX}$ box transformed to the requested position and using the current matrix is done using a mixture of $\text{\TeX}$ and low-level manipulation. The offset can be handled by $\text{\TeX}$, 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.

```

1605 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1606 {
1607   \__kernel_backend_scope_begin:
1608   <*luatex | pdftex>
1609   \__kernel_backend_matrix:n { #2 ~ #3 ~ #4 ~ #5 }
1610   </luatex | pdftex>
1611   <*dvipdfmx | xetex>
1612   \__kernel_backend_literal:n
1613   { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1614   </dvipdfmx | xetex>
1615   \hbox_overlap_right:n { \box_use:N #1 }
1616   <*dvipdfmx | xetex>
1617   \__kernel_backend_literal:n { pdf:etrans }
1618   </dvipdfmx | xetex>
1619   \__kernel_backend_scope_end:
1620 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1621 </dvipdfmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1622 <*dvisvgm>

```

The same as the more general literal call.

```

\__draw_backend_literal:n The same as the more general literal call.
\__draw_backend_literal:e
1623 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1624 \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:
1625 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1626 \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
\__draw_backend_end: inside a scope, which as described below

```

```

1627 \cs_new_protected:Npn \__draw_backend_begin:
1628 {
1629   \__kernel_backend_scope_begin:
1630   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1631 }
1632 \cs_new_eq:NN \__draw_backend_end: \__kernel_backend_scope_end:

```

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

```

\__draw_backend_moveto:nn Once again, some work is needed to get path constructs correct. Rather than write the
\__draw_backend_lineto:nn values as they are given, the entire path needs to be collected up before being output
  \__draw_backend_rectangle:nnnn in one go. For that we use a dedicated storage routine, which adds spaces as required.
  \__draw_backend_curveto:nnnnnn Since paths should be fully expanded there is no need to worry about the internal e-type
  \__draw_backend_add_to_path:n expansion.
\g__draw_backend_path_tl

```

```

1633 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1634 {

```

```

1635 \__draw_backend_add_to_path:n
1636 { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1637 }
1638 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1639 {
1640 \__draw_backend_add_to_path:n
1641 { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1642 }
1643 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1644 {
1645 \__draw_backend_add_to_path:n
1646 {
1647 M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1648 h ~ \dim_to_decimal:n {#3} ~
1649 v ~ \dim_to_decimal:n {#4} ~
1650 h ~ \dim_to_decimal:n { -#3 } ~
1651 Z
1652 }
1653 }
1654 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1655 {
1656 \__draw_backend_add_to_path:n
1657 {
1658 C ~
1659 \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1660 \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1661 \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1662 }
1663 }
1664 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1665 {
1666 \tl_gset:Nx \g__draw_backend_path_tl
1667 {
1668 \g__draw_backend_path_tl
1669 \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1670 #1
1671 }
1672 }
1673 \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: 1674 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1675 { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1676 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1677 { \__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:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int

```

```

1678 \cs_new_protected:Npn \__draw_backend_closepath:
1679 { \__draw_backend_add_to_path:n { Z } }
1680 \cs_new_protected:Npn \__draw_backend_path:n #1
1681 {
1682   \bool_if:NTF \g__draw_draw_clip_bool
1683   {
1684     \int_gincr:N \g__kernel_clip_path_int
1685     \__draw_backend_literal:e
1686     {
1687       < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1688       { ?nl }
1689       <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1690       < /clipPath > { ? nl }
1691       <
1692       use~xlink:href =
1693       "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1694       #1
1695       />
1696     }
1697     \__kernel_backend_scope:e
1698     {
1699       clip-path =
1700       "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int )"
1701     }
1702   }
1703   {
1704     \__draw_backend_literal:e
1705     { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1706   }
1707   \tl_gclear:N \g__draw_backend_path_tl
1708   \bool_gset_false:N \g__draw_draw_clip_bool
1709 }
1710 \int_new:N \g__draw_backend_path_int
1711 \cs_new_protected:Npn \__draw_backend_stroke:
1712 { \__draw_backend_path:n { style="fill:none" } }
1713 \cs_new_protected:Npn \__draw_backend_closestroke:
1714 {
1715   \__draw_backend_closepath:
1716   \__draw_backend_stroke:
1717 }
1718 \cs_new_protected:Npn \__draw_backend_fill:
1719 { \__draw_backend_path:n { style="stroke:none" } }
1720 \cs_new_protected:Npn \__draw_backend_fillstroke:
1721 { \__draw_backend_path:n { } }
1722 \cs_new_protected:Npn \__draw_backend_clip:
1723 { \bool_gset_true:N \g__draw_draw_clip_bool }
1724 \bool_new:N \g__draw_draw_clip_bool
1725 \cs_new_protected:Npn \__draw_backend_discardpath:
1726 {
1727   \bool_if:NT \g__draw_draw_clip_bool
1728   {
1729     \int_gincr:N \g__kernel_clip_path_int
1730     \__draw_backend_literal:e
1731     {

```

```

1732         < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1733         { ?nl }
1734         <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1735         < /clipPath >
1736     }
1737     \__kernel_backend_scope:e
1738     {
1739         clip-path =
1740         "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1741     }
1742 }
1743 \tl_gclear:N \g__draw_backend_path_tl
1744 \bool_gset_false:N \g__draw_draw_clip_bool
1745 }

```

(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_but:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1746 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1747 {
1748     \use:e
1749     {
1750         \__draw_backend_dash_aux:nn
1751         { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1752         { \dim_to_decimal:n {#2} }
1753     }
1754 }
1755 \cs_new:Npn \__draw_backend_dash:n #1
1756 { , \dim_to_decimal_in_bp:n {#1} }
1757 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1758 {
1759     \__kernel_backend_scope:e
1760     {
1761         stroke-dasharray =
1762         "
1763         \tl_if_empty:nTF {#1}
1764         { none }
1765         { \use_none:n #1 }
1766         " ~
1767         stroke-offset=" #2 "
1768     }
1769 }
1770 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1771 { \__kernel_backend_scope:e { stroke-width=" \dim_to_decimal:n {#1} " } }
1772 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1773 { \__kernel_backend_scope:e { stroke-miterlimit=" #1 " } }
1774 \cs_new_protected:Npn \__draw_backend_cap_but:
1775 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1776 \cs_new_protected:Npn \__draw_backend_cap_round:
1777 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1778 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1779 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1780 \cs_new_protected:Npn \__draw_backend_join_miter:

```

```

1781 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1782 \cs_new_protected:Npn \__draw_backend_join_round:
1783 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1784 \cs_new_protected:Npn \__draw_backend_join_bevel:
1785 { \__kernel_backend_scope:n { stroke-linejoin="bevel" } }

```

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

__draw_backend_transform:nnnn
__draw_backend_shift:nn

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

```

1786 \cs_new_protected:Npn \__draw_backend_transform:nnnn #1#2#3#4
1787 {
1788   \__kernel_backend_scope:n
1789   {
1790     transform =
1791     " matrix ( #1 , #2 , #3 , #4 , 0pt , 0pt ) "
1792   }
1793 }
1794 \cs_new_protected:Npn \__draw_backend_shift:nn #1#2
1795 {
1796   \__kernel_backend_scope:n
1797   {
1798     transform =
1799     " matrix ( 1 , 0 , 0 , 1 , #1pt , #2pt ) "
1800   }
1801 }

```

(End of definition for __draw_backend_transform:nnnn and __draw_backend_shift:nn.)

__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.

```

1802 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1803 {
1804   \__kernel_backend_scope_begin:
1805   \__draw_backend_transform:nnnn {#2} {#3} {#4} {#5}
1806   \__kernel_backend_literal_svg:n
1807   {
1808     < g~
1809     stroke="none"~
1810     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1811     >
1812   }
1813   \box_set_wd:Nn #1 { 0pt }
1814   \box_set_ht:Nn #1 { 0pt }
1815   \box_set_dp:Nn #1 { 0pt }
1816   \box_use:N #1
1817   \__kernel_backend_literal_svg:n { </g> }
1818   \__kernel_backend_scope_end:
1819 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1820 </dvisvgm>
1821 </package>

```


5 l3backend-graphics implementation

```
1822 <*package>
1823 <@@=graphics>
```

5.1 dvips backend

```
1824 <*dvips>
```

```
\l_graphics_search_ext_seq
```

```
1825 \seq_set_from_clist:Nn \l_graphics_search_ext_seq { .eps , .ps }
```

(End of definition for \l_graphics_search_ext_seq.)

```
\_graphics_backend_getbb_eps:n
\_graphics_backend_getbb_ps:n
```

Simply use the generic function.

```
1826 \cs_new_eq:NN \_graphics_backend_getbb_eps:n \_graphics_read_bb:n
1827 \cs_new_eq:NN \_graphics_backend_getbb_ps:n \_graphics_read_bb:n
```

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

```
\_graphics_backend_include_eps:n
\_graphics_backend_include_ps:n
```

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

```
1828 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1829 {
1830   \__kernel_backend_literal:e
1831   {
1832     PSfile = #1 \c_space_tl
1833     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1834     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1835     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1836     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1837   }
1838 }
1839 \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
```

```
1840 \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n
```

(End of definition for _graphics_backend_get_pagecount:n.)

```
1841 </dvips>
```

5.2 LuaTeX and pdfTeX backends

```
1842 <*luatex | pdftex>
```

```
\l_graphics_search_ext_seq
```

```
1843 \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1844 { .pdf , .eps , .ps , .png , .jpg , .jpeg }
```

(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.

```
1845 \tl_new:N \l__graphics_attr_tl
```

(End of definition for `\l__graphics_attr_tl`.)

`\l__graphics_transgroup_bool` Needed to indicate that a transparency group should be applied: only currently for PDF images, but could be extended.

```
1846 \bool_new:N \l__graphics_transgroup_bool
```

(End of definition for `\l__graphics_transgroup_bool`.)

`\__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.

```
1847 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
```

```
1848 {
1849   \int_zero:N \l__graphics_page_int
1850   \tl_clear:N \l__graphics_pagebox_tl
1851   \bool_set_false:N \l__graphics_transgroup_bool
1852   \tl_set:Ne \l__graphics_attr_tl
1853   {
1854     \tl_if_empty:NF \l__graphics_decodearray_str
1855     { :D \l__graphics_decodearray_str }
1856     \bool_if:NT \l__graphics_interpolate_bool
1857     { :I }
1858     \str_if_empty:NF \l__graphics_pdf_str
1859     { :X \l__graphics_pdf_str }
1860   }
1861   \__graphics_backend_getbb_auxi:n {#1}
1862 }
```

```
1863 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
```

```
1864 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
```

```
1865 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
```

```
1866 {
1867   \tl_clear:N \l__graphics_decodearray_str
1868   \bool_set_true:N \l__graphics_transgroup_bool
1869   \bool_set_false:N \l__graphics_interpolate_bool
1870   \tl_set:Ne \l__graphics_attr_tl
1871   {
1872     : \l__graphics_pagebox_tl
1873     \int_compare:nNnT \l__graphics_page_int > 1
1874     { :P \int_use:N \l__graphics_page_int }
1875     \str_if_empty:NF \l__graphics_pdf_str
1876     { :X \l__graphics_pdf_str }
1877   }
1878   \__graphics_backend_getbb_auxi:n {#1}
1879 }
```

```

1880 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1881 {
1882   \__graphics_bb_restore:eF { #1 \l__graphics_attr_tl }
1883   { \__graphics_backend_getbb_auxii:n {#1} }
1884 }

```

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. We always apply a transparency group attribute here as included PDFs otherwise may have non-obvious behavior.

```

1885 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1886 {
1887   \exp_args:Ne \__graphics_backend_getbb_auxiii:n
1888   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1889   \int_const:cn { c__graphics_ #1 \l__graphics_attr_tl _int }
1890   { \tex_the:D \tex_pdflastximage:D }
1891   \__graphics_bb_save:e { #1 \l__graphics_attr_tl }
1892 }
1893 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1894 {
1895   \tex_immediate:D \tex_pdfximage:D
1896   \bool_lazy_any:nT
1897   {
1898     { \l__graphics_interpolate_bool }
1899     { \l__graphics_transgroup_bool }
1900     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1901     { ! \str_if_empty_p:N \l__graphics_pdf_str }
1902   }
1903   {
1904     attr
1905     {
1906       \tl_if_empty:NF \l__graphics_decodearray_str
1907       { /Decode~[ \l__graphics_decodearray_str ] }
1908       \bool_if:NT \l__graphics_transgroup_bool
1909       { /Group << /S /Transparency /K ~ false /I ~ false >> }
1910       \bool_if:NT \l__graphics_interpolate_bool
1911       { /Interpolate~true }
1912       \l__graphics_pdf_str
1913     }
1914   }
1915   \int_compare:nNnT \l__graphics_page_int > 0
1916   { page ~ \int_use:N \l__graphics_page_int }
1917   \tl_if_empty:NF \l__graphics_pagebox_tl
1918   { \l__graphics_pagebox_tl }
1919   {#1}
1920   \hbox_set:Nn \l__graphics_tmp_box
1921   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1922   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_tmp_box }
1923   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_tmp_box }
1924 }
1925 \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.

```

1926 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1927 {
1928   \tex_pdfrefximage:D
1929   \int_use:c { c__graphics_ #1 \l__graphics_attr_tl _int }
1930 }
1931 \cs_new_eq:NN \_graphics_backend_include_jpeg:n \_graphics_backend_include_jpg:n
1932 \cs_new_eq:NN \_graphics_backend_include_pdf:n \_graphics_backend_include_jpg:n
1933 \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. Modeled on the `epstopdf` L^AT_EX 2_ε package, but simplified, conversion takes place here if we have shell access.

```

1934 \sys_if_shell:T
1935 {
1936   \str_new:N \l__graphics_backend_dir_str
1937   \str_new:N \l__graphics_backend_name_str
1938   \str_new:N \l__graphics_backend_ext_str
1939   \cs_new_protected:Npn \_graphics_backend_getbb_eps:n #1
1940   {
1941     \file_parse_full_name:nNNN {#1}
1942     \l__graphics_backend_dir_str
1943     \l__graphics_backend_name_str
1944     \l__graphics_backend_ext_str
1945     \exp_args:Ne \_graphics_backend_getbb_eps:nn
1946     {
1947       \exp_args:Ne \__kernel_file_name_quote:n
1948       {
1949         \l__graphics_backend_name_str
1950         - \str_tail:N \l__graphics_backend_ext_str
1951         -converted-to.pdf
1952       }
1953     }
1954     {#1}
1955   }
1956   \cs_new_eq:NN \_graphics_backend_getbb_ps:n \_graphics_backend_getbb_eps:n
1957   \cs_new_protected:Npn \_graphics_backend_getbb_eps:nn #1#2
1958   {
1959     \file_compare_timestamp:nNnT {#2} > {#1}
1960     {
1961       \sys_shell_now:n
1962       { repstopdf ~ #2 ~ #1 }
1963     }
1964     \tl_set:Nn \l__graphics_final_name_str {#1}
1965     \_graphics_backend_getbb_pdf:n {#1}
1966   }
1967   \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1968   {
1969     \file_parse_full_name:nNNN {#1}
1970     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str

```

```

1971         \exp_args:Ne \__graphics_backend_include_pdf:n
1972         {
1973             \exp_args:Ne \__kernel_file_name_quote:n
1974             {
1975                 \l__graphics_backend_name_str
1976                 - \str_tail:N \l__graphics_backend_ext_str
1977                 -converted-to.pdf
1978             }
1979         }
1980     }
1981     \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1982 }

```

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

__graphics_backend_get_pagecount:n

Simply load and store.

```

1983 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1984 {
1985     \tex_pdfximage:D {#1}
1986     \int_const:cn { c__graphics_#1_pages_int }
1987     { \int_use:N \tex_pdflastximagepages:D }
1988 }

```

(End of definition for __graphics_backend_get_pagecount:n.)

```
1989 </luatex | pdftex>
```

5.3 dvipdfmx backend

```
1990 <dvipdfmx | xetex>
```

\l_graphics_search_ext_seq

```

1991 \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1992 { .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }

```

(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

```
1993 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
```

__graphics_backend_getbb_jpg:n

```
1994 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
```

__graphics_backend_getbb_jpeg:n

```
1995 <dvipdfmx>
```

__graphics_backend_getbb_pdf:n

```
1996 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
```

__graphics_backend_getbb_png:n

```
1997 {
1998     \int_zero:N \l__graphics_page_int
```

__graphics_backend_getbb_bmp:n

```
1999     \tl_clear:N \l__graphics_pagebox_tl
```

```
2000     \__graphics_extract_bb:n {#1}
```

```
2001 }
```

```
2002 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
```

```
2003 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
```

```
2004 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
```

```
2005 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
```

```

2006 {
2007     \tl_clear:N \l__graphics_decodearray_str
2008     \bool_set_false:N \l__graphics_interpolate_bool
2009     \__graphics_extract_bb:n {#1}
2010 }

```

```
2011 </dvipdfmx>
```

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

`\l__graphics_transgroup_bool` Needed to indicate that a transparency group should be applied: only currently for PDF images, but could be extended.

2012 `\bool_new:N \l__graphics_transgroup_bool`

(End of definition for `\l__graphics_transgroup_bool`.)

`\g__graphics_track_int` Used to track the object number associated with each graphic.

2013 `\int_new:N \g__graphics_track_int`

(End of definition for `\g__graphics_track_int`.)

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

2014 `\cs_new_protected:Npn \__graphics_backend_include_eps:n #1`

2015 `{`

2016 `\__kernel_backend_literal:e`

2017 `{`

2018 `PSfile = #1 \c_space_tl`

2019 `llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl`

2020 `lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl`

2021 `urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl`

2022 `ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim`

2023 `}`

2024 `}`

2025 `\cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n`

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.

2026 `\cs_new_protected:Npn \__graphics_backend_include_jpg:n #1`

2027 `{`

2028 `\bool_set_false:N \l__graphics_transgroup_bool`

2029 `\__graphics_backend_include_auxi:n {#1}`

2030 `}`

2031 `\cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n`

2032 `\cs_new_eq:NN \__graphics_backend_include_bmp:n \__graphics_backend_include_jpg:n`

2033 `\cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n`

2034 `\cs_new_protected:Npn \__graphics_backend_include_pdf:n #1`

2035 `{`

2036 `\bool_set_true:N \l__graphics_transgroup_bool`

2037 `\__graphics_backend_include_auxi:n {#1}`

2038 `}`

2039 `\cs_new_protected:Npn \__graphics_backend_include_auxi:n #1`

2040 `{`

2041 `\__graphics_backend_include_auxii:en`

2042 `{`

2043 `\tl_if_empty:NF \l__graphics_pagebox_tl`

2044 `{ : \l__graphics_pagebox_tl }`

2045 `\int_compare:nNnT \l__graphics_page_int > 1`

2046 `{ :P \int_use:N \l__graphics_page_int }`

```

2047     \tl_if_empty:NF \l__graphics_decodearray_str
2048     { :D \l__graphics_decodearray_str }
2049     \bool_if:NT \l__graphics_interpolate_bool
2050     { :I }
2051   }
2052   {#1}
2053 }
2054 \cs_new_protected:Npn \__graphics_backend_include_auxii:nn #1#2
2055 {
2056   \int_if_exist:cTF { c__graphics_ #2#1 _int }
2057   {
2058     \__kernel_backend_literal:e
2059     { pdf:useobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
2060   }
2061   { \__graphics_backend_include_auxiii:nn {#2} {#1} }
2062 }
2063 \cs_generate_variant:Nn \__graphics_backend_include_auxii:nn { 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! We use the `dvipdfmx` special in all cases as it allows attributes to be added to the XObject.

```

2064 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nn #1#2
2065 {
2066   \int_gincr:N \g__graphics_track_int
2067   \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
2068   \__kernel_backend_literal:e
2069   {
2070     pdf:image ~
2071     @graphic \int_use:c { c__graphics_ #1#2 _int } ~
2072     \int_compare:nNnT \l__graphics_page_int > 1
2073     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2074     \tl_if_empty:NF \l__graphics_pagebox_tl
2075     {
2076       pagebox ~ \l__graphics_pagebox_tl \c_space_tl
2077       bbox ~
2078       \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2079       \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2080       \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2081       \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
2082     }
2083     (#1)
2084     \bool_lazy_any:nT
2085     {
2086       { \l__graphics_interpolate_bool }
2087       { \l__graphics_transgroup_bool }
2088       { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
2089     }
2090     {
2091       <<
2092       \tl_if_empty:NF \l__graphics_decodearray_str
2093       { /Decode~[ \l__graphics_decodearray_str ] }
2094       \bool_if:NT \l__graphics_transgroup_bool
2095       { /Group << /S /Transparency /K ~ false /I ~ false >> }

