

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}{2026-06-19}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2026-06-19}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2026-06-19}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2026-06-19}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2026-06-19}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2026-06-19}{ }
25   {L3 backend support: XeTeX}
26 </xetex>
```

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

```
27 \cs_if_exist:NTF \__kernel_dependency_version_check:nn
28   {
29     \__kernel_dependency_version_check:nn {2023-10-