```

```

2096             \bool_if:NT \l__graphics_interpolate_bool
2097             { /Interpolate~true }
2098         >>
2099     }
2100 }
2101 }

```

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

`\__graphics_backend_get_pagecount:n`

```

2102 <*dvipdfmx>
2103 \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n
2104 </dvipdfmx>

```

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

```

2105 </dvipdfmx | xetex>

```

5.4 X_YTeX backend

```

2106 <*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.

```

2107 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2108 {
2109     \int_zero:N \l__graphics_page_int
2110     \tl_clear:N \l__graphics_pagebox_tl
2111     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2112 }
2113 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2114 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
2115 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2116 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2117 {
2118     \tl_clear:N \l__graphics_decodearray_str
2119     \bool_set_false:N \l__graphics_interpolate_bool
2120     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2121 }
2122 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2123 {
2124     \int_compare:nNnTF \l__graphics_page_int > 1
2125     { \__graphics_backend_getbb_auxii:nN \l__graphics_page_int {#1} #2 }
2126     { \__graphics_backend_getbb_auxiii:nNn {#1} #2 { :P 1 } { page 1 } }
2127 }
2128 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nN #1#2#3
2129 { \__graphics_backend_getbb_auxiii:nNn {#2} #3 { :P #1 } { page #1 } }
2130 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nN { V }
2131 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNn #1#2#3#4
2132 {
2133     \tl_if_empty:NTF \l__graphics_pagebox_tl
2134     { \__graphics_backend_getbb_auxiv:nNnn \l__graphics_pagebox_tl }
2135     { \__graphics_backend_getbb_auxv:nNnn }

```



```

2136     {#1} #2 {#3} {#4}
2137   }
2138   \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2139   {
2140     \use:e
2141     {
2142       \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2143       {
2144         #5
2145         \tl_if_blank:nF {#1}
2146         { \c_space_tl \__graphics_backend_getbb_pagebox:w #1 }
2147       }
2148     }
2149   }
2150   \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2151   \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2152   {
2153     \__graphics_bb_restore:nF {#1#3}
2154     { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2155   }
2156   \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2157   {
2158     \hbox_set:Nn \l__graphics_tmp_box { #2 #1 ~ #4 }
2159     \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_tmp_box }
2160     \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_tmp_box }
2161     \__graphics_bb_save:n {#1#3}
2162   }
2163   \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

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

__graphics_backend_get_pagecount:n

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

```

2164   \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
2165   {
2166     \int_const:cn { c__graphics_ #1 _pages_int }
2167     {
2168       \int_max:nn
2169       { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2170       { 1 }
2171     }
2172   }

```

(End of definition for __graphics_backend_get_pagecount:n.)

2173 </xetex>

5.5 dvisvgm backend

2174 <*dvisvgm>

\l_graphics_search_ext_seq

```

2175   \seq_set_from_clist:Nn \l_graphics_search_ext_seq
2176   { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }

```

(End of definition for \l_graphics_search_ext_seq.)

```

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

```

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

```

2177 \cs_new_protected:Npn \__graphics_backend_getbb_svg:n #1
2178 {
2179   \__graphics_bb_restore:nF {#1}
2180   {
2181     \ior_open:Nn \l__graphics_tmp_ior {#1}
2182     \ior_if_eof:NTF \l__graphics_tmp_ior
2183     { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2184     {
2185       \dim_zero:N \l__graphics_llx_dim
2186       \dim_zero:N \l__graphics_lly_dim
2187       \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2188       \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2189       \ior_map_inline:Nn \l__graphics_tmp_ior
2190       {
2191         \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2192         {
2193           \__graphics_backend_getbb_svg_auxi:nNn
2194           { width } \l__graphics_urx_dim {##1}
2195         }
2196         \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2197         {
2198           \__graphics_backend_getbb_svg_auxi:nNn
2199           { height } \l__graphics_ury_dim {##1}
2200         }
2201         \bool_lazy_and:nnF
2202         { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2203         { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2204         { \ior_map_break: }
2205       }
2206       \__graphics_bb_save:n {#1}
2207     }
2208     \ior_close:N \l__graphics_tmp_ior
2209   }
2210 }
2211 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2212 {
2213   \use:e
2214   {
2215     \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxii:w
2216     ##1 \tl_to_str:n {#1} = ##2 \tl_to_str:n {#1} = ##3
2217     \s__graphics_stop
2218   }
2219   {
2220     \tl_if_blank:nF {##2}
2221     {
2222       \peek_remove_spaces:n
2223       {
2224         \peek_meaning:NTF ' % '
2225         { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2226         {

```

```

2227         \peek_meaning:NTF " % "
2228         { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2229         { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2230     }
2231 }
2232 ##2 \s__graphics_stop
2233 }
2234 }
2235 \use:e
2236 {
2237     \__graphics_backend_getbb_svg_auxii:w #3
2238     \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2239     \s__graphics_stop
2240 }
2241 }
2242 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2243 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop
2244 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2245 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2246 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2247 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2248 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2249 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2250 {
2251     \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2252     \l__graphics_tmp_dim #2 bp \scan_stop:
2253     \dim_set_eq:NN #1 \l__graphics_tmp_dim
2254 }
2255 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvii:w #1 \scan_stop: { }

```

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

__graphics_backend_getbb_eps:n
__graphics_backend_getbb_ps:n

Simply use the generic function.

```

2256 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2257 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n

```

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

__graphics_backend_getbb_png:n
__graphics_backend_getbb_jpg:n
__graphics_backend_getbb_jpeg:n

These can be included by extracting the bounding box data.

```

2258 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2259 {
2260     \int_zero:N \l__graphics_page_int
2261     \tl_clear:N \l__graphics_pagebox_tl
2262     \__graphics_extract_bb:n {#1}
2263 }
2264 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2265 \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

```

2266 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2267 {
2268     \tl_clear:N \l__graphics_decodearray_str

```

```

2269     \bool_set_false:N \l__graphics_interpolate_bool
2270     \__graphics_extract_bb:n {#1}
2271 }

```

(End of definition for __graphics_backend_getbb_pdf:n.)

```

\__graphics_backend_include_eps:n
\__graphics_backend_include_ps:n
\__graphics_backend_include_pdf:n
\__graphics_backend_include:n

```

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

```

2272 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2273 { \__graphics_backend_include:nn { PSfile } {#1} }
2274 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
2275 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2276 { \__graphics_backend_include:nn { pdffile } {#1} }
2277 \cs_new_protected:Npn \__graphics_backend_include:n #1#2
2278 {
2279     \__kernel_backend_literal:e
2280     {
2281         #1 = #2 \c_space_tl
2282         llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2283         lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2284         urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2285         ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2286     }
2287 }

```

(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.

```

2288 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2289 {
2290     \box_move_up:nn { \l__graphics_ury_dim }
2291     {
2292         \hbox:n
2293         {
2294             \__kernel_backend_literal:e
2295             {
2296                 dvisvgm:img~
2297                 \dim_to_decimal:n { \l__graphics_urx_dim } ~
2298                 \dim_to_decimal:n { \l__graphics_ury_dim } ~
2299                 \__graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2300             }
2301         }
2302     }
2303 }
2304 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_svg:n
2305 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_svg:n
2306 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_svg:n
2307 \cs_new:Npn \__graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2308 {#2}

```

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

`\_graphics_backend_get_pagecount:n`

2309 `\cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n`

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

2310 `\</dvisvgm>`

2311 `\</package>`

6 l3backend-pdf implementation

2312 `\*package>`

2313 `\@@=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 dvips backend

2314 `\*dvips>`

`\_pdf_backend_pdfmark:n`

`\_pdf_backend_pdfmark:e`

Used often enough it should be a separate function.

2315 `\cs_new_protected:Npn \_pdf_backend_pdfmark:n #1`

2316 `{ \_kernel_backend_postscript:n { mark #1 ~ pdfmark } }`

2317 `\cs_generate_variant:Nn \_pdf_backend_pdfmark:n { e }`

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

6.1.1 Catalogue entries

`\_pdf_backend_catalog_gput:nn`

`\_pdf_backend_info_gput:nn`

2318 `\cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2`

2319 `{ \_pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }`

2320 `\cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2`

2321 `{ \_pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }`

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

6.1.2 Objects

`\_pdf_backend_object_new:`

`\_pdf_backend_object_ref:n`

`\_pdf_backend_object_id:n`

2322 `\cs_new_protected:Npn \_pdf_backend_object_new:`

2323 `{ \int_gincr:N \g_pdf_backend_object_int }`

2324 `\cs_new:Npn \_pdf_backend_object_ref:n #1 { { pdf.obj #1 } }`

2325 `\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
\__pdf_backend_object_write:nne
\__pdf_backend_object_write_aux:nnn
\__pdf_backend_object_write_array:nn
\__pdf_backend_object_write_dict:nn
\__pdf_backend_object_write_fstream:nn
\__pdf_backend_object_write_stream:nn
\__pdf_backend_object_write_stream:nnn

```

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

```

2326 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2327 {
2328   \__pdf_backend_object_write_aux:nnn
2329   { \__pdf_backend_object_ref:n {#1} }
2330   {#2} {#3}
2331 }
2332 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2333 \cs_new_protected:Npn \__pdf_backend_object_write_aux:nnn #1#2#3
2334 {
2335   \__pdf_backend_pdfmark:e
2336   {
2337     /_objdef ~ #1
2338     /type
2339     \str_case:nn {#2}
2340     {
2341       { array } { /array }
2342       { dict } { /dict }
2343       { fstream } { /stream }
2344       { stream } { /stream }
2345     }
2346     /OBJ
2347   }
2348   \use:c { \__pdf_backend_object_write_ #2 :nn } {#1} {#3}
2349 }
2350 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2351 {
2352   \__pdf_backend_pdfmark:e
2353   { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2354 }
2355 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2356 {
2357   \__pdf_backend_pdfmark:e
2358   { #1 << \exp_not:n {#2} >> /PUT }
2359 }
2360 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2361 {
2362   \exp_args:Ne
2363   \__pdf_backend_object_write_fstream:nnn {#1} #2
2364 }
2365 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2366 {
2367   \__kernel_backend_postscript:n
2368   {
2369     SDict ~ begin ~
2370     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2371     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2372     end
2373   }
2374 }
2375 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2376 {
2377   \exp_args:Ne

```

```

2378     \__pdf_backend_object_write_stream:nnn {#1} #2
2379   }
2380 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2381 {
2382   \__kernel_backend_postscript:n
2383   {
2384     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2385     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2386   }
2387 }

```

(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
2388 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2389 {
2390   \int_gincr:N \g__pdf_backend_object_int
2391   \__pdf_backend_object_write_aux:nnn
2392   { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2393   {#1} {#2}
2394 }
2395 \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.

```

2396 \cs_new:Npn \__pdf_backend_object_last:
2397 { { 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.

```

2398 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2399 { { Page #1 } }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.1.3 Destinations

__pdf_backend_destination:nn Here, we need to turn the zoom into a scale. We also need to know where the current anchor point actually is: worked out in PostScript. For the rectangle version, we have a bit more PostScript: we need two points. fitr without rule spec doesn't work, so it falls back to /Fit here.

```

2400 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2401 {
2402   \__kernel_backend_postscript:n { pdf.dest.anchor }
2403   \__pdf_backend_pdfmark:e
2404   {
2405     /View
2406     [
2407       \str_case:nnF {#2}
2408       {
2409         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2410         { fit } { /Fit }

```

```

2411         { fitb } { /FitB }
2412         { fitbh } { /FitBH ~ pdf.dest.y }
2413         { fitbv } { /FitBV ~ pdf.dest.x }
2414         { fith } { /FitH ~ pdf.dest.y }
2415         { fitv } { /FitV ~ pdf.dest.x }
2416         { fitr } { /Fit }
2417     }
2418     {
2419         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2420     }
2421 ]
2422 /Dest ( \exp_not:n {#1} ) cvn
2423 /DEST
2424 }
2425 }
2426 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2427 {
2428     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2429     { \dim_eval:n {#2} } {#1} {#3} {#4}
2430 }
2431 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2432 {
2433     \vbox_to_zero:n
2434     {
2435         \dim_vertical:n {#4}
2436         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2437         \tex_vss:D
2438     }
2439     \dim_horizontal:n {#1}
2440     \vbox_to_zero:n
2441     {
2442         \dim_vertical:n { -#3 }
2443         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2444         \tex_vss:D
2445     }
2446     \dim_horizontal:n { -#1 }
2447     \__pdf_backend_pdfmark:n
2448     {
2449         /View
2450         [
2451             /FitR ~
2452             pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2453             pdf.urx ~ pdf.ury ~ pdf.dest2device
2454         ]
2455         /Dest ( #2 ) cvn
2456         /DEST
2457     }
2458 }

```

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

6.1.4 Structure

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2459 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2460 {
2461   \int_compare:nNnT {#1} = 0
2462   {
2463     \__kernel_backend_literal_postscript:n
2464     {
2465       /setdistillerparams ~ where
2466       { pop << /CompressPages ~ false >> setdistillerparams }
2467       if
2468     }
2469   }
2470 }
2471 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2472 {
2473   \bool_if:nF {#1}
2474   {
2475     \__kernel_backend_literal_postscript:n
2476     {
2477       /setdistillerparams ~ where
2478       { pop << /CompressStreams ~ false >> setdistillerparams }
2479       if
2480     }
2481   }
2482 }

```

(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
2483 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2484 {
2485   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2486 }
2487 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2488 {
2489   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2490 }

```

(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:
2491 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2492 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

```

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

6.1.5 Marked content

```

\__pdf_backend_bdc:nn
\__pdf_backend_emc:
2493 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2494 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2495 \cs_new_protected:Npn \__pdf_backend_emc:
2496 { \__pdf_backend_pdfmark:n { /EMC } }

```

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

2497 `</dvips>`

6.2 LuaTeX and pdfTeX backend

2498 `<*luatex | pdftex>`

6.2.1 Destinations

`\_pdf_backend_destination:nn`
`\_pdf_backend_destination:nnnn`

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

```

2499 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2500 {
2501   <*luatex>
2502     \tex_pdfextension:D dest ~
2503   </luatex>
2504   <*pdftex>
2505     \tex_pdfdest:D
2506   </pdftex>
2507     name {#1}
2508     \str_case:nnF {#2}
2509     {
2510       { xyz } { xyz }
2511       { fit } { fit }
2512       { fitb } { fitb }
2513       { fitbh } { fitbh }
2514       { fitbv } { fitbv }
2515       { fith } { fith }
2516       { fitv } { fitv }
2517       { fitr } { fitr }
2518     }
2519     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2520     \scan_stop:
2521   }
2522 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2523 {
2524   <*luatex>
2525     \tex_pdfextension:D dest ~
2526   </luatex>
2527   <*pdftex>
2528     \tex_pdfdest:D
2529   </pdftex>
2530     name {#1}
2531     fitr ~
2532     width \dim_eval:n {#2} ~
2533     height \dim_eval:n {#3} ~
2534     depth \dim_eval:n {#4} \scan_stop:
2535   }

```

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

6.2.2 Catalogue entries

```

\__pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
2536 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2537 {
2538   <*luatex>
2539     \tex_pdfextension:D catalog
2540   </luatex>
2541   <*pdftex>
2542     \tex_pdfcatalog:D
2543   </pdftex>
2544     { / #1 ~ #2 }
2545   }
2546 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2547 {
2548   <*luatex>
2549     \tex_pdfextension:D info
2550   </luatex>
2551   <*pdftex>
2552     \tex_pdfinfo:D
2553   </pdftex>
2554     { / #1 ~ #2 }
2555   }

```

(End of definition for __pdf_backend_catalog_gput:nn and __pdf_backend_info_gput:nn.)

6.2.3 Objects

```

\g__pdf_backend_object_prop For tracking objects to allow finalization.
2556 \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
2557 \cs_new_protected:Npn \__pdf_backend_object_new:
2558 {
2559   <*luatex>
2560     \tex_pdfextension:D obj ~
2561   </luatex>
2562   <*pdftex>
2563     \tex_pdfobj:D
2564   </pdftex>
2565     reserveobjnum ~
2566     \int_gset:Nn \g__pdf_backend_object_int
2567   <*luatex>
2568     { \tex_pdffeedback:D lastobj }
2569   </luatex>
2570   <*pdftex>
2571     { \tex_pdflastobj:D }
2572   </pdftex>
2573   }
2574 \cs_new:Npn \__pdf_backend_object_ref:n #1 { #1 ~ 0 ~ R }
2575 \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`
`\_pdf_backend_object_write:nne`
`\_pdf_backend_object_write:nn`
`\_pdf_exp_not_i:nn`
`\_pdf_exp_not_ii:nn`

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

```

2576 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
2577 {
2578   \*luatex
2579   \tex_immediate:D \tex_pdfextension:D obj ~
2580   \*pdfTeX
2581   \tex_immediate:D \tex_pdfobj:D
2582   \*pdfTeX
2583   \*pdfTeX
2584   useobjnum ~ #1
2585   \_pdf_backend_object_write:nn {#2} {#3}
2586 }
2587 \cs_new:Npn \_pdf_backend_object_write:nn #1#2
2588 {
2589   \str_case:nn {#1}
2590   {
2591     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2592     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2593     { fstream }
2594     {
2595       stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2596       file ~ { \_pdf_exp_not_ii:nn #2 }
2597     }
2598     { stream }
2599     {
2600       stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2601       { \_pdf_exp_not_ii:nn #2 }
2602     }
2603   }
2604 }
2605 \cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nne }
2606 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2607 \cs_new:Npn \_pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

```

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

`\_pdf_backend_object_now:nn`
`\_pdf_backend_object_now:ne`

Much like writing, but direct creation.

```

2608 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2609 {
2610   \*luatex
2611   \tex_immediate:D \tex_pdfextension:D obj ~
2612   \*pdfTeX
2613   \tex_immediate:D \tex_pdfobj:D
2614   \*pdfTeX
2615   \_pdf_backend_object_write:nn {#1} {#2}
2616 }
2617 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { ne }
2618

```

(End of definition for `\_pdf_backend_object_now:nn`.)

`\_pdf_backend_object_last:` Much like annotation.

```

2619 \cs_new:Npe \_pdf_backend_object_last:
2620 {
2621   \exp_not:N \int_value:w
2622   <*luatex>
2623   \exp_not:N \tex_pdffeedback:D lastobj ~
2624   </luatex>
2625   <*pdftex>
2626   \exp_not:N \tex_pdflastobj:D
2627   </pdftex>
2628   \c_space_tl 0 ~ R
2629 }

```

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

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

```

2630 \cs_new:Npe \_pdf_backend_pageobject_ref:n #1
2631 {
2632   \exp_not:N \int_value:w
2633   <*luatex>
2634   \exp_not:N \tex_pdffeedback:D pageref
2635   </luatex>
2636   <*pdftex>
2637   \exp_not:N \tex_pdfpageref:D
2638   </pdftex>
2639   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2640 }

```

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

6.2.4 Structure

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

```

\_pdf_backend_compresslevel:n
\_pdf_backend_compress_objects:n
\_pdf_backend_objcompresslevel:n
2641 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2642 {
2643   \tex_global:D
2644   <*luatex>
2645   \tex_pdfvariable:D compresslevel
2646   </luatex>
2647   <*pdftex>
2648   \tex_pdfcompresslevel:D
2649   </pdftex>
2650   \int_value:w \int_eval:n {#1} \scan_stop:
2651 }
2652 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2653 {
2654   \bool_if:nTF {#1}
2655   { \_pdf_backend_objcompresslevel:n { 2 } }
2656   { \_pdf_backend_objcompresslevel:n { 0 } }
2657 }
2658 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2659 {
2660   \tex_global:D
2661   <*luatex>

```

```

2662     \tex_pdfvariable:D objcompresslevel
2663 </luatex>
2664 <*pdftex>
2665     \tex_pdfobjcompresslevel:D
2666 </pdftex>
2667     #1 \scan_stop:
2668 }

```

(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

```

2669 \cs_new_protected:Npe \__pdf_backend_version_major_gset:n #1
2670 {
2671 <*luatex>
2672     \int_compare:nNnT \tex_luatexversion:D > { 106 }
2673     {
2674         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2675         \exp_not:N \int_eval:n {#1} \scan_stop:
2676     }
2677 </luatex>
2678 <*pdftex>
2679     \cs_if_exist:NT \tex_pdfmajorversion:D
2680     {
2681         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2682         \exp_not:N \int_eval:n {#1} \scan_stop:
2683     }
2684 </pdftex>
2685 }
2686 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2687 {
2688     \tex_global:D
2689 <*luatex>
2690     \tex_pdfvariable:D minorversion
2691 </luatex>
2692 <*pdftex>
2693     \tex_pdfminorversion:D
2694 </pdftex>
2695     \int_eval:n {#1} \scan_stop:
2696 }

```

(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:
2697 \cs_new:Npe \__pdf_backend_version_major:
2698 {
2699 <*luatex>
2700     \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2701     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2702     { 1 }
2703 </luatex>
2704 <*pdftex>
2705     \cs_if_exist:NTF \tex_pdfmajorversion:D
2706     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }

```

```

2707         { 1 }
2708     </pdftex>
2709 }
2710 \cs_new:Npn \__pdf_backend_version_minor:
2711 {
2712     \tex_the:D
2713 <*luatex>
2714     \tex_pdfvariable:D minorversion
2715 </luatex>
2716 <*pdftex>
2717     \tex_pdfminorversion:D
2718 </pdftex>
2719 }

```

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

6.2.5 Marked content

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

```

2720 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2721 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2722 \cs_new_protected:Npn \__pdf_backend_emc:
2723 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2724 </luatex | pdftex>

```

6.3 dvipdfmx backend

```

2725 <*dvipdfmx | xetex>

```

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

```

\__pdf_backend:e
2726 \cs_new_protected:Npe \__pdf_backend:n #1
2727 { \__kernel_backend_literal:n { pdf: #1 } }
2728 \cs_generate_variant:Nn \__pdf_backend:n { e }

```

(End of definition for __pdf_backend:n.)

6.3.1 Catalogue entries

```

\__pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
2729 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2730 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2731 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2732 { \__pdf_backend:n { docinfo << /#1 ~ #2 >> } }

```

(End of definition for __pdf_backend_catalog_gput:nn and __pdf_backend_info_gput:nn.)

6.3.2 Objects

`\g_pdf_backend_object_prop` For tracking objects to allow finalization.

2733 `\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` 2734 `\cs_new_protected:Npn \__pdf_backend_object_new:`
`\__pdf_backend_object_id:n` 2735 `{ \int_gincr:N \g_pdf_backend_object_int }`
2736 `\cs_new:Npn \__pdf_backend_object_ref:n #1 { @pdf.obj #1 }`
2737 `\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` 2738 `\cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3`
`\_pdf_backend_object_write_array:nn` 2739 `{`
`\_pdf_backend_object_write_dict:nn` 2740 `\use:c { \_pdf_backend_object_write_ #2 :nn }`
`\_pdf_backend_object_write_fstream:nn` 2741 `{ \_pdf_backend_object_ref:n {#1} } {#3}`
`\_pdf_backend_object_write_stream:nn` 2742 `}`
`\_pdf_backend_object_write_stream:nnnn` 2743 `\cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nne }`
2744 `\cs_new_protected:Npn \_pdf_backend_object_write_array:nn #1#2`
2745 `{`
2746 `\__pdf_backend:e`
2747 `{ obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }`
2748 `}`
2749 `\cs_new_protected:Npn \_pdf_backend_object_write_dict:nn #1#2`
2750 `{`
2751 `\__pdf_backend:e`
2752 `{ obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }`
2753 `}`
2754 `\cs_new_protected:Npn \_pdf_backend_object_write_fstream:nn #1#2`
2755 `{ \_pdf_backend_object_write_stream:nnnn { f } {#1} #2 }`
2756 `\cs_new_protected:Npn \_pdf_backend_object_write_stream:nn #1#2`
2757 `{ \_pdf_backend_object_write_stream:nnnn { } {#1} #2 }`
2758 `\cs_new_protected:Npn \_pdf_backend_object_write_stream:nnnn #1#2#3#4`
2759 `{`
2760 `\__pdf_backend:e`
2761 `{`
2762 `#1 stream ~ #2 ~`
2763 `( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>`
2764 `}`
2765 `}`

(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` 2766 `\cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2`
2767 `{`
2768 `\int_gincr:N \g_pdf_backend_object_int`
2769 `\exp_args:Nne \use:c { \_pdf_backend_object_write_ #1 :nn }`
2770 `{ @pdf.obj \int_use:N \g_pdf_backend_object_int }`
2771 `{#2}`


```

2772 }
2773 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2774 \cs_new:Npn \__pdf_backend_object_last:
2775 { @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.

```

2776 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2777 { @page #1 }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.3.3 Destinations

__pdf_backend_destination:nn
__pdf_backend_destination:nnnn
__pdf_backend_destination_aux:nnnn

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

```

2778 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2779 {
2780   \__pdf_backend:e
2781   {
2782     dest ~ ( \exp_not:n {#1} )
2783     [
2784       @thispage
2785       \str_case:nnF {#2}
2786       {
2787         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2788         { fit } { /Fit }
2789         { fitb } { /FitB }
2790         { fitbh } { /FitBH }
2791         { fitbv } { /FitBV ~ @xpos }
2792         { fith } { /FitH ~ @ypos }
2793         { fitv } { /FitV ~ @xpos }
2794         { fitr } { /Fit }
2795       }
2796       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2797     ]
2798   }
2799 }
2800 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2801 {
2802   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2803   { \dim_eval:n {#2} } {#1} {#3} {#4}
2804 }
2805 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2806 {
2807   \vbox_to_zero:n
2808   {

```

```

2809     \dim_vertical:n {#4}
2810     \hbox:n
2811     {
2812         \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2813         \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2814     }
2815     \tex_vss:D
2816 }
2817 \dim_horizontal:n {#1}
2818 \vbox_to_zero:n
2819 {
2820     \dim_vertical:n { -#3 }
2821     \hbox:n
2822     {
2823         \__pdf_backend:n
2824         {
2825             dest ~ (#2)
2826             [
2827                 @thispage
2828                 /FitR ~
2829                 @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2830                 @xpos ~ @ypos
2831             ]
2832         }
2833     }
2834     \tex_vss:D
2835 }
2836 \dim_horizontal:n { -#1 }
2837 }

```

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

6.3.4 Structure

_pdf_backend_compresslevel:n
_pdf_backend_compress_objects:n

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

```

2838 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2839 { \__kernel_backend_literal:e { dvipdfmx:config~z~ \int_eval:n {#1} } }
2840 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2841 {
2842     \bool_if:nF {#1}
2843     { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2844 }

```

(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.

```

2845 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2846 {
2847     \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2848     \__kernel_backend_literal:e { pdf:majorversion~ \__pdf_backend_version_major: }
2849 }
2850 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2851 {
2852     \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }

```

```

2853 \__kernel_backend_literal:e { pdf:minorversion~ \__pdf_backend_version_minor: }
2854 }

```

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

```

\__pdf_backend_version_major: We start with the assumption that the default is active.
\__pdf_backend_version_minor: 2855 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2856 \cs_new:Npn \__pdf_backend_version_minor: { 7 }

```

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

6.3.5 Marked content

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

```

2857 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2858 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2859 \cs_new_protected:Npn \__pdf_backend_emc:
2860 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2861 </dviPDFmx | xetex>

```

6.4 dvisvgm backend

```

2862 <*dvisvgm>

```

6.4.1 Destinations

```

\__pdf_backend_destination:nn
\__pdf_backend_destination:nnnn 2863 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2 { }
2864 \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.)

6.4.2 Catalogue entries

__pdf_backend_catalog_gput:nn No-op.
__pdf_backend_info_gput:nn 2865 \cs_new_protected:Npn __pdf_backend_catalog_gput:nn #1#2 { }
2866 \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.)

6.4.3 Objects

__pdf_backend_object_new: All no-ops here.
__pdf_backend_object_ref:n 2867 \cs_new_protected:Npn __pdf_backend_object_new: { }
__pdf_backend_object_id:n 2868 \cs_new:Npn __pdf_backend_object_ref:n #1 { }
__pdf_backend_object_write:nnn 2869 \cs_new:Npn __pdf_backend_object_id:n #1 { }
__pdf_backend_object_write:ne 2870 \cs_new_protected:Npn __pdf_backend_object_write:nnn #1#2#3 { }
__pdf_backend_object_now:nn 2871 \cs_new_protected:Npn __pdf_backend_object_write:nne #1#2#3 { }
__pdf_backend_object_now:ne 2872 \cs_new_protected:Npn __pdf_backend_object_now:nn #1#2 { }
__pdf_backend_object_last: 2873 \cs_new_protected:Npn __pdf_backend_object_now:ne #1#2 { }
__pdf_backend_pageobject_ref:n 2874 \cs_new:Npn __pdf_backend_object_last: { }
2875 \cs_new:Npn __pdf_backend_pageobject_ref:n #1 { }

(End of definition for __pdf_backend_object_new: and others.)

6.4.4 Structure

These are all no-ops.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2876 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2877 \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
\__pdf_backend_version_minor_gset:n
Data not available!
2878 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2879 \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:
\__pdf_backend_version_minor:
Data not available!
2880 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2881 \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
\__pdf_backend_emc:
More no-ops.
2882 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2883 \cs_new_protected:Npn \__pdf_backend_emc: { }

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

2884 </dvisvgm>

```

6.5 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.

```

2885 <*dvipdfmx | dvips>

\__pdf_backend_pagesize_gset:nn
This is done as a backend literal, so we deal with it using the shipout hook.
2886 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2
2887 {
2888   \__kernel_backend_first_shipout:n
2889   {
2890     \__kernel_backend_literal:e
2891     {
2892       <*dvipdfmx>
2893         pdf:pagesize ~
2894         width ~ \dim_eval:n {#1} ~
2895         height ~ \dim_eval:n {#2}
2896       </dvipdfmx>
2897       <*dvips>
2898         papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
2899       </dvips>
2900     }
2901   }
2902 }

```

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

2903 `</dvipdfmx | dvips>`

2904 `<*luatex | pdftex | xetex>`

`\_pdf_backend_pagesize_gset:nn` Pass to the primitives.

2905 `\cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2`

2906 `{`

2907 `\dim_gset:Nn \tex_pagewidth:D {#1}`

2908 `\dim_gset:Nn \tex_pageheight:D {#2}`

2909 `}`

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

2910 `</luatex | pdftex | xetex>`

2911 `<*dvisvgm>`

`\_pdf_backend_pagesize_gset:nn` A no-op.

2912 `\cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2 { }`

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

2913 `</dvisvgm>`

2914 `</package>`

7 l3backend-pdfannot implementation

2915 `<*package>`

2916 `<@@=pdfannot>`

7.1 dvips backend

2917 `<*dvips>`

In `dvips`, annotations have to be constructed manually. As such, we need the object code above for some definitions. Here, the PostScript uses the `pdf` namespace: unlike for `expl3`, we do not really control the namespacing and also have to cut across PDF-related areas.

`\l_pdfannot_backend_content_box` The content of an annotation.

2918 `\box_new:N \l__pdfannot_backend_content_box`

(End of definition for `\l__pdfannot_backend_content_box`.)

`\l_pdfannot_backend_model_box` For creating model sizing for links.

2919 `\box_new:N \l__pdfannot_backend_model_box`

(End of definition for `\l__pdfannot_backend_model_box`.)

`\g__pdfannot_backend_int` Needed to track annotations.

2920 `\int_new:N \g__pdfannot_backend_int`

(End of definition for `\g__pdfannot_backend_int`.)

_pdfannot_backend_generic:nnnn
_pdfannot_backend_generic_aux:nnnn

Annotations are objects but 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.

```

2921 \cs_new_protected:Npn \__pdfannot_backend_generic:nnnn #1#2#3#4
2922 {
2923   \exp_args:Nf \__pdfannot_backend_generic_aux:nnnn
2924   { \dim_eval:n {#1} } {#2} {#3} {#4}
2925 }
2926 \cs_new_protected:Npn \__pdfannot_backend_generic_aux:nnnn #1#2#3#4
2927 {
2928   \box_move_down:nn {#3}
2929   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2930   \box_move_up:nn {#2}
2931   {
2932     \hbox:n
2933     {
2934       \dim_horizontal:n {#1}
2935       \__kernel_backend_postscript:n { pdf.save.ur }
2936       \dim_horizontal:n { -#1 }
2937     }
2938   }
2939   \int_gincr:N \g__pdfannot_backend_int
2940   \__kernel_backend_postscript:e
2941   {
2942     mark
2943     /_objdef { pdf.annot \int_use:N \g__pdfannot_backend_int }
2944     pdf.rect
2945     #4 ~
2946     /ANN ~
2947     pdfmark
2948   }
2949 }

```

(End of definition for __pdfannot_backend_generic:nnnn and __pdfannot_backend_generic_aux:nnnn.)

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

```

2950 \cs_new:Npn \__pdfannot_backend_last:
2951 { { pdf.annot \int_use:N \g__pdfannot_backend_int } }

```

(End of definition for __pdfannot_backend_last:.)

\g__pdfannot_backend_link_int To track annotations which are links.

```

2952 \int_new:N \g__pdfannot_backend_link_int

```

(End of definition for \g__pdfannot_backend_link_int.)

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

```

2953 \tl_new:N \g__pdfannot_backend_link_dict_tl

```

(End of definition for \g__pdfannot_backend_link_dict_tl.)

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

```

2954 \int_new:N \g__pdfannot_backend_link_sf_int

```

(End of definition for `\g__pdfannot_backend_link_sf_int`.)

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

2955 `\bool_new:N \g__pdfannot_backend_link_math_bool`

(End of definition for `\g__pdfannot_backend_link_math_bool`.)

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

2956 `\bool_new:N \g__pdfannot_backend_link_bool`

(End of definition for `\g__pdfannot_backend_link_bool`.)

`\l__pdfannot_backend_breaklink_pdfmark_tl` Swappable content for link breaking.

2957 `\tl_new:N \l__pdfannot_backend_breaklink_pdfmark_tl`

2958 `\tl_set:Nn \l__pdfannot_backend_breaklink_pdfmark_tl { pdfmark }`

(End of definition for `\l__pdfannot_backend_breaklink_pdfmark_tl`.)

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

2959 `\cs_new_protected:Npn \__pdfannot_backend_breaklink_postscript:n #1 { }`

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

`\__pdfannot_backend_breaklink_usebox:N` Swappable box unpacking or use.

2960 `\cs_new_eq:NN \__pdfannot_backend_breaklink_usebox:N \box_use:N`

(End of definition for `\__pdfannot_backend_breaklink_usebox:N`.)

`\__pdfannot_backend_link_begin_goto:nnw`

`\__pdfannot_backend_link_begin_user:nnw`

`\__pdfannot_backend_link:nw`

`\__pdfannot_backend_link_aux:nw`

`\__pdfannot_backend_link_end:`

`\__pdfannot_backend_link_end_aux:`

`\__pdfannot_backend_link_minima:`

`\__pdfannot_backend_link_outerbox:n`

`\__pdfannot_backend_link_sf_save:`

`\__pdfannot_backend_link_sf_restore:`

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

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

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

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

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

2961 `\cs_new_protected:Npn \__pdfannot_backend_link_begin_goto:nnw #1#2`

2962 `{`

2963 `\__pdfannot_backend_link_begin:nw`

2964 `{ #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }`

2965 `}`

2966 `\cs_new_protected:Npn \__pdfannot_backend_link_begin_user:nnw #1#2`

2967 `{ \__pdfannot_backend_link_begin:nw {#1#2} }`

2968 `\cs_new_protected:Npn \__pdfannot_backend_link_begin:nw #1`

2969 `{`

```

2970 \bool_if:NF \g__pdfannot_backend_link_bool
2971 { \__pdfannot_backend_link_begin_aux:nw {#1} }
2972 }

```

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.

```

2973 \cs_new_protected:Npn \__pdfannot_backend_link_begin_aux:nw #1
2974 {
2975   \bool_gset_true:N \g__pdfannot_backend_link_bool
2976   \__kernel_backend_postscript:n
2977   { /pdf.link.dict ( #1 ) def }
2978   \tl_gset:Nn \g__pdfannot_backend_link_dict_tl {#1}
2979   \__pdfannot_backend_link_sf_save:
2980   \mode_if_math:TF
2981     { \bool_gset_true:N \g__pdfannot_backend_link_math_bool }
2982     { \bool_gset_false:N \g__pdfannot_backend_link_math_bool }
2983   \hbox_set:Nw \l__pdfannot_backend_content_box
2984   \__pdfannot_backend_link_sf_restore:
2985   \bool_if:NT \g__pdfannot_backend_link_math_bool
2986     { \c_math_toggle_token }
2987 }
2988 \cs_new_protected:Npn \__pdfannot_backend_link_end:
2989 {
2990   \bool_if:NT \g__pdfannot_backend_link_bool
2991     { \__pdfannot_backend_link_end_aux: }
2992 }
2993 \cs_new_protected:Npn \__pdfannot_backend_link_end_aux:
2994 {
2995   \bool_if:NT \g__pdfannot_backend_link_math_bool
2996     { \c_math_toggle_token }
2997   \__pdfannot_backend_link_sf_save:
2998   \hbox_set_end:
2999   \__pdfannot_backend_link_minima:
3000   \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
3001   \exp_args:Ne \__pdfannot_backend_link_outerbox:n
3002   {
3003     \int_if_odd:nTF { \value { page } }
3004       { \oddsidemargin }
3005       { \evensidemargin }
3006   }
3007   \box_move_down:nn { \box_dp:N \l__pdfannot_backend_content_box }
3008     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
3009   \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.begin }
3010   \__pdfannot_backend_breaklink_usebox:N \l__pdfannot_backend_content_box
3011   \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.end }
3012   \box_move_up:nn { \box_ht:N \l__pdfannot_backend_content_box }
3013     {
3014       \hbox:n
3015         { \__kernel_backend_postscript:n { pdf.save.linkur } }
3016     }
3017   \int_gincr:N \g__pdfannot_backend_int
3018   \int_gset_eq:NN \g__pdfannot_backend_link_int \g__pdfannot_backend_int
3019   \__kernel_backend_postscript:e
3020   {

```



```

3021         mark
3022         /_objdef { pdf.annot \int_use:N \g__pdfannot_backend_link_int }
3023         \g__pdfannot_backend_link_dict_tl \c_space_tl
3024         pdf.rect
3025         /ANN ~ \l__pdfannot_backend_breaklink_pdfmark_tl
3026     }
3027     \__pdfannot_backend_link_sf_restore:
3028     \bool_gset_false:N \g__pdfannot_backend_link_bool
3029 }
3030 \cs_new_protected:Npn \__pdfannot_backend_link_minima:
3031 {
3032     \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
3033     \__kernel_backend_postscript:e
3034     {
3035         /pdf.linkdp.pad ~
3036         \dim_to_decimal:n
3037         {
3038             \dim_max:nn
3039             {
3040                 \box_dp:N \l__pdfannot_backend_model_box
3041                 - \box_dp:N \l__pdfannot_backend_content_box
3042             }
3043             { Opt }
3044         } ~
3045         pdf.pt.dvi ~ def
3046         /pdf.linkht.pad ~
3047         \dim_to_decimal:n
3048         {
3049             \dim_max:nn
3050             {
3051                 \box_ht:N \l__pdfannot_backend_model_box
3052                 - \box_ht:N \l__pdfannot_backend_content_box
3053             }
3054             { Opt }
3055         } ~
3056         pdf.pt.dvi ~ def
3057     }
3058 }
3059 \cs_new_protected:Npn \__pdfannot_backend_link_outerbox:n #1
3060 {
3061     \__kernel_backend_postscript:e
3062     {
3063         /pdf.outerbox
3064         [
3065             \dim_to_decimal:n {#1} ~
3066             \dim_to_decimal:n { -\box_dp:N \l__pdfannot_backend_model_box } ~
3067             \dim_to_decimal:n { #1 + \textwidth } ~
3068             \dim_to_decimal:n { \box_ht:N \l__pdfannot_backend_model_box }
3069         ]
3070         [ exch { pdf.pt.dvi } forall ] def
3071         /pdf.baselineskip ~
3072         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
3073         { pdf.pt.dvi ~ def }
3074         { pop ~ pop }

```

```

3075         ifelse
3076     }
3077 }
3078 \cs_new_protected:Npn \__pdfannot_backend_link_sf_save:
3079 {
3080     \int_gset:Nn \g__pdfannot_backend_link_sf_int
3081     {
3082         \mode_if_horizontal:TF
3083         { \tex_spacefactor:D }
3084         { 0 }
3085     }
3086 }
3087 \cs_new_protected:Npn \__pdfannot_backend_link_sf_restore:
3088 {
3089     \mode_if_horizontal:T
3090     {
3091         \int_compare:nNnT \g__pdfannot_backend_link_sf_int > { 0 }
3092         { \int_set:Nn \tex_spacefactor:D \g__pdfannot_backend_link_sf_int }
3093     }
3094 }

```

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

Hooks to allow link breaking: something will be needed in format mode at some stage. At present this code is disabled, pending a decision to activate.

```

3095 \use_none:nnn
3096 \cs_if_exist:NT \hook_gput_code:nnn
3097 {
3098     \hook_gput_code:nnn { build/column/after } { backend }
3099     {
3100         \box_if_empty:NF \l_shipout_box
3101         {
3102             \vbox_set:Nn \l_shipout_box
3103             {
3104                 \__kernel_backend_postscript:n
3105                 {
3106                     pdf.globaldict /pdf.brokenlink.rect ~ known
3107                     { pdf.bordertracking.continue }
3108                     if
3109                 }
3110                 \vbox_unpack_drop:N \l_shipout_box
3111                 \__kernel_backend_postscript:n
3112                 { pdf.bordertracking.endpage }
3113             }
3114         }
3115     }
3116     \tl_set:Nn \l__pdfannot_backend_breaklink_pdfmark_tl { pdf.pdfmark }
3117     \cs_set_eq:NN \__pdfannot_backend_breaklink_postscript:n
3118     \__kernel_backend_postscript:n
3119     \cs_set_eq:NN \__pdfannot_backend_breaklink_usebox:N \hbox_unpack:N
3120 }

```

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

```

3121 \cs_new:Npn \__pdfannot_backend_link_last:
3122 { { pdf.annot \int_use:N \g__pdfannot_backend_link_int } }

```

(End of definition for `\_pdfannot_backend_link_last:`)

`\_pdfannot_backend_link_margin:n` Convert to big points and pass to PostScript.

```

3123 \cs_new_protected:Npn \_pdfannot_backend_link_margin:n #1
3124 {
3125   \_kernel_backend_postscript:e
3126   {
3127     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
3128   }
3129 }
```

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

`\_pdfannot_backend_link_on:`

```

\_pdfannot_backend_link_off: 3130 \cs_new_protected:Npn \_pdfannot_backend_link_on: { }
3131 \cs_new_protected:Npn \_pdfannot_backend_link_off: { }
```

(End of definition for `\_pdfannot_backend_link_on:` and `\_pdfannot_backend_link_off:`.)

```
3132 </dvips>
```

7.2 LuaTeX and pdfTeX backend

```
3133 <*luatex | pdftex>
```

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

```

3134 \cs_new_protected:Npn \_pdfannot_backend_generic:nnnn #1#2#3#4
3135 {
3136   <*luatex>
3137   \tex_pdfextension:D annot ~
3138   </luatex>
3139   <*pdftex>
3140   \tex_pdfannot:D
3141   </pdftex>
3142   width ~ \dim_eval:n {#1} ~
3143   height ~ \dim_eval:n {#2} ~
3144   depth ~ \dim_eval:n {#3} ~
3145   {#4}
3146 }
```

(End of definition for `\_pdfannot_backend_generic:nnnn`.)

`\_pdfannot_backend_last:`

A tiny amount of extra data gets added here; we use `e`-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.

```

3147 \cs_new:Npe \_pdfannot_backend_last:
3148 {
3149   \exp_not:N \int_value:w
3150   <*luatex>
3151   \exp_not:N \tex_pdffeedback:D lastannot ~
3152   </luatex>
3153   <*pdftex>
3154   \exp_not:N \tex_pdflastannot:D
3155   </pdftex>
3156   \c_space_tl 0 ~ R
3157 }
```

(End of definition for `\__pdfannot_backend_last:`.)

`\__pdfannot_backend_link_begin_goto:nnw`
`\__pdfannot_backend_link_begin_user:nnw`
`\__pdfannot_backend_link_begin:nnnw`
`\__pdfannot_backend_link_end:`

Links are all created using the same internals.

```

3158 \cs_new_protected:Npn \__pdfannot_backend_link_begin_goto:nnw #1#2
3159 { \__pdfannot_backend_link_begin:nnnw {#1} { goto~name } {#2} }
3160 \cs_new_protected:Npn \__pdfannot_backend_link_begin_user:nnw #1#2
3161 { \__pdfannot_backend_link_begin:nnnw {#1} { user } {#2} }
3162 \cs_new_protected:Npn \__pdfannot_backend_link_begin:nnnw #1#2#3
3163 {
3164   \*luatex
3165     \tex_pdfextension:D startlink ~
3166   \*pdfTeX
3167     \tex_pdfstartlink:D
3168   \*pdfTeX
3169     attr {#1}
3170     #2 {#3}
3171   }
3172 }
3173 \cs_new_protected:Npn \__pdfannot_backend_link_end:
3174 {
3175   \*luatex
3176     \tex_pdfextension:D endlink \scan_stop:
3177   \*pdfTeX
3178     \tex_pdfendlink:D
3179   \*pdfTeX
3180   }
3181 }
```

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

`\__pdfannot_backend_link_last:`

Formatted for direct use.

```

3182 \cs_new:Npe \__pdfannot_backend_link_last:
3183 {
3184   \exp_not:N \int_value:w
3185   \*luatex
3186     \exp_not:N \tex_pdffeedback:D lastlink ~
3187   \*pdfTeX
3188     \exp_not:N \tex_pdflastlink:D
3189   \*pdfTeX
3190   \c_space_tl 0 ~ R
3191 }
3192 }
```

(End of definition for `\__pdfannot_backend_link_last:`.)

`\__pdfannot_backend_link_margin:n`

A simple task: pass the data to the primitive.

```

3193 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1
3194 {
3195   \*luatex
3196     \tex_pdfvariable:D linkmargin
3197   \*pdfTeX
3198     \tex_pdflinkmargin:D
3199   \*pdfTeX
3200 }
```

```

3201     \dim_eval:n {#1} \scan_stop:
3202   }

(End of definition for \__pdfannot_backend_link_margin:n.)

```

__pdfannot_backend_link_on: Separate definitions for the two engines.

```

\__pdfannot_backend_link_off: 3203 \cs_new_protected:Npn \__pdfannot_backend_link_on:
3204   <*luatex>
3205   { \tex_pdfextension:D linkstate 0 ~ }
3206 </luatex>
3207 <*pdftex>
3208   { \tex_pdfrunninglinkon:D }
3209 </pdftex>
3210 \cs_new_protected:Npn \__pdfannot_backend_link_off:
3211   <*luatex>
3212   { \tex_pdfextension:D linkstate 1 ~ }
3213 </luatex>
3214 <*pdftex>
3215   { \tex_pdfrunninglinkoff:D }
3216 </pdftex>

(End of definition for \__pdfannot_backend_link_on: and \__pdfannot_backend_link_off:.)
3217 </luatex | pdftex>

```

7.3 dvipdfmx backend

```

3218 <*dvipdfmx | xetex>

\__pdfannot_backend:n A generic function for the backend PDF specials
\__pdfannot_backend:e 3219 \cs_new_protected:Npe \__pdfannot_backend:n #1
3220   { \__kernel_backend_literal:n { pdf: #1 } }
3221 \cs_generate_variant:Nn \__pdfannot_backend:n { e }

(End of definition for \__pdfannot_backend:n.)

\g__pdfannot_backend_int Annotations are objects: but made with a separate tracker integer.
3222 \int_new:N \g__pdfannot_backend_int

(End of definition for \g__pdfannot_backend_int.)

\__pdfannot_backend_generic:nnnn Simply pass the raw data through, just dealing with evaluation of dimensions.
3223 \cs_new_protected:Npn \__pdfannot_backend_generic:nnnn #1#2#3#4
3224   {
3225     \int_gincr:N \g__pdfannot_backend_int
3226     \__pdfannot_backend:e
3227     {
3228       ann ~ @pdfannot \int_use:N \g__pdfannot_backend_int \c_space_tl
3229       width ~ \dim_eval:n {#1} ~
3230       height ~ \dim_eval:n {#2} ~
3231       depth ~ \dim_eval:n {#3} ~
3232       << /Type /Annot #4 >>
3233     }
3234   }

(End of definition for \__pdfannot_backend_generic:nnnn.)

```

_pdfannot_backend_last:

```
3235 \cs_new:Npn \__pdfannot_backend_last:
3236 { @pdfannot \int_use:N \g__pdfannot_backend_int }
```

(End of definition for __pdfannot_backend_last:.)

\g_pdfannot_backend_link_int

To track annotations which are links.

```
3237 \int_new:N \g__pdfannot_backend_link_int
```

(End of definition for \g__pdfannot_backend_link_int.)

_pdfannot_backend_link_begin_goto:nnw

All created using the same internals.

_pdfannot_backend_link_begin_user:nnw

```
3238 \cs_new_protected:Npn \__pdfannot_backend_link_begin_goto:nnw #1#2
```

_pdfannot_backend_link_begin:n

```
3239 {
```

_pdfannot_backend_link_end:

```
3240 \__pdfannot_backend_link_begin:n
```

```
3241 { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> }
```

```
3242 }
```

```
3243 \cs_new_protected:Npn \__pdfannot_backend_link_begin_user:nnw #1#2
```

```
3244 { \__pdfannot_backend_link_begin:n {#1#2} }
```

```
3245 \cs_new_protected:Npe \__pdfannot_backend_link_begin:n #1
```

```
3246 {
```

```
3247 \int_gincr:N \exp_not:N \g__pdfannot_backend_int
```

```
3248 \int_gset_eq:NN \exp_not:N \g__pdfannot_backend_link_int
```

```
3249 \exp_not:N \g__pdfannot_backend_int
```

```
3250 \__pdfannot_backend:e
```

```
3251 {
```

```
3252 bann ~
```

```
3253 @pdfannot
```

```
3254 \exp_not:N \int_use:N \exp_not:N \g__pdfannot_backend_link_int
```

```
3255 \c_space_tl
```

```
3256 <<
```

```
3257 /Type /Annot
```

```
3258 #1
```

```
3259 >>
```

```
3260 }
```

```
3261 }
```

```
3262 \cs_new_protected:Npn \__pdfannot_backend_link_end:
```

```
3263 { \__pdfannot_backend:n { eann } }
```

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

_pdfannot_backend_link_last:

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

```
3264 \cs_new:Npn \__pdfannot_backend_link_last:
```

```
3265 { @pdfannot \int_use:N \g__pdfannot_backend_link_int }
```

(End of definition for __pdfannot_backend_link_last:.)

_pdfannot_backend_link_margin:n

Pass to dvipdfmx.

```
3266 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1
```

```
3267 { \__kernel_backend_literal:e { dvipdfmx:config~g~ \dim_eval:n {#1} } }
```

(End of definition for __pdfannot_backend_link_margin:n.)

_pdfannot_backend_link_on:

_pdfannot_backend_link_off:

```
3268 \cs_new_protected:Npn \__pdfannot_backend_link_on: { \__pdfannot_backend:n { link } }
```

```
3269 \cs_new_protected:Npn \__pdfannot_backend_link_off: { \__pdfannot_backend:n { noline } }
```

(End of definition for `\_pdfannot_backend_link_on:` and `\_pdfannot_backend_link_off:.`)

3270 `</dvipdfmx | xetex>`

7.4 dvisvgm backend

3271 `<*dvisvgm>`

`\_pdfannot_backend_generic:nnnn`

3272 `\cs_new_protected:Npn \_pdfannot_backend_generic:nnnn #1#2#3#4 { }`

(End of definition for `\_pdfannot_backend_generic:nnnn.`)

`\_pdfannot_backend_last:`

3273 `\cs_new:Npn \_pdfannot_backend_last: { }`

(End of definition for `\_pdfannot_backend_last:.`)

`\_pdfannot_backend_link_begin_goto:nnw`

`\_pdfannot_backend_link_begin_user:nnw`

`\_pdfannot_backend_link_begin:nnnw`

`\_pdfannot_backend_link_end:`

3274 `\cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nnw #1#2 { }`

3275 `\cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nnw #1#2 { }`

3276 `\cs_new_protected:Npn \_pdfannot_backend_link_begin:nnnw #1#2#3 { }`

3277 `\cs_new_protected:Npn \_pdfannot_backend_link_end: { }`

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

`\_pdfannot_backend_link_last:`

3278 `\cs_new:Npe \_pdfannot_backend_link_last: { }`

(End of definition for `\_pdfannot_backend_link_last:.`)

`\_pdfannot_backend_link_margin:n`

3279 `\cs_new_protected:Npn \_pdfannot_backend_link_margin:n #1 { }`

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

`\_pdfannot_backend_link_on:` For handling places like headers.

`\_pdfannot_backend_link_off:`

3280 `\cs_new_protected:Npn \_pdfannot_backend_link_on: { }`

3281 `\cs_new_protected:Npn \_pdfannot_backend_link_off: { }`

(End of definition for `\_pdfannot_backend_link_on:` and `\_pdfannot_backend_link_off:.`)

3282 `</dvisvgm>`

7.5 Transitional code

This block is temporary: we have moved the backend functions here to a dedicated prefix. To facilitate that, we turn off DocStrip substitution and handle things manually.

```

3283 <@@=)
3284 \cs_new_eq:NN \__pdf_backend_annotation:nnnn \__pdfannot_backend_generic:nnnn
3285 \cs_new_eq:NN \__pdf_backend_annotation_last: \__pdfannot_backend_last:
3286 \clist_map_inline:nn
3287 {
3288   begin_goto:nnw ,
3289   begin_user:nnw ,
3290   begin:nnnw ,
3291   end: ,
3292   last: ,
3293   margin:n
3294 }
3295 { \cs_new_eq:cc { __pdf_backend_link_ #1 } { __pdfannot_backend_link_ #1 } }
3296 </package>

```

8 l3backend-opacity implementation

```

3297 <*package>
3298 <@@=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.

```

3299 <*dvips>

```

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

```

3300 \cs_new_protected:Npn \__opacity_backend_select:n #1
3301 {
3302   \__opacity_backend:nnn {#1} { fill } { ca }
3303   \__opacity_backend:nnn {#1} { stroke } { CA }
3304 }
3305 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3306 {
3307   \__opacity_backend:nnn
3308     { #1 }
3309     { fill }
3310     { ca }
3311 }
3312 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3313 {
3314   \__opacity_backend:nnn
3315     { #1 }

```



```

3316     { stroke }
3317     { CA }
3318   }
3319   \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3320   {
3321     \__kernel_backend_postscript:n
3322     {
3323       product ~ (Ghostscript) ~ search
3324       {
3325         pop ~ pop ~ pop ~
3326         #1 ~ .set #2 constantalpha
3327       }
3328       {
3329         pop ~
3330         mark ~
3331         /#3 ~ #1
3332         /SetTransparency ~
3333         pdfmark
3334       }
3335     }
3336   }
3337 }
3338 \cs_new_protected:Npn \__opacity_backend_reset:
3339 {
3340   \__opacity_backend_reset_fill:
3341   \__opacity_backend_reset_stroke:
3342 }
3343 \cs_new_protected:Npn \__opacity_backend_reset_fill:
3344 {
3345   \__opacity_backend:nnn
3346   { 1 }
3347   { fill }
3348   { ca }
3349 }
3350 \cs_new_protected:Npn \__opacity_backend_reset_stroke:
3351 {
3352   \__opacity_backend:nnn
3353   { 1 }
3354   { stroke }
3355   { CA }
3356 }

```

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

```
3357 </dvips>
```

```
3358 <*dvipdfmx | luatex | pdftex | xetex>
```

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

```

3359 \bool_lazy_and:nnT
3360 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3361 { \pdfmanagement_if_active_p: }
3362 {
3363 <*luatex | pdftex>
3364   \__kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int

```

```

3365         { page ~ direct } { /opacity 1 ~ gs }
3366 </luatex|pdfTeX>
3367     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3368     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3369 }

```

(End of definition for \c_opacity_backend_stack_int.)

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

```

3370 \tl_new:N \l_opacity_backend_fill_tl
3371 \tl_new:N \l_opacity_backend_stroke_tl
3372 \tl_set:Nn \l_opacity_backend_fill_tl { 1 }
3373 \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.

```

3374 \cs_new_protected:Npn \_opacity_backend_select:n #1
3375 {
3376     \tl_set:Nn \l_opacity_backend_fill_tl {#1}
3377     \tl_set:Nn \l_opacity_backend_stroke_tl {#1}
3378     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3379     { opacity #1 }
3380     { << /ca ~ #1 /CA ~ #1 >> }
3381 <*dvipdfmx|xetex>
3382     \_kernel_backend_literal_pdf:n
3383 </dvipdfmx|xetex>
3384 <*luatex|pdfTeX>
3385     \_kernel_color_backend_stack_push:nn \c_opacity_backend_stack_int
3386 </luatex|pdfTeX>
3387     { /opacity #1 ~ gs }
3388 }
3389 \cs_new_protected:Npn \_opacity_backend_reset:
3390 {
3391 <*dvipdfmx|xetex>
3392     \_kernel_backend_literal_pdf:n
3393     { /opacity 1 ~ gs }
3394 </dvipdfmx|xetex>
3395 <*luatex|pdfTeX>
3396     \_kernel_color_backend_stack_pop:n \c_opacity_backend_stack_int
3397 </luatex|pdfTeX>
3398 }
3399 \cs_new_eq:NN \_opacity_backend_reset_fill: \_opacity_backend_reset:
3400 \cs_new_eq:NN \_opacity_backend_reset_stroke: \_opacity_backend_reset:

```

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

_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 stick to a single setting.

```

3401 \cs_new_protected:Npn \_opacity_backend_fill:n #1
3402 {
3403     \exp_args:Nno \_opacity_backend_fill_stroke:nn
3404     { #1 }
3405     { \l_opacity_backend_stroke_tl }

```

```

3406 }
3407 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3408 {
3409   \exp_args:No \__opacity_backend_fill_stroke:nn
3410   { \l__opacity_backend_fill_tl }
3411   { #1 }
3412 }
3413 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3414 {
3415   \str_if_eq:nnTF {#1} {#2}
3416   { \__opacity_backend_select:n {#1} }
3417   {
3418     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3419     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3420     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3421     { opacity.fill #1 }
3422     { << /ca ~ #1 >> }
3423     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3424     { opacity.stroke #2 }
3425     { << /CA ~ #2 >> }
3426     <*\dvipdfmx|xetex>
3427     \__kernel_backend_literal_pdf:n
3428     </dvipdfmx|xetex>
3429     <*\luatex|pdfTeX>
3430     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3431     </luatex|pdfTeX>
3432     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3433   }
3434 }

```

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

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

```

3435 \bool_lazy_and:nnF
3436 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3437 { \pdfmanagement_if_active_p: }
3438 {
3439   \cs_gset_protected:Npn \__opacity_backend_select:n #1 { }
3440   \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2 { }
3441   \cs_gset_protected:Npn \__opacity_backend_reset: { }
3442   \cs_gset_eq:NN \__opacity_backend_reset_fill: \__opacity_backend_reset:
3443   \cs_gset_eq:NN \__opacity_backend_reset_stroke: \__opacity_backend_reset:
3444 }

```

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

```

3445 </dvipdfmx|luatex|pdfTeX|xetex>
3446 <*\dvisvgm>

```

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

```

3447 \cs_new_protected:Npn \__opacity_backend_select:n #1
3448 { \__opacity_backend:nn {#1} { } }
3449 \cs_new_protected:Npn \__opacity_backend_fill:n #1

```

```

3450 { \_opacity_backend:nn {#1} { fill- } }
3451 \cs_new_protected:Npn \_opacity_backend_stroke:n #1
3452 { \_opacity_backend:nn {#1} { stroke- } }
3453 \cs_new_protected:Npn \_opacity_backend:nn #1#2
3454 { \_kernel_backend_scope:e { #2 opacity = " #1 " } }
3455 \cs_new_protected:Npn \_opacity_backend_reset: { }
3456 \cs_new_eq:NN \_opacity_backend_reset_fill: \_opacity_backend_reset:
3457 \cs_new_eq:NN \_opacity_backend_reset_stroke: \_opacity_backend_reset:

```

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

```

3458 \end{divisvgn}

```

```

3459 \end{package}

```

8.1 Font handling integration

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

```

3460 \lua

```

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

```

3461 local pdfmanagement_active do
3462   local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3463   local cmd = pdfmanagement_if_active_p.cmdname
3464   if cmd == 'undefined_cs' then
3465     pdfmanagement_active = false
3466   else
3467     token.put_next(pdfmanagement_if_active_p)
3468     pdfmanagement_active = token.scan_int() ~= 0
3469   end
3470 end
3471
3472 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3473   luaotfload.set_transparent_colorstack(function() return token.create'c_opacity_backend_st
3474
3475   local transparent_register = {
3476     token.create'pdfmanagement_add:nnn',
3477     token.new(0, 1),
3478     'Page/Resources/ExtGState',
3479     token.new(0, 2),
3480     token.new(0, 1),
3481     '',
3482     token.new(0, 2),
3483     token.new(0, 1),
3484     '<</ca ',
3485     '',
3486     '/CA ',
3487     '',
3488     '>>',
3489     token.new(0, 2),
3490   }
3491   luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)
3492     value = (octet * -1):match(value)
3493     if not value then

```

```

3494         tex.error'Invalid transparency value'
3495     return
3496 end
3497 value = value:sub(1, -2)
3498 local result = 'opacity' .. value
3499 tex.runtoks(function()
3500     transparent_register[6], transparent_register[10], transparent_register[12] = result,
3501     tex.sprint(-2, transparent_register)
3502 end)
3503 return '/' .. result .. ' gs'
3504 end, 'l3opacity')
3505 end
3506  $\langle$ /lua $\rangle$ 

```

9 l3backend-header implementation

```

3507  $\langle$ *dvips & header $\rangle$ 

```

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

```

3508 /color.sc { } def

```

(End of definition for color.sc.)

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

```

3509 TeXDict begin
3510 /TeXcolorseparation { setcolor } def
3511 end

```

(End of definition for TeXcolorseparation and separation.)

pdf.globaldict A small global dictionary for backend use.

```

3512 true setglobal
3513 /pdf.globaldict 4 dict def
3514 false setglobal

```

(End of definition for pdf.globaldict.)

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

```

3515 /pdf.cvs { 65534 string cvs } def
3516 /pdf.dvi.pt { 72.27 mul Resolution div } def
3517 /pdf.pt.dvi { 72.27 div Resolution mul } def
3518 /pdf.rect.ht { dup 1 get neg exch 3 get add } def

```

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

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

```

3519 /pdf.linkmargin { 1 pdf.pt.dvi } def
3520 /pdf.linkdp.pad { 0 } def
3521 /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	3522 /pdf.rect
pdf.save.linkur	3523 { /Rect [pdf.llx pdf.lly pdf.urx pdf.ury] } def
pdf.llx	3524 /pdf.save.ll
pdf.lly	3525 {
pdf.urx	3526 currentpoint
pdf.ury	3527 /pdf.lly exch def
	3528 /pdf.llx exch def
	3529 }
	3530 def
	3531 /pdf.save.ur
	3532 {
	3533 currentpoint
	3534 /pdf.ury exch def
	3535 /pdf.urx exch def
	3536 }
	3537 def
	3538 /pdf.save.linkll
	3539 {
	3540 currentpoint
	3541 pdf.linkmargin add
	3542 pdf.linkdp.pad add
	3543 /pdf.lly exch def
	3544 pdf.linkmargin sub
	3545 /pdf.llx exch def
	3546 }
	3547 def
	3548 /pdf.save.linkur
	3549 {
	3550 currentpoint
	3551 pdf.linkmargin sub
	3552 pdf.linkht.pad sub
	3553 /pdf.ury exch def
	3554 pdf.linkmargin add
	3555 /pdf.urx exch def
	3556 }
	3557 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	3558 /pdf.dest.anchor
pdf.dev.y	3559 {
pdf.tmpa	3560 currentpoint exch
pdf.tmpb	3561 pdf.dvi.pt 72 add
pdf.tmpc	3562 /pdf.dest.x exch def
pdf.tmpd	3563 pdf.dvi.pt
	3564 vsize 72 sub exch sub

```

3565     /pdf.dest.y exch def
3566   }
3567   def
3568 /pdf.dest.point
3569   { pdf.dest.x pdf.dest.y } def
3570 /pdf.dest2device
3571   {
3572     /pdf.dest.y exch def
3573     /pdf.dest.x exch def
3574     matrix currentmatrix
3575     matrix defaultmatrix
3576     matrix invertmatrix
3577     matrix concatmatrix
3578     cvx exec
3579     /pdf.dev.y exch def
3580     /pdf.dev.x exch def
3581     /pdf.tmpd exch def
3582     /pdf.tmpc exch def
3583     /pdf.tmpb exch def
3584     /pdf.tmpa exch def
3585     pdf.dest.x pdf.tmpa mul
3586     pdf.dest.y pdf.tmpc mul add
3587     pdf.dev.x add
3588     pdf.dest.x pdf.tmpb mul
3589     pdf.dest.y pdf.tmpd mul add
3590     pdf.dev.y add
3591   }
3592   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	3593 /pdf.bordertracking false def
pdf.brokenlink.skip	3594 /pdf.bordertracking.begin
pdf.brokenlink.dict	3595 {
pdf.bordertracking.endpage	3596 SDict /pdf.bordertracking true put
pdf.bordertracking.continue	3597 SDict /pdf.leftboundary undef
pdf.originx	3598 SDict /pdf.rightboundary undef
pdf.originy	3599 /a where
	3600 {
	3601 /a
	3602 {
	3603 currentpoint pop
	3604 SDict /pdf.rightboundary known dup
	3605 {
	3606 SDict /pdf.rightboundary get 2 index lt
	3607 { not }
	3608 if
	3609 }
	3610 if
	3611 { pop }

```

3612         { SDict exch /pdf.rightboundary exch put }
3613     ifelse
3614     moveto
3615     currentpoint pop
3616     SDict /pdf.leftboundary known dup
3617     {
3618         SDict /pdf.leftboundary get 2 index gt
3619         { not }
3620         if
3621     }
3622     if
3623     { pop }
3624     { SDict exch /pdf.leftboundary exch put }
3625     ifelse
3626 }
3627 put
3628 }
3629 if
3630 }
3631 def
3632 /pdf.bordertracking.end
3633 {
3634     /a where { /a { moveto } put } if
3635     /x where { /x { 0 exch rmoveto } put } if
3636     SDict /pdf.leftboundary known
3637     { pdf.outerbox 0 pdf.leftboundary put }
3638     if
3639     SDict /pdf.rightboundary known
3640     { pdf.outerbox 2 pdf.rightboundary put }
3641     if
3642     SDict /pdf.bordertracking false put
3643 }
3644 def
3645 /pdf.bordertracking.endpage
3646 {
3647     pdf.bordertracking
3648     {
3649         pdf.bordertracking.end
3650         true setglobal
3651         pdf.globaldict
3652         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3653         pdf.globaldict
3654         /pdf.brokenlink.skip pdf.baselineskip put
3655         pdf.globaldict
3656         /pdf.brokenlink.dict
3657         pdf.link.dict pdf.cvs put
3658         false setglobal
3659         mark pdf.link.dict cvx exec /Rect
3660         [
3661             pdf.llx
3662             pdf.lly
3663             pdf.outerbox 2 get pdf.linkmargin add
3664             currentpoint exch pop
3665             pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub

```



```

3666     ]
3667     /ANN pdf.pdfmark
3668   }
3669   if
3670 }
3671   def
3672 /pdf.bordertracking.continue
3673   {
3674     /pdf.link.dict pdf.globaldict
3675     /pdf.brokenlink.dict get def
3676     /pdf.outerbox pdf.globaldict
3677     /pdf.brokenlink.rect get def
3678     /pdf.baselineskip pdf.globaldict
3679     /pdf.brokenlink.skip get def
3680     pdf.globaldict dup dup
3681     /pdf.brokenlink.dict undef
3682     /pdf.brokenlink.skip undef
3683     /pdf.brokenlink.rect undef
3684     currentpoint
3685     /pdf.originy exch def
3686     /pdf.originx exch def
3687     /a where
3688     {
3689       /a
3690       {
3691         moveto
3692         SDict
3693         begin
3694         currentpoint pdf.originy ne exch
3695         pdf.originx ne or
3696         {
3697           pdf.save.link11
3698           /pdf.lly
3699           pdf.lly pdf.outerbox 1 get sub def
3700           pdf.bordertracking.begin
3701         }
3702         if
3703         end
3704       }
3705       put
3706     }
3707   if
3708 /x where
3709   {
3710     /x
3711     {
3712       0 exch rmoveto
3713       SDict
3714       begin
3715       currentpoint
3716       pdf.originy ne exch pdf.originx ne or
3717       {
3718         pdf.save.link11
3719         /pdf.lly

```

```

3720         pdf.lly pdf.outerbox 1 get sub def
3721         pdf.bordertracking.begin
3722     }
3723     if
3724     end
3725 }
3726 put
3727 }
3728 if
3729 }
3730 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.

```

3731 /pdf.breaklink
3732 {
3733     pop
3734     counttomark 2 mod 0 eq
3735     {
3736         counttomark /pdf.count exch def
3737         {
3738             pdf.count 0 eq { exit } if
3739             counttomark 2 roll
3740             1 index /Rect eq
3741             {
3742                 dup 4 array copy
3743                 dup dup
3744                 1 get
3745                 pdf.outerbox pdf.rect.ht
3746                 pdf.linkmargin 2 mul add sub
3747                 3 exch put
3748                 dup
3749                 pdf.outerbox 2 get
3750                 pdf.linkmargin add
3751                 2 exch put
3752                 dup dup
3753                 3 get
3754                 pdf.outerbox pdf.rect.ht
3755                 pdf.linkmargin 2 mul add add
3756                 1 exch put
3757             /pdf.currentrect exch def
3758             pdf.breaklink.write
3759             {
3760                 pdf.currentrect
3761                 dup
3762                 pdf.outerbox 0 get
3763                 pdf.linkmargin sub
3764                 0 exch put
3765                 dup

```

```

3766         pdf.outerbox 2 get
3767         pdf.linkmargin add
3768         2 exch put
3769     dup dup
3770     1 get
3771     pdf.baselineskip add
3772     1 exch put
3773     dup dup
3774     3 get
3775     pdf.baselineskip add
3776     3 exch put
3777     /pdf.currentrect exch def
3778     pdf.breaklink.write
3779 }
3780 1 index 3 get
3781 pdf.linkmargin 2 mul add
3782 pdf.outerbox pdf.rect.ht add
3783 2 index 1 get sub
3784 pdf.baselineskip div round cvi 1 sub
3785     exch
3786     repeat
3787     pdf.currentrect
3788     dup
3789     pdf.outerbox 0 get
3790     pdf.linkmargin sub
3791     0 exch put
3792     dup dup
3793     1 get
3794     pdf.baselineskip add
3795     1 exch put
3796     dup dup
3797     3 get
3798     pdf.baselineskip add
3799     3 exch put
3800     dup 2 index 2 get 2 exch put
3801     /pdf.currentrect exch def
3802     pdf.breaklink.write
3803     SDict /pdf.pdfmark.good false put
3804     exit
3805 }
3806 { pdf.count 2 sub /pdf.count exch def }
3807 ifelse
3808 }
3809 loop
3810 }
3811 if
3812 /ANN
3813 }
3814 def
3815 /pdf.breaklink.write
3816 {
3817     counttomark 1 sub
3818     index /_objdef eq
3819     {

```

```

3820         counttomark -2 roll
3821         dup wcheck
3822         {
3823             readonly
3824             counttomark 2 roll
3825         }
3826         { pop pop }
3827         ifelse
3828     }
3829     if
3830     counttomark 1 add copy
3831     pop pdf.currentrect
3832     /ANN pdfmark
3833 }
3834 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	function. Only mark types which are known are altered. At present, this is purely ANN
pdf.baselineskip	marks, which are measured relative to the size of the baseline skip. If they are more than
pdf.pdfmark.dict	one apparent line high, breaking is applied.

```

3835 /pdf.pdfmark
3836 {
3837     SDict /pdf.pdfmark.good true put
3838     dup /ANN eq
3839     {
3840         pdf.pdfmark.store
3841         pdf.pdfmark.dict
3842         begin
3843             Subtype /Link eq
3844             currentdict /Rect known and
3845             SDict /pdf.outerbox known and
3846             SDict /pdf.baselineskip known and
3847             {
3848                 Rect 3 get
3849                 pdf.linkmargin 2 mul add
3850                 pdf.outerbox pdf.rect.ht add
3851                 Rect 1 get sub
3852                 pdf.baselineskip div round cvi 0 gt
3853                 { pdf.breaklink }
3854                 if
3855             }
3856             if
3857         end
3858         SDict /pdf.outerbox undef
3859         SDict /pdf.baselineskip undef
3860         currentdict /pdf.pdfmark.dict undef
3861     }
3862     if
3863     pdf.pdfmark.good
3864     { pdfmark }
3865     { cleartomark }

```

```

3866     ifelse
3867   }
3868   def
3869 /pdf.pdfmark.store
3870 {
3871   /pdf.pdfmark.dict 65534 dict def
3872   counttomark 1 add copy
3873   pop
3874   {
3875     dup mark eq
3876     {
3877       pop
3878       exit
3879     }
3880     {
3881       pdf.pdfmark.dict
3882       begin def end
3883     }
3884     ifelse
3885   }
3886   loop
3887 }
3888 def

```

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

```

3889 </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>	1181
A	
<code>\AtBeginDvi</code>	61
B	
bool commands:	
<code>\bool_gset_false:N</code>	1267, 1286, 1309, 1331, 1347, 1456, 1708, 1744, 2982, 3028
<code>\bool_gset_true:N</code>	1265, 1334, 1454, 1723, 2975, 2981
<code>\bool_if:NTF</code>	71, 625, 1277, 1281, 1297, 1300, 1304, 1315, 1322, 1326, 1338, 1342, 1467, 1472, 1477, 1682, 1727, 1856, 1908, 1910, 2049, 2094, 2096, 2970, 2985, 2990, 2995
<code>\bool_if:nTF</code>	2473, 2654, 2842
<code>\bool_lazy_and:nnTF</code>	855, 2201, 3359, 3435
<code>\bool_lazy_any:nTF</code>	1896, 2084
<code>\bool_new:N</code>	1268, 1335, 1457, 1724, 1846, 2012, 2955, 2956
<code>\bool_set_false:N</code>	1851, 1869, 2008, 2028, 2119, 2269
<code>\bool_set_true:N</code>	1868, 2036
box commands:	
<code>\box_dp:N</code>	240, 242, 290, 292, 347, 349, 396, 398, 400, 402, 3007, 3040, 3041, 3066
<code>\box_ht:N</code>	242, 292, 349, 400, 402, 1923, 2160, 3012, 3051, 3052, 3068
<code>\box_if_empty:N</code>	3100
<code>\box_move_down:nn</code>	2928, 3007
<code>\box_move_up:nn</code>	2290, 2930, 3012
<code>\box_new:N</code>	2918, 2919
<code>\box_set_dp:Nn</code>	1815
<code>\box_set_ht:Nn</code>	1814
<code>\box_set_wd:Nn</code>	304, 1813
<code>\box_use:N</code>	247, 265, 279, 295, 322, 336, 352, 368, 380, 431, 445, 464, 1407, 1615, 1816, 2960
<code>\box_wd:N</code>	241, 249, 291, 297, 348, 354, 397, 399, 1922, 2159
box internal commands:	
<code>__box_backend_clip:N</code>	229, 229, 284, 284, 341, 341, 385, 385
<code>\l__box_backend_cos_fp</code>	299
<code>__box_backend_rotate:Nn</code>	251, 251, 299, 299, 356, 356, 435, 435
<code>__box_backend_rotate_aux:Nn</code>	251, 252, 253, 299, 300, 301, 356, 357, 358
<code>__box_backend_scale:Nnn</code>	268, 268, 327, 327, 371, 371, 448, 448
<code>\l__box_backend_sin_fp</code>	299
C	
clist commands:	
<code>\clist_map_function:nN</code>	1355, 1487, 1751
<code>\clist_map_inline:nn</code>	3286
color internal commands:	
<code>__color_backend:nnn</code>	1089, 1104, 1112, 1118
<code>\g__color_backend_colorant_prop</code>	585, 604, 607, 633, 891
<code>__color_backend_current_cmyk:n</code>	521, 521, 573, 573
<code>__color_backend_current_devicen:nn</code>	616, 617, 805, 806, 834, 836
<code>__color_backend_current_gray:n</code>	521, 522, 573, 574
<code>__color_backend_current_iccbased:nn</code>	616, 618, 805, 807, 834, 837
<code>__color_backend_current_rgb:n</code>	521, 523, 573, 575
<code>__color_backend_current_separation:nn</code>	616, 616, 617, 618, 805, 805, 806, 807, 834, 835, 836, 837
<code>__color_backend_devicen_colorants:n</code>	586, 586, 794, 949
<code>__color_backend_devicen_colorants:w</code>	586, 594, 601, 609
<code>__color_backend_devicen_init:nnn</code>	781, 781, 916, 916, 1139, 1139
<code>__color_backend_devicen_init:w</code>	916, 925, 954, 958
<code>__color_backend_fill:n</code>	995, 995, 997, 998, 999, 1021, 1022, 1024, 1026, 1027, 1046, 1055, 1056, 1058, 1060, 1061, 1072, 1081, 1082, 1084, 1086, 1087

```

\__color_backend_fill_cmyk:n ...
..... 995, 997, 1021,
1021, 1055, 1055, 1081, 1081, 1091
\__color_backend_fill_devicen:nn
..... 1005, 1015,
1045, 1049, 1071, 1075, 1133, 1135
\__color_backend_fill_gray:n 995,
998, 1021, 1023, 1055, 1057, 1081, 1083
\__color_backend_fill_reset: ...
..... 1017, 1017,
1051, 1051, 1077, 1077, 1137, 1137
\__color_backend_fill_rgb:n 995,
999, 1021, 1025, 1055, 1059, 1081, 1085
\__color_backend_fill_separation:nn
1005, 1005, 1015, 1045, 1045, 1049,
1071, 1071, 1075, 1133, 1133, 1135
\l__color_backend_fill_tl .....
..... 542, 562, 1029, 1043
\__color_backend_iccbased_-
device:nnn ..... 978, 978
\__color_backend_iccbased_-
init:nnn .....
..... 800, 800, 960, 960, 1139, 1140
\__color_backend_init_resource:n
..... 852, 852, 881, 952, 976, 991
\__color_backend_raw_cmyk:n ....
.... 511, 512, 517, 546, 546, 570, 570
\__color_backend_raw_devicen:nn .
..... 613, 614, 803, 804, 830, 832
\__color_backend_raw_gray:n ....
.... 511, 513, 518, 546, 547, 570, 571
\__color_backend_raw_iccbased:nn
.... 613, 615, 812, 812, 826, 830, 833
\__color_backend_raw_rgb:n .....
.... 511, 514, 519, 546, 548, 570, 572
\__color_backend_raw_separation:nn
..... 613, 613, 614,
803, 803, 804, 830, 830, 832, 833, 847
\__color_backend_reset: .....
524, 538, 549, 566, 576, 581, 1017,
1018, 1051, 1052, 1077, 1093, 1137
\__color_backend_select:n .....
..... 576, 576, 578, 579, 580
\__color_backend_select:nn .....
..... 524, 525, 527,
529, 530, 549, 550, 552, 554, 555, 620
\__color_backend_select:w .....
..... 549, 557, 560, 846
\__color_backend_select_cmyk:n ..
..... 524, 524, 549, 549, 576, 578
\__color_backend_select_devicen:nn
..... 619, 621, 808, 839, 850
\__color_backend_select_devicen:nnN
..... 809

\__color_backend_select_gray:n ..
..... 524, 526, 549, 551, 576, 579
\__color_backend_select_iccbased:nn
..... 619, 622, 812, 823, 839, 851
\__color_backend_select_rgb:n ...
..... 524, 528, 549, 553, 576, 580
\__color_backend_select_separation:nn
..... 619, 619, 621,
808, 808, 809, 839, 840, 844, 850, 851
\__color_backend_separation_-
init:n ..... 623, 704, 717
\__color_backend_separation_-
init:nn ..... 869, 879, 883
\__color_backend_separation_-
init:nnn ..... 623, 658, 679
\__color_backend_separation_-
init:nnnn ..... 623, 681, 693
\__color_backend_separation_-
init:nnnnn ..... 623,
623, 644, 737, 810, 810, 869, 869, 909
\__color_backend_separation_-
init:nw ..... 623, 708, 719, 733
\__color_backend_separation_-
init:w ..... 623, 695, 710, 715
\__color_backend_separation_-
init_/DeviceCMYK:nnn ..... 623
\__color_backend_separation_-
init_/DeviceGray:nnn ..... 623
\__color_backend_separation_-
init_/DeviceRGB:nnn ..... 623
\__color_backend_separation_-
init_aux:nnnnnn ..... 623, 629, 645
\__color_backend_separation_-
init_CIELAB:nnn .....
..... 623, 735, 810, 869, 894
\__color_backend_separation_-
init_CIELAB:nnnnnn ..... 811
\__color_backend_separation_-
init_count:n ..... 623, 682, 685
\__color_backend_separation_-
init_count:w ... 623, 686, 687, 691
\__color_backend_separation_-
init_Device:Nn .....
..... 623, 667, 669, 671, 672
\l__color_backend_stack_int .....
..... 472, 564, 567, 1030, 1042
\__color_backend_stroke:n .. 995,
1000, 1002, 1003, 1004, 1021, 1034,
1036, 1038, 1039, 1048, 1055, 1069
\__color_backend_stroke_cmyk:n ..
..... 995, 1002,
1021, 1033, 1055, 1063, 1089, 1089
\__color_backend_stroke_devicen:nn
..... 1005, 1016,

```

1045 , 1050 , 1071 , 1076 , 1133 , 1136	<code>\cs_new_eq:NN</code>
<code>__color_backend_stroke_gray:n</code> . .	 51 , 61 , 63 , 578 , 579 , 580 ,
. 995 , 1003 ,	614 , 617 , 618 , 621 , 804 , 806 , 807 ,
1021 , 1035 , 1055 , 1065 , 1089 , 1095	809 , 832 , 833 , 836 , 837 , 850 , 851 ,
<code>__color_backend_stroke_gray-</code>	997 , 998 , 999 , 1002 , 1003 , 1004 ,
<code>aux:n</code> 1089 , 1099 , 1103	1015 , 1016 , 1017 , 1018 , 1049 , 1050 ,
<code>__color_backend_stroke_reset:</code> . .	1051 , 1052 , 1075 , 1076 , 1077 , 1135 ,
. 1017 , 1018 ,	1136 , 1137 , 1212 , 1416 , 1417 , 1422 ,
1051 , 1052 , 1077 , 1078 , 1137 , 1138	1423 , 1623 , 1625 , 1626 , 1632 , 1826 ,
<code>__color_backend_stroke_rgb:n</code> . . .	1827 , 1839 , 1840 , 1863 , 1864 , 1931 ,
995 , 1004 , 1021 , 1037 , 1067 , 1089 , 1105	1932 , 1933 , 1956 , 1981 , 1993 , 1994 ,
<code>__color_backend_stroke_rgb:n.</code> 1055	2002 , 2003 , 2004 , 2025 , 2031 , 2032 ,
<code>__color_backend_stroke_rgb:w</code> . . .	2033 , 2103 , 2113 , 2114 , 2115 , 2256 ,
. 1089 , 1106 , 1107	2257 , 2264 , 2265 , 2274 , 2304 , 2305 ,
<code>__color_backend_stroke_separation:nn</code>	2306 , 2309 , 2325 , 2737 , 2960 , 3284 ,
1005 , 1010 , 1016 , 1045 , 1047 , 1050 ,	3285 , 3295 , 3399 , 3400 , 3456 , 3457
1071 , 1073 , 1076 , 1133 , 1134 , 1136	<code>\cs_new_protected:Npe</code>
<code>\l__color_backend_stroke_tl</code>	. . . 623 , 1118 , 2669 , 2726 , 3219 , 3245
. 542 , 563 , 1031 , 1041	<code>\cs_new_protected:Npn</code> . . . 52 , 58 ,
<code>\g__color_model_int</code> 630 , 639 , 787 ,	65 , 68 , 76 , 82 , 87 , 89 , 93 , 103 , 113 ,
818 , 881 , 887 , 888 , 942 , 943 , 952 , 976	123 , 133 , 142 , 151 , 161 , 173 , 176 ,
<code>\c__color_model_range_CIELAB_tl</code> .	178 , 180 , 184 , 189 , 198 , 208 , 218 ,
. 742 , 777 , 905 , 912	229 , 251 , 253 , 268 , 284 , 299 , 301 ,
<code>color.sc</code> 3508	327 , 341 , 356 , 358 , 371 , 385 , 435 ,
<code>cs commands:</code>	448 , 475 , 489 , 499 , 524 , 526 , 528 ,
<code>\cs:w</code> 558	530 , 538 , 549 , 551 , 553 , 555 , 560 ,
<code>\cs_end:</code> 558	566 , 576 , 581 , 619 , 622 , 645 , 735 ,
<code>\cs_generate_variant:Nn</code> . . 67 , 70 ,	781 , 800 , 808 , 810 , 811 , 823 , 840 ,
175 , 186 , 217 , 223 , 644 , 1213 , 1624 ,	844 , 852 , 869 , 883 , 894 , 916 , 960 ,
2063 , 2130 , 2150 , 2317 , 2332 , 2395 ,	978 , 995 , 1000 , 1005 , 1010 , 1021 ,
2605 , 2618 , 2728 , 2743 , 2773 , 3221	1023 , 1025 , 1027 , 1033 , 1035 , 1037 ,
<code>\cs_gset:Npe</code> . . 2485 , 2489 , 2847 , 2852	1039 , 1045 , 1047 , 1055 , 1057 , 1059 ,
<code>\cs_gset_eq:NN</code> 3442 , 3443	1061 , 1063 , 1065 , 1067 , 1069 , 1071 ,
<code>\cs_gset_protected:Npn</code>	1073 , 1078 , 1081 , 1083 , 1085 , 1087 ,
. 3439 , 3440 , 3441	1089 , 1095 , 1103 , 1105 , 1107 , 1133 ,
<code>\cs_if_exist:NTF</code>	1134 , 1138 , 1139 , 1140 , 1214 , 1220 ,
. 31 , 54 , 2679 , 2705 , 3096	1225 , 1227 , 1229 , 1237 , 1245 , 1254 ,
<code>\cs_if_exist_p:N</code> 856 , 3360 , 3436	1264 , 1266 , 1269 , 1271 , 1288 , 1293 ,
<code>\cs_if_exist_use:NTF</code> 43 , 657	1311 , 1333 , 1336 , 1349 , 1362 , 1367 ,
<code>\cs_new:Npe</code>	1369 , 1371 , 1373 , 1375 , 1377 , 1379 ,
586 , 2619 , 2630 , 2697 , 3147 , 3182 , 3278	1381 , 1386 , 1391 , 1418 , 1420 , 1424 ,
<code>\cs_new:Npn</code> 512 ,	1429 , 1434 , 1444 , 1453 , 1455 , 1458 ,
513 , 514 , 517 , 518 , 519 , 521 , 522 ,	1460 , 1462 , 1464 , 1469 , 1474 , 1479 ,
523 , 546 , 547 , 548 , 570 , 571 , 572 ,	1481 , 1494 , 1499 , 1501 , 1503 , 1505 ,
573 , 574 , 575 , 601 , 613 , 615 , 616 ,	1507 , 1509 , 1511 , 1513 , 1532 , 1556 ,
666 , 668 , 670 , 672 , 679 , 685 , 687 ,	1562 , 1574 , 1586 , 1598 , 1605 , 1627 ,
693 , 710 , 717 , 719 , 803 , 805 , 812 ,	1633 , 1638 , 1643 , 1654 , 1664 , 1674 ,
830 , 835 , 954 , 1360 , 1492 , 1755 ,	1676 , 1678 , 1680 , 1711 , 1713 , 1718 ,
1925 , 2163 , 2307 , 2324 , 2396 , 2398 ,	1720 , 1722 , 1725 , 1746 , 1757 , 1770 ,
2491 , 2492 , 2574 , 2575 , 2587 , 2606 ,	1772 , 1774 , 1776 , 1778 , 1780 , 1782 ,
2607 , 2710 , 2736 , 2774 , 2776 , 2855 ,	1784 , 1786 , 1794 , 1802 , 1828 , 1847 ,
2856 , 2868 , 2869 , 2874 , 2875 , 2880 ,	1865 , 1880 , 1885 , 1893 , 1926 , 1939 ,
2881 , 2950 , 3121 , 3235 , 3264 , 3273	1957 , 1967 , 1983 , 1996 , 2005 , 2014 ,
	2026 , 2034 , 2039 , 2054 , 2064 , 2107 ,

2116, 2122, 2128, 2131, 2138, 2151,
 2156, 2164, 2177, 2211, 2242, 2243,
 2245, 2247, 2249, 2255, 2258, 2266,
 2272, 2275, 2277, 2288, 2315, 2318,
 2320, 2322, 2326, 2333, 2350, 2355,
 2360, 2365, 2375, 2380, 2388, 2400,
 2426, 2431, 2459, 2471, 2483, 2487,
 2493, 2495, 2499, 2522, 2536, 2546,
 2557, 2576, 2608, 2641, 2652, 2658,
 2686, 2720, 2722, 2729, 2731, 2734,
 2738, 2744, 2749, 2754, 2756, 2758,
 2766, 2778, 2800, 2805, 2838, 2840,
 2845, 2850, 2857, 2859, 2863, 2864,
 2865, 2866, 2867, 2870, 2871, 2872,
 2873, 2876, 2877, 2878, 2879, 2882,
 2883, 2886, 2905, 2912, 2921, 2926,
 2959, 2961, 2966, 2968, 2973, 2988,
 2993, 3030, 3059, 3078, 3087, 3123,
 3130, 3131, 3134, 3158, 3160, 3162,
 3173, 3193, 3203, 3210, 3223, 3238,
 3243, 3262, 3266, 3268, 3269, 3272,
 3274, 3275, 3276, 3277, 3279, 3280,
 3281, 3300, 3305, 3312, 3319, 3338,
 3343, 3350, 3374, 3389, 3401, 3407,
 3413, 3447, 3449, 3451, 3453, 3455
 \dim_set_eq:NN 3117, 3119
 \cs_set_protected:Npn 2215

D

dim commands:

\dim_compare:nNnTF 2191, 2196
 \dim_compare_p:nNn 2202, 2203
 \dim_eval:n
 ... 2429, 2532, 2533, 2534, 2803,
 2894, 2895, 2898, 2924, 3142, 3143,
 3144, 3201, 3229, 3230, 3231, 3267
 \dim_gset:Nn 2907, 2908
 \dim_horizontal:n
 ... 2439, 2446, 2817, 2836, 2934, 2936
 \dim_max:nn 3038, 3049
 \dim_set:Nn
 ... 1922, 1923, 2159, 2160, 2187, 2188
 \dim_set_eq:NN 2253
 \dim_to_decimal:n .. 396, 397, 398,
 399, 400, 402, 1636, 1641, 1647,
 1648, 1649, 1650, 1659, 1660, 1661,
 1752, 1771, 2297, 2298, 3036, 3047,
 3065, 3066, 3067, 3068, 3072, 3127
 \dim_to_decimal_in_bp:n
 240, 241, 242, 290, 291, 292,
 347, 348, 349, 1233, 1234, 1241,
 1242, 1249, 1250, 1258, 1259, 1260,
 1357, 1361, 1365, 1427, 1432, 1438,
 1439, 1440, 1448, 1449, 1489, 1493,

1497, 1756, 1833, 1834, 1835, 1836,
 2019, 2020, 2021, 2022, 2078, 2079,
 2080, 2081, 2282, 2283, 2284, 2285
 \dim_vertical:n 2435, 2442, 2809, 2820
 \dim_zero:N 2185, 2186
 \c_max_dim
 .. 2187, 2188, 2191, 2196, 2202, 2203
 draw internal commands:
 __draw_backend_add_to_path:n ..
 1633,
 1635, 1640, 1645, 1656, 1664, 1679
 __draw_backend_begin:
 .. 1214, 1214, 1418, 1418, 1627, 1627
 __draw_backend_box_use:Nnnnn
 .. 1391, 1391, 1605, 1605, 1802, 1802
 __draw_backend_cap_but:
 .. 1349, 1369, 1481, 1501, 1746, 1774
 __draw_backend_cap_rectangle: ..
 .. 1349, 1373, 1481, 1505, 1746, 1778
 __draw_backend_cap_round:
 .. 1349, 1371, 1481, 1503, 1746, 1776
 __draw_backend_clip:
 .. 1269, 1333, 1458, 1474, 1678, 1722
 __draw_backend_closepath:
 1269, 1269,
 1290, 1458, 1458, 1678, 1678, 1715
 __draw_backend_closestroke: ...
 .. 1269, 1288, 1458, 1462, 1678, 1713
 __draw_backend_curveto:nnnnnn ..
 .. 1229, 1254, 1424, 1434, 1633, 1654
 __draw_backend_dash:n
 1349, 1355, 1360,
 1481, 1487, 1492, 1746, 1751, 1755
 __draw_backend_dash_aux:nn
 1746, 1750, 1757
 __draw_backend_dash_pattern:nn ..
 .. 1349, 1349, 1481, 1481, 1746, 1746
 __draw_backend_discardpath: ...
 .. 1269, 1336, 1458, 1479, 1678, 1725
 __draw_backend_end:
 .. 1214, 1220, 1418, 1420, 1627, 1632
 __draw_backend_evenodd_rule: ...
 .. 1264, 1264, 1453, 1453, 1674, 1674
 __draw_backend_fill:
 .. 1269, 1293, 1458, 1464, 1678, 1718
 __draw_backend_fillstroke:
 .. 1269, 1311, 1458, 1469, 1678, 1720
 __draw_backend_join_bevel:
 .. 1349, 1379, 1481, 1511, 1746, 1784
 __draw_backend_join_miter:
 .. 1349, 1375, 1481, 1507, 1746, 1780
 __draw_backend_join_round:
 .. 1349, 1377, 1481, 1509, 1746, 1782

<code>__draw_backend_lineto:nn</code>	<code>__draw_backend_transform_-</code>
.. 1229 , 1237 , 1424 , 1429 , 1633 , 1638	<code>decompose_auxiii:nnnnN</code>
<code>__draw_backend_linewidth:n</code>	 1561 , 1590 , 1598
.. 1349 , 1362 , 1481 , 1494 , 1746 , 1770	<code>\g__draw_draw_clip_bool</code> .. 1269 , 1678
<code>__draw_backend_literal:n</code>	<code>\g__draw_draw_eor_bool</code>
1212 , 1212 , 1213 , 1216 , 1217 , 1218 ,	... 1264 , 1281 , 1297 , 1304 , 1315 ,
1222 , 1223 , 1226 , 1228 , 1231 , 1239 ,	1326 , 1342 , 1453 , 1467 , 1472 , 1477
1247 , 1256 , 1270 , 1273 , 1274 , 1275 ,	<code>\g__draw_draw_path_int</code>
1276 , 1279 , 1285 , 1295 , 1302 , 1308 ,	1678
1313 , 1318 , 1319 , 1320 , 1321 , 1324 ,	
1330 , 1340 , 1346 , 1351 , 1364 , 1368 ,	
1370 , 1372 , 1374 , 1376 , 1378 , 1380 ,	
1383 , 1388 , 1393 , 1394 , 1395 , 1396 ,	
1397 , 1398 , 1399 , 1400 , 1401 , 1405 ,	
1406 , 1408 , 1409 , 1410 , 1411 , 1412 ,	
1416 , 1416 , 1417 , 1426 , 1431 , 1436 ,	
1446 , 1459 , 1461 , 1463 , 1466 , 1471 ,	
1476 , 1480 , 1483 , 1496 , 1500 , 1502 ,	
1504 , 1506 , 1508 , 1510 , 1512 , 1558 ,	
1623 , 1623 , 1624 , 1685 , 1704 , 1730	
<code>__draw_backend_miterlimit:n</code> ...	
.. 1349 , 1367 , 1481 , 1499 , 1746 , 1772	
<code>__draw_backend_moveto:nn</code>	
.. 1229 , 1229 , 1424 , 1424 , 1633 , 1633	
<code>__draw_backend_nonzero_rule:</code> ...	
.. 1264 , 1266 , 1453 , 1455 , 1674 , 1676	
<code>__draw_backend_path:n</code>	
..... 1678 , 1680 , 1712 , 1719 , 1721	
<code>\g__draw_backend_path_int</code> 1693 , 1710	
<code>\g__draw_backend_path_tl</code>	
.. 1633 , 1689 , 1705 , 1707 , 1734 , 1743	
<code>__draw_backend_rectangle:nnnn</code> ..	
.. 1229 , 1245 , 1424 , 1444 , 1633 , 1643	
<code>__draw_backend_scope_begin:</code> 1225 ,	
1225 , 1419 , 1422 , 1422 , 1625 , 1625	
<code>__draw_backend_scope_end:</code> 1225 ,	
1227 , 1421 , 1422 , 1423 , 1625 , 1626	
<code>__draw_backend_shift:nn</code>	
.. 1381 , 1386 , 1513 , 1556 , 1786 , 1794	
<code>__draw_backend_stroke:</code> 1269 , 1271 ,	
1291 , 1458 , 1460 , 1678 , 1711 , 1716	
<code>__draw_backend_transform:nnnn</code> ..	
..... 1381 , 1381 , 1402 , 1403 ,	
1404 , 1513 , 1513 , 1786 , 1786 , 1805	
<code>__draw_backend_transform_-</code>	
<code>aux:nnnn</code>	
1513 , 1527 , 1532	
<code>__draw_backend_transform_-</code>	
<code>decompose:nnnnN</code> . 1526 , 1561 , 1562	
<code>__draw_backend_transform_-</code>	
<code>decompose_auxi:nnnnN</code>	
1561 , 1566 , 1574	
<code>__draw_backend_transform_-</code>	
<code>decompose_auxii:nnnnN</code>	
1561 , 1578 , 1586	
	E
	<code>\errmessage</code>
	43
	<code>\evensidemargin</code>
	3005
	exp commands:
	<code>\exp_after:wN</code>
	557 , 846
	<code>\exp_args:Ne</code>
	627 ,
	681 , 879 , 1887 , 1945 , 1947 , 1971 ,
	1973 , 2362 , 2377 , 2428 , 2802 , 3001
	<code>\exp_args:Nf</code>
	1354 , 1486 , 2923
	<code>\exp_args:Nne</code>
	2769
	<code>\exp_args:NNf</code>
	252 , 300 , 357
	<code>\exp_args:Nno</code>
	3403
	<code>\exp_args:No</code>
	3409
	<code>\exp_not:N</code>
	588 ,
	594 , 595 , 596 , 627 , 629 , 630 , 633 ,
	634 , 639 , 2621 , 2623 , 2626 , 2632 ,
	2634 , 2637 , 2674 , 2675 , 2681 , 2682 ,
	2701 , 2706 , 3149 , 3151 , 3154 , 3184 ,
	3186 , 3189 , 3247 , 3248 , 3249 , 3254
	<code>\exp_not:n</code>
	53 , 101 , 121 , 159 ,
	968 , 2353 , 2358 , 2422 , 2591 , 2592 ,
	2606 , 2607 , 2747 , 2752 , 2763 , 2782
	<code>\ExplBackendFileDate</code>
	1
	F
	file commands:
	<code>\file_compare_timestamp:nNnTF</code> . 1959
	<code>\file_parse_full_name:nNNN</code> 1941 , 1969
	<code>\fmtversion</code>
	56
	fp commands:
	<code>\fp_compare:nNnTF</code>
	259 , 306 , 312 , 364 , 1537 , 1550 , 1600
	<code>\fp_eval:n</code>
	252 , 261 , 274 , 275 , 300 , 317 , 332 ,
	334 , 357 , 366 , 377 , 378 , 442 , 457 ,
	458 , 1100 , 1113 , 1114 , 1115 , 1539 ,
	1544 , 1545 , 1552 , 1567 , 1568 , 1569 ,
	1570 , 1579 , 1580 , 1581 , 1582 , 1591 ,
	1592 , 1593 , 1594 , 2419 , 2519 , 2796
	<code>\fp_new:N</code>
	325 , 326
	<code>\fp_set:Nn</code>
	305 , 308
	<code>\fp_use:N</code>
	311 , 315 , 320
	<code>\fp_zero:N</code>
	307
	<code>\c_zero_fp</code> 259 , 306 , 312 , 364 , 1537 , 1550

G

graphics commands:

\l_graphics_search_ext_seq
 1825, 1843, 1991, 2175

graphics internal commands:

\l_graphics_attr_tl 1845,
 1852, 1870, 1882, 1889, 1891, 1929
 _graphics_backend_dequote:w
 1847, 1888, 1925
 \l_graphics_backend_dir_str . . 1934
 \l_graphics_backend_ext_str . . 1934
 _graphics_backend_get_pagecount:n
 1840, 1840, 1983, 1983,
 2102, 2103, 2164, 2164, 2309, 2309
 _graphics_backend_getbb_auxi:n
 1847, 1861, 1878, 1880
 _graphics_backend_getbb_-
 auxi:nN 2107, 2111, 2120, 2122
 _graphics_backend_getbb_-
 auxii:n 1847, 1883, 1885
 _graphics_backend_getbb_-
 auxii:nnN . . 2107, 2125, 2128, 2130
 _graphics_backend_getbb_-
 auxiii:n 1847, 1887, 1893
 _graphics_backend_getbb_-
 auxiii:nNnn . 2107, 2126, 2129, 2131
 _graphics_backend_getbb_-
 auxiv:nnNnn . 2107, 2134, 2138, 2150
 _graphics_backend_getbb_-
 auxv:nNnn . . 2107, 2135, 2142, 2151
 _graphics_backend_getbb_-
 auxvi:nNnn 2154, 2156
 _graphics_backend_getbb_bmp:n .
 1993, 2004, 2107, 2115
 _graphics_backend_getbb_eps:n .
 1826, 1826, 1934,
 1939, 1956, 1993, 1993, 2256, 2256
 _graphics_backend_getbb_eps:nm
 1934
 _graphics_backend_getbb_eps:nn
 1945, 1957
 _graphics_backend_getbb_jpeg:n
 1847, 1863,
 1993, 2002, 2107, 2113, 2258, 2264
 _graphics_backend_getbb_jpg:n .
 1847, 1847, 1863, 1864, 1993, 1996,
 2002, 2003, 2004, 2107, 2107, 2113,
 2114, 2115, 2258, 2258, 2264, 2265
 _graphics_backend_getbb_-
 pagebox:w 2107, 2146, 2163
 _graphics_backend_getbb_pdf:n .
 1847, 1865, 1965,
 1993, 2005, 2107, 2116, 2266, 2266

_graphics_backend_getbb_png:n .
 1847, 1864,
 1993, 2003, 2107, 2114, 2258, 2265
 _graphics_backend_getbb_ps:n . .
 1826, 1827,
 1934, 1956, 1993, 1994, 2256, 2257
 _graphics_backend_getbb_svg:n .
 2177, 2177
 _graphics_backend_getbb_svg_-
 auxi:nNn . . . 2177, 2193, 2198, 2211
 _graphics_backend_getbb_svg_-
 auxii:w 2177, 2215, 2237, 2242
 _graphics_backend_getbb_svg_-
 auxiii:Nw 2177, 2225, 2243
 _graphics_backend_getbb_svg_-
 auxiv:Nw 2177, 2228, 2245
 _graphics_backend_getbb_svg_-
 auxv:Nw 2177, 2229, 2247
 _graphics_backend_getbb_svg_-
 auxvi:Nn 2177, 2244, 2246, 2248, 2249
 _graphics_backend_getbb_svg_-
 auxvii:w 2177, 2251, 2255
 _graphics_backend_include:nn . .
 2272, 2273, 2276, 2277
 _graphics_backend_include_-
 auxi:n 2014, 2029, 2037, 2039
 _graphics_backend_include_-
 auxii:nn . . . 2014, 2041, 2054, 2063
 _graphics_backend_include_-
 auxiii:nn 2014, 2061, 2064
 _graphics_backend_include_-
 bmp:n 2014, 2032
 _graphics_backend_include_-
 dequote:w 2288, 2299, 2307
 _graphics_backend_include_-
 eps:n 1828,
 1828, 1839, 1934, 1967, 1981,
 2014, 2014, 2025, 2272, 2272, 2274
 _graphics_backend_include_-
 jpeg:n . 1926, 1931, 2031, 2288, 2305
 _graphics_backend_include_-
 jpg:n 1926,
 1926, 1931, 1932, 1933, 2014,
 2026, 2031, 2032, 2033, 2288, 2306
 _graphics_backend_include_-
 jpseg:n 2014
 _graphics_backend_include_-
 pdf:n 1926,
 1932, 1971, 2014, 2034, 2272, 2275
 _graphics_backend_include_-
 png:n
 . . 1926, 1933, 2014, 2033, 2288, 2304
 _graphics_backend_include_ps:n
 1828, 1839,

[1934](#), [1981](#), [2014](#), [2025](#), [2272](#), [2274](#)
`\__graphics_backend_include_`
`svg:n` .. [2288](#), [2288](#), [2304](#), [2305](#), [2306](#)
`\l__graphics_backend_name_str` . [1934](#)
`\__graphics_bb_restore:nTF`
..... [1882](#), [2153](#), [2179](#)
`\__graphics_bb_save:n` [1891](#), [2161](#), [2206](#)
`\l__graphics_decodearray_str` ...
..... [1854](#), [1855](#),
[1867](#), [1900](#), [1906](#), [1907](#), [2007](#), [2047](#),
[2048](#), [2088](#), [2092](#), [2093](#), [2118](#), [2268](#)
`\__graphics_extract_bb:n`
..... [2000](#), [2009](#), [2262](#), [2270](#)
`\l__graphics_final_name_str` .. [1964](#)
`\__graphics_get_pagecount:n`
..... [1840](#), [2103](#), [2309](#)
`\l__graphics_interpolate_bool` ...
..... [1856](#), [1869](#), [1898](#), [1910](#),
[2008](#), [2049](#), [2086](#), [2096](#), [2119](#), [2269](#)
`\l__graphics_llx_dim`
..... [1833](#), [2019](#), [2078](#), [2185](#), [2282](#)
`\l__graphics_lly_dim`
..... [1834](#), [2020](#), [2079](#), [2186](#), [2283](#)
`\l__graphics_page_int` [1849](#), [1873](#),
[1874](#), [1915](#), [1916](#), [1998](#), [2045](#), [2046](#),
[2072](#), [2073](#), [2109](#), [2124](#), [2125](#), [2260](#)
`\l__graphics_pagebox_tl` ... [1850](#),
[1872](#), [1917](#), [1918](#), [1999](#), [2043](#), [2044](#),
[2074](#), [2076](#), [2110](#), [2133](#), [2134](#), [2261](#)
`\l__graphics_pdf_str`
.. [1858](#), [1859](#), [1875](#), [1876](#), [1901](#), [1912](#)
`\__graphics_read_bb:n`
.. [1826](#), [1827](#), [1993](#), [1994](#), [2256](#), [2257](#)
`\l__graphics_tmp_box`
.. [1920](#), [1922](#), [1923](#), [2158](#), [2159](#), [2160](#)
`\l__graphics_tmp_dim` [2252](#), [2253](#)
`\l__graphics_tmp_ior`
..... [2181](#), [2182](#), [2189](#), [2208](#)
`\g__graphics_track_int`
..... [2013](#), [2066](#), [2067](#)
`\l__graphics_transgroup_bool` ...
..... [1846](#), [1851](#), [1868](#), [1899](#),
[1908](#), [2012](#), [2028](#), [2036](#), [2087](#), [2094](#)
`\l__graphics_urx_dim`
... [1835](#), [1922](#), [2021](#), [2080](#), [2159](#),
[2187](#), [2191](#), [2194](#), [2202](#), [2284](#), [2297](#)
`\l__graphics_ury_dim`
[1836](#), [1923](#), [2022](#), [2081](#), [2160](#), [2188](#),
[2196](#), [2199](#), [2203](#), [2285](#), [2290](#), [2298](#)
group commands:
`\group_begin:` [195](#), [214](#)
`\group_end:` [203](#)

H

hbox commands:

`\hbox:n` [2292](#), [2436](#), [2443](#),
[2810](#), [2821](#), [2929](#), [2932](#), [3008](#), [3014](#)
`\hbox_overlap_right:n` [247](#),
[279](#), [295](#), [336](#), [352](#), [380](#), [464](#), [1407](#), [1615](#)
`\hbox_set:Nn` .. [1920](#), [2158](#), [3000](#), [3032](#)
`\hbox_set:Nw` [2983](#)
`\hbox_set_end:` [2998](#)
`\hbox_unpack:N` [3119](#)
hook commands:
`\hook_gput_code:nnn` .. [59](#), [3096](#), [3098](#)

I

int commands:

`\int_compare:nNnTF` [1873](#), [1915](#), [2045](#),
[2072](#), [2124](#), [2461](#), [2672](#), [2700](#), [3091](#)
`\int_const:Nn`
..... [477](#), [1889](#), [1986](#), [2067](#), [2166](#)
`\int_eval:n` [497](#), [507](#), [677](#), [686](#), [699](#),
[701](#), [705](#), [718](#), [2485](#), [2489](#), [2650](#),
[2675](#), [2682](#), [2695](#), [2839](#), [2847](#), [2852](#)
`\int_gincr:N` [221](#),
[387](#), [1684](#), [1729](#), [2066](#), [2323](#), [2390](#),
[2735](#), [2768](#), [2939](#), [3017](#), [3225](#), [3247](#)
`\int_gset:Nn` [196](#), [215](#), [2566](#), [3080](#)
`\int_gset_eq:NN` [204](#), [3018](#), [3248](#)
`\int_if_exist:NnTF` [2056](#)
`\int_if_odd:nTF` [3003](#)
`\int_max:nn` [2168](#)
`\int_new:N` [187](#), [188](#), [434](#), [472](#), [1710](#),
[2013](#), [2920](#), [2952](#), [2954](#), [3222](#), [3237](#)
`\int_set:Nn` [3092](#)
`\int_set_eq:NN` [192](#), [211](#)
`\int_step_function:nnnN` [703](#)
`\int_use:N` [389](#),
[420](#), [630](#), [639](#), [787](#), [818](#), [881](#), [887](#),
[888](#), [942](#), [943](#), [952](#), [976](#), [1687](#), [1693](#),
[1700](#), [1732](#), [1740](#), [1874](#), [1916](#), [1929](#),
[1987](#), [2046](#), [2059](#), [2071](#), [2073](#), [2169](#),
[2392](#), [2397](#), [2770](#), [2775](#), [2943](#), [2951](#),
[3022](#), [3122](#), [3228](#), [3236](#), [3254](#), [3265](#)
`\int_value:w`
..... [2621](#), [2632](#), [2650](#), [3149](#), [3184](#)
`\int_zero:N` ... [1849](#), [1998](#), [2109](#), [2260](#)
ior commands:
`\ior_close:N` [2208](#)
`\ior_if_eof:NnTF` [2182](#)
`\ior_map_break:` [2204](#)
`\ior_open:Nn` [2181](#)
`\ior_str_map_inline:Nn` [2189](#)

K

kernel internal commands:

`\__kernel_backend_align_begin:` . .
 [76](#), [76](#), [232](#), [256](#), [271](#)
`\__kernel_backend_align_end:` . . .
 [76](#), [82](#), [246](#), [264](#), [278](#)
`\__kernel_backend_first_shipout:`n
 [54](#), [58](#), [61](#), [63](#), [73](#), [627](#), [2888](#)
`\g__kernel_backend_header_bool` . .
 [71](#), [625](#)
`\__kernel_backend_literal:n`
 [51](#), [51](#), [52](#), [53](#), [66](#), [69](#), [74](#), [78](#),
 [85](#), [88](#), [90](#), [174](#), [177](#), [179](#), [181](#), [185](#),
 [361](#), [374](#), [532](#), [539](#), [577](#), [582](#), [647](#),
 [783](#), [841](#), [996](#), [1001](#), [1007](#), [1012](#),
 [1062](#), [1088](#), [1521](#), [1522](#), [1523](#), [1534](#),
 [1541](#), [1547](#), [1612](#), [1617](#), [1830](#), [2016](#),
 [2058](#), [2068](#), [2279](#), [2294](#), [2727](#), [2839](#),
 [2843](#), [2848](#), [2853](#), [2890](#), [3220](#), [3267](#)
`\__kernel_backend_literal_page:n`
 [113](#), [113](#),
 [123](#), [176](#), [176](#), [2721](#), [2723](#), [2858](#), [2860](#)
`\__kernel_backend_literal_pdf:n` .
 [4](#), [6](#), [93](#), [93](#), [103](#), [173](#), [173](#), [175](#),
 [287](#), [344](#), [1416](#), [1417](#), [3382](#), [3392](#), [3427](#)
`\__kernel_backend_literal_-`
 `postscript:n` [65](#),
 [65](#), [67](#), [79](#), [80](#), [84](#), [233](#), [234](#), [236](#),
 [237](#), [245](#), [257](#), [272](#), [1212](#), [2463](#), [2475](#)
`\__kernel_backend_literal_svg:n` .
 [184](#), [184](#), [186](#), [191](#), [202](#), [210](#), [220](#),
 [388](#), [390](#), [407](#), [825](#), [1623](#), [1806](#), [1817](#)
`\__kernel_backend_matrix:n`
 [151](#), [151](#), [161](#), [309](#), [330](#), [1516](#), [1609](#)
`\__kernel_backend_postscript:n` . .
 [68](#), [68](#), [70](#), [535](#), [1064](#),
 [1066](#), [1068](#), [1070](#), [1074](#), [2316](#), [2367](#),
 [2382](#), [2402](#), [2436](#), [2443](#), [2929](#), [2935](#),
 [2940](#), [2976](#), [3008](#), [3015](#), [3019](#), [3033](#),
 [3061](#), [3104](#), [3111](#), [3118](#), [3125](#), [3321](#)
`\__kernel_backend_scope:n`
 [189](#), [218](#), [223](#), [417](#), [422](#), [1092](#),
 [1120](#), [1630](#), [1675](#), [1677](#), [1697](#), [1737](#),
 [1759](#), [1771](#), [1773](#), [1775](#), [1777](#), [1779](#),
 [1781](#), [1783](#), [1785](#), [1788](#), [1796](#), [3454](#)
`\__kernel_backend_scope_begin:` . .
 [87](#), [87](#), [133](#), [133](#), [178](#), [178](#), [189](#), [189](#),
 [231](#), [255](#), [270](#), [286](#), [303](#), [329](#), [343](#),
 [360](#), [373](#), [1422](#), [1607](#), [1625](#), [1629](#), [1804](#)
`\__kernel_backend_scope_begin:n` .
 [189](#), [208](#), [217](#), [409](#), [437](#), [450](#)
`\__kernel_backend_scope_end:` . . .
 [87](#), [89](#), [133](#), [142](#),
 [178](#), [180](#), [189](#), [198](#), [248](#), [266](#), [280](#),

[296](#), [323](#), [337](#), [353](#), [369](#), [381](#), [432](#),
[446](#), [465](#), [1423](#), [1619](#), [1626](#), [1632](#), [1818](#)
`\g__kernel_backend_scope_int` . . .
 [187](#), [194](#), [196](#), [201](#), [205](#), [213](#), [215](#), [221](#)
`\l__kernel_backend_scope_int` . . .
 [187](#), [193](#), [206](#), [212](#)
`\g__kernel_clip_path_int`
 [385](#), [1684](#), [1687](#), [1700](#), [1729](#), [1732](#), [1740](#)
`\__kernel_color_backend_stack_-`
 `init:Nnn` [475](#), [475](#), [3364](#)
`\__kernel_color_backend_stack_-`
 `pop:n` [489](#), [499](#), [567](#), [3396](#)
`\__kernel_color_backend_stack_-`
 `push:nn` [489](#),
 [489](#), [564](#), [1030](#), [1042](#), [3385](#), [3430](#)
`\__kernel_dependency_version_-`
 `check:Nn` [1](#)
`\__kernel_dependency_version_-`
 `check:nn` [31](#), [33](#)
`\__kernel_file_name_quote:n`
 [1947](#), [1973](#)

L

lua commands:

`\lua_load_module:n` [1206](#)

M

`\MessageBreak` [45](#)

mode commands:

`\mode_if_horizontal:TF` [3082](#), [3089](#)
`\mode_if_math:TF` [2980](#)

msg commands:

`\msg_error:nnn` [2183](#)

O

`\oddsidemargin` [3004](#)

opacity internal commands:

`\__opacity_backend:nn`
 [3447](#), [3448](#), [3450](#), [3452](#), [3453](#)
`\__opacity_backend:nnn` [3300](#), [3302](#),
 [3303](#), [3307](#), [3314](#), [3319](#), [3345](#), [3352](#)
`\__opacity_backend_fill:n`
 [3300](#), [3305](#), [3401](#), [3401](#), [3447](#), [3449](#)
`\__opacity_backend_fill_stroke:nn`
 [3401](#), [3403](#), [3409](#), [3413](#), [3435](#), [3440](#)
`\l__opacity_backend_fill_tl`
 [3370](#), [3376](#), [3410](#), [3418](#)
`\__opacity_backend_reset:`
 [3300](#), [3338](#),
 [3374](#), [3389](#), [3399](#), [3400](#), [3435](#), [3441](#),
 [3442](#), [3443](#), [3447](#), [3455](#), [3456](#), [3457](#)
`\__opacity_backend_reset_fill:` . .
 [3300](#), [3340](#), [3343](#),
 [3374](#), [3399](#), [3435](#), [3442](#), [3447](#), [3456](#)

__opacity_backend_reset_stroke:
 3300, 3341, 3350,
 3374, 3400, 3435, 3443, 3447, 3457
 __opacity_backend_select:n
 3300, 3300, 3374,
 3374, 3416, 3435, 3439, 3447, 3447
 \c__opacity_backend_stack_int ...
 3359, 3385, 3396, 3430
 __opacity_backend_stroke:n
 3300, 3312, 3401, 3407, 3447, 3451
 \l__opacity_backend_stroke_tl ...
 3370, 3377, 3405, 3419

P

pdf commands:

\pdf_object_if_exist:nTF 896, 962, 980
 \pdf_object_new:n
 887, 898, 942, 964, 982
 \pdf_object_ref:n
 841, 911, 975, 990, 1008, 1013
 \pdf_object_ref_last:
 864, 889, 892, 948
 \pdf_object_unnamed_write:nn ...
 871, 918, 974, 989
 \pdf_object_write:nnn
 888, 899, 943, 965, 983

pdf internal commands:

__pdf_backend:n
 2726, 2726, 2728, 2730, 2732, 2746,
 2751, 2760, 2780, 2812, 2813, 2823
 __pdf_backend_annotation:nnnn 3284
 __pdf_backend_annotation_last: 3285
 __pdf_backend_bdc:nn 2493, 2493,
 2720, 2720, 2857, 2857, 2882, 2882
 __pdf_backend_catalog_gput:nn ..
 2318, 2318,
 2536, 2536, 2729, 2729, 2865, 2865
 __pdf_backend_compress_objects:n
 2459, 2471,
 2641, 2652, 2838, 2840, 2876, 2877
 __pdf_backend_compresslevel:n ..
 2459, 2459,
 2641, 2641, 2838, 2838, 2876, 2876
 __pdf_backend_destination:nn ...
 2400, 2400,
 2499, 2499, 2778, 2778, 2863, 2863
 __pdf_backend_destination:nnnn .
 2400, 2426,
 2499, 2522, 2778, 2800, 2863, 2864
 __pdf_backend_destination_-
 aux:nnnn
 2400, 2428, 2431, 2778, 2802, 2805
 __pdf_backend_emc: .. 2493, 2495,
 2720, 2722, 2857, 2859, 2882, 2883

__pdf_backend_info_gput:nn
 2318, 2320,
 2536, 2546, 2729, 2731, 2865, 2866
 __pdf_backend_objcompresslevel:n
 2641, 2655, 2656, 2658
 __pdf_backend_object_id:n
 2322, 2325,
 2557, 2575, 2734, 2737, 2867, 2869
 \g__pdf_backend_object_int
 2323, 2390, 2392,
 2397, 2566, 2735, 2768, 2770, 2775
 __pdf_backend_object_last:
 2396, 2396,
 2619, 2619, 2774, 2774, 2867, 2874
 __pdf_backend_object_new:
 2322, 2322,
 2557, 2557, 2734, 2734, 2867, 2867
 __pdf_backend_object_now:nn ...
 2388, 2388, 2395, 2608, 2608, 2618,
 2766, 2766, 2773, 2867, 2872, 2873
 \g__pdf_backend_object_prop
 2556, 2733
 __pdf_backend_object_ref:n
 2322, 2324, 2325, 2329, 2557, 2574,
 2734, 2736, 2737, 2741, 2867, 2868
 __pdf_backend_object_write:nn ..
 2576, 2585, 2587, 2616, 2867
 __pdf_backend_object_write:nnn .
 2326, 2326, 2332, 2576, 2576, 2605,
 2738, 2738, 2743, 2867, 2870, 2871
 __pdf_backend_object_write_-
 array:nn ... 2326, 2350, 2738, 2744
 __pdf_backend_object_write_-
 aux:nnn 2326, 2328, 2333, 2391
 __pdf_backend_object_write_-
 dict:nn 2326, 2355, 2738, 2749
 __pdf_backend_object_write_-
 fstream:nn . 2326, 2360, 2738, 2754
 __pdf_backend_object_write_-
 fstream:nnn 2363, 2365
 __pdf_backend_object_write_-
 stream:nn .. 2326, 2375, 2738, 2756
 __pdf_backend_object_write_-
 stream:nnn 2326, 2378, 2380
 __pdf_backend_object_write_-
 stream:nnnn . 2738, 2755, 2757, 2758
 __pdf_backend_pageobject_ref:n .
 2398, 2398,
 2630, 2630, 2776, 2776, 2867, 2875
 __pdf_backend_pagesize_gset:nn .
 2886, 2886, 2905, 2905, 2912, 2912
 __pdf_backend_pdfmark:n
 2315, 2315, 2317, 2319, 2321, 2335,
 2352, 2357, 2403, 2447, 2494, 2496

_pdf_backend_version_major: ...	pdf.rightboundary	3593
... 2485, 2491, 2491, 2697, 2697,	pdf.save.linkll	3522
2847, 2848, 2855, 2855, 2880, 2880	pdf.save.linkur	3522
_pdf_backend_version_major_-	pdf.save.ll	3522
gset:n	pdf.save.ur	3522
2483, 2483,	pdf.tmpa	3558
2669, 2669, 2845, 2845, 2878, 2878	pdf.tmpb	3558
_pdf_backend_version_minor: ...	pdf.tmpc	3558
... 2489, 2491, 2492, 2697, 2710,	pdf.tmpd	3558
2852, 2853, 2855, 2856, 2880, 2881	pdf.urx	3522
_pdf_backend_version_minor_-	pdf.ury	3522
gset:n	pdfannot internal commands:	
2483, 2487,	_pdfannot_backend:n	3219, 3219,
2669, 2686, 2845, 2850, 2878, 2879	3221, 3226, 3250, 3263, 3268, 3269	
_pdf_exp_not_i:nn	\l_pdfannot_backend_breaklink_-	
2576, 2595, 2600, 2606	pdfmark_tl	2957, 3025, 3116
_pdf_exp_not_ii:nn	_pdfannot_backend_breaklink_-	
2576, 2596, 2601, 2607	postscript:n	2959, 2959, 3009, 3011, 3117
pdf.baselineskip	_pdfannot_backend_breaklink_-	
3835	usebox:N	2960, 2960, 3010, 3119
pdf.bordertracking	\l_pdfannot_backend_content_box	2918,
3593	2983, 3007, 3010, 3012, 3041, 3052	
pdf.bordertracking.begin	_pdfannot_backend_generic:nnnn	2921, 2921, 3134,
3593	3134, 3223, 3223, 3272, 3272, 3284	
pdf.bordertracking.continue	_pdfannot_backend_generic_-	
3593	aux:nnnn	2921, 2923, 2926
pdf.bordertracking.end	\g_pdfannot_backend_int	2920, 2939, 2943, 2951, 3017, 3018,
3593	3222, 3225, 3228, 3236, 3247, 3249	
pdf.bordertracking.endpage	_pdfannot_backend_last:	2950, 2950, 3147,
3593	3147, 3235, 3235, 3273, 3273, 3285	
pdf.breaklink	_pdfannot_backend_link:nw	2961
3731	_pdfannot_backend_link_aux:nw	2961
pdf.breaklink.write	_pdfannot_backend_link_begin:n	3238, 3240, 3244, 3245
3731	_pdfannot_backend_link_-	
pdf.brokenlink.dict	begin:nnnw	3158, 3159, 3161, 3162, 3274, 3276
3593	_pdfannot_backend_link_-	
pdf.brokenlink.rect	begin:nw	2963, 2967, 2968
3593	_pdfannot_backend_link_begin_-	
pdf.brokenlink.skip	aux:nw	2971, 2973
3593	_pdfannot_backend_link_begin_-	
pdf.count	goto:nnw	2961, 2961,
3731	3158, 3158, 3238, 3238, 3274, 3274	
pdf.currentrect	_pdfannot_backend_link_begin_-	
3731	user:nnw	2961, 2966,
pdf.cvs	\g_pdfannot_backend_link_bool	3158, 3160, 3238, 3243, 3274, 3275
3515	2956, 2970, 2975, 2990, 3028	
pdf.dest.anchor		
3558		
pdf.dest.point		
3558		
pdf.dest.x		
3558		
pdf.dest.y		
3558		
pdf.dest2device		
3558		
pdf.dev.x		
3558		
pdf.dev.y		
3558		
pdf.dvi.pt		
3515		
pdf.globaldict		
3512		
pdf.leftboundary		
3593		
pdf.linkdp.pad		
3519		
pdf.linkht.pad		
3519		
pdf.linkmargin		
3519		
pdf.llx		
3522		
pdf.lly		
3522		
pdf.originx		
3593		
pdf.originy		
3593		
pdf.outerbox		
3835		
pdf.pdfmark		
3835		
pdf.pdfmark.dict		
3835		
pdf.pdfmark.good		
3835		
pdf.pt.dvi		
3515		
pdf.rect		
3522		
pdf.rect.ht		
3515		

<code>\g_pdfannot_backend_link_dict_-</code> <code>tl</code> 2953, 2978, 3023	
<code>_pdfannot_backend_link_end:</code> 2961, 2988, 3158, 3173, 3238, 3262, 3274, 3277	
<code>_pdfannot_backend_link_end_-</code> <code>aux:</code> 2961, 2991, 2993	
<code>\g_pdfannot_backend_link_int</code> 2952, 3018, 3022, 3122, 3237, 3248, 3254, 3265	
<code>_pdfannot_backend_link_last:</code> 3121, 3121, 3182, 3182, 3264, 3264, 3278, 3278	
<code>_pdfannot_backend_link_-</code> <code>margin:n</code> 3123, 3123, 3193, 3193, 3266, 3266, 3279, 3279	
<code>\g_pdfannot_backend_link_math_-</code> <code>bool</code> ... 2955, 2981, 2982, 2985, 2995	
<code>_pdfannot_backend_link_minima:</code> 2961, 2999, 3030	
<code>_pdfannot_backend_link_off:</code> 3130, 3131, 3203, 3210, 3268, 3269, 3280, 3281	
<code>_pdfannot_backend_link_on:</code> 3130, 3130, 3203, 3203, 3268, 3268, 3280, 3280	
<code>_pdfannot_backend_link_-</code> <code>outerbox:n</code> 2961, 3001, 3059	
<code>\g_pdfannot_backend_link_sf_int</code> 2954, 3080, 3091, 3092	
<code>_pdfannot_backend_link_sf_-</code> <code>restore:</code> ... 2961, 2984, 3027, 3087	
<code>_pdfannot_backend_link_sf_-</code> <code>save:</code> 2961, 2979, 2997, 3078	
<code>\l_pdfannot_backend_model_box</code> 2919, 3000, 3032, 3040, 3051, 3066, 3068	
pdfmanagement commands:	
<code>\pdfmanagement_add:nnn</code> 861, 3367, 3378, 3420, 3423	
<code>\pdfmanagement_if_active_p:</code> 856, 857, 3360, 3361, 3436, 3437	
peek commands:	
<code>\peek_meaning:NTF</code> 2224, 2227	
<code>\peek_remove_spaces:n</code> 2222	
prg commands:	
<code>\prg_replicate:nn</code> 200, 675, 696, 706, 924	
prop commands:	
<code>\prop_gput:Nnn</code> 633, 891	
<code>\prop_if_in:NnTF</code> 604	
<code>\prop_item:Nn</code> 607	
<code>\prop_new:N</code> 585, 2556, 2733	
<code>\ProvidesExplFile</code> 2	
	Q
	quark commands:
	<code>\quark_if_recursion_tail_stop:n</code> 603
	<code>\q_recursion_stop</code> 596
	<code>\q_recursion_tail</code> 595
	<code>\q_stop</code> 558, 560, 847
	S
	scan commands:
	<code>\scan_stop:</code> 136, 145, 507, 2252, 2255, 2520, 2534, 2650, 2667, 2675, 2682, 2695, 3176, 3201
	scan internal commands:
	<code>\s_color_stop</code> 686, 687, 691, 695, 708, 711, 715, 719, 733, 925, 954, 958, 1106, 1108
	<code>\s_graphics_stop</code> 1888, 1925, 2217, 2232, 2239, 2243, 2245, 2247, 2299, 2307
	separation 3509
	seq commands:
	<code>\seq_set_from_clist:Nn</code> 1825, 1843, 1991, 2175
	shipout commands:
	<code>\l_shipout_box</code> 3100, 3102, 3110
	skip commands:
	<code>\skip_horizontal:n</code> 249, 297, 354
	str commands:
	<code>\c_hash_str</code> 420, 1693, 1700, 1740
	<code>\c_percent_str</code> 1126, 1127, 1128
	<code>\str_case:nn</code> 930, 2339, 2589
	<code>\str_case:nnTF</code> 2407, 2508, 2785
	<code>\str_convert_pdfname:n</code> . 634, 654, 880
	<code>\str_if_empty:NTF</code> 1858, 1875
	<code>\str_if_empty_p:N</code> 1901
	<code>\str_if_eq:nnTF</code> 816, 1519, 3415
	<code>\str_new:N</code> 1936, 1937, 1938
	<code>\str_tail:N</code> 1950, 1976
	sys commands:
	<code>\sys_if_shell:TF</code> 1934
	<code>\sys_shell_now:n</code> 1961
	T
	TeX and L ^A T _E X 2 _ε commands:
	<code>\@ifl@t@r</code> 54, 56
	<code>\current@color</code> 28
	<code>\special</code> 2
	tex commands:
	<code>\tex_afterassignment:D</code> 2251
	<code>\tex_baselineskip:D</code> 3072
	<code>\tex_endinput:D</code> 49
	<code>\tex_global:D</code> 2643, 2660, 2674, 2681, 2688

<code>\tex_immediate:D</code>	2169
..... 1895, 2579, 2582, 2611, 2614	
<code>\tex_luatexversion:D</code>	2111
..... 2672, 2700	
<code>\tex_pageheight:D</code>	3509
..... 2908	
<code>\tex_pagewidth:D</code>	3067
..... 2907	
<code>\tex_pdfannot:D</code>	
..... 3140	
<code>\tex_pdfcatalog:D</code>	
..... 2542	
<code>\tex_pdfcolorstack:D</code>	
..... 495, 505	
<code>\tex_pdfcolorstackinit:D</code>	
..... 483	
<code>\tex_pdfcompresslevel:D</code>	
..... 2648	
<code>\tex_pdfdest:D</code>	
..... 2505, 2528	
<code>\tex_pdfendlink:D</code>	
..... 3179	
<code>\tex_pdfextension:D</code> .	
96, 106, 116,	
126, 136, 145, 154, 164, 492, 502,	
2502, 2525, 2539, 2549, 2560, 2579,	
2611, 3137, 3165, 3176, 3205, 3212	
<code>\tex_pdffeedback:D</code>	
... 480, 2568, 2623, 2634, 3151, 3186	
<code>\tex_pdfinfo:D</code>	
..... 2552	
<code>\tex_pdflastannot:D</code>	
..... 3154	
<code>\tex_pdflastlink:D</code>	
..... 3189	
<code>\tex_pdflastobj:D</code>	
..... 2571, 2626	
<code>\tex_pdflastximage:D</code>	
..... 1890, 1921	
<code>\tex_pdflastximagepages:D</code>	
..... 1987	
<code>\tex_pdflinkmargin:D</code>	
..... 3199	
<code>\tex_pdfliteral:D</code> ...	
99, 109, 119, 129	
<code>\tex_pdfmajorversion:D</code>	
..... 2679, 2681, 2705, 2706	
<code>\tex_pdfminorversion:D</code> ...	
..... 2693, 2717	
<code>\tex_pdfobj:D</code>	
..... 2563, 2582, 2614	
<code>\tex_pdfobjcompresslevel:D</code> ...	
..... 2665	
<code>\tex_pdfpageref:D</code>	
..... 2637	
<code>\tex_pdfrefximage:D</code>	
..... 1921, 1928	
<code>\tex_pdfrestore:D</code>	
..... 148	
<code>\tex_pdfrunninglinkoff:D</code>	
..... 3215	
<code>\tex_pdfrunninglinkon:D</code>	
..... 3208	
<code>\tex_pdfsave:D</code>	
..... 139	
<code>\tex_pdfsetmatrix:D</code>	
..... 157, 167	
<code>\tex_pdfstartlink:D</code>	
..... 3168	
<code>\tex_pdfvariable:D</code>	
..... 2645,	
2662, 2674, 2690, 2701, 2714, 3196	
<code>\tex_pdfximage:D</code>	
..... 1895, 1985	
<code>\tex_spacefactor:D</code>	
..... 3083, 3092	
<code>\tex_special:D</code>	
..... 51	
<code>\tex_the:D</code>	
..... 1890, 2701, 2706, 2712	
<code>\tex_vss:D</code>	
..... 2437, 2444, 2815, 2834	
<code>\tex_XeTeXpdfffile:D</code>	
..... 2120	
<code>\tex_XeTeXpdfpagecount:D</code>	2169
<code>\tex_XeTeXpicfile:D</code>	2111
<code>TeXcolorseparation</code>	3509
<code>\textwidth</code>	3067
tl commands:	
<code>\c_space_tl</code>	311, 316, 319, 546,
547, 548, 560, 590, 595, 639, 742,	
819, 831, 1043, 1669, 1832, 1833,	
1834, 1835, 2018, 2019, 2020, 2021,	
2073, 2076, 2078, 2079, 2080, 2081,	
2146, 2281, 2282, 2283, 2284, 2628,	
2639, 3023, 3156, 3191, 3228, 3255	
<code>\tl_clear:N</code>	1850, 1867,
1999, 2007, 2110, 2118, 2261, 2268	
<code>\tl_gclear:N</code>	1707, 1743
<code>\tl_gset:Nn</code>	1666, 2978
<code>\tl_if_blank:nTF</code>	485, 588,
690, 707, 714, 732, 875, 957, 2145, 2220	
<code>\tl_if_empty:NTF</code> .	1669, 1854, 1906,
1917, 2043, 2047, 2074, 2092, 2133	
<code>\tl_if_empty:nTF</code>	969, 1763
<code>\tl_if_empty_p:N</code>	1900, 2088
<code>\tl_new:N</code>	542,
543, 1673, 1845, 2953, 2957, 3370, 3371	
<code>\tl_set:Nn</code> 544, 545, 562, 563, 1029,	
1041, 1852, 1870, 1964, 2958, 3116,	
3372, 3373, 3376, 3377, 3418, 3419	
<code>\tl_to_str:n</code>	2216, 2238
<code>\tl_use:N</code>	774, 904
token commands:	
<code>\c_math_toggle_token</code>	2986, 2996
U	
use commands:	
<code>\use:N</code>	48, 533, 2348, 2740, 2769
<code>\use:n</code>	63, 859, 885,
940, 1097, 1110, 1354, 1486, 1564,	
1576, 1588, 1748, 2140, 2213, 2235	
<code>\use_none:n</code>	1765
<code>\use_none:nnn</code>	3095
V	
<code>\value</code>	3003
vbox commands:	
<code>\vbox_set:Nn</code>	3102
<code>\vbox_to_zero:n</code> 2433, 2440, 2807, 2818	
<code>\vbox_unpack_drop:N</code>	3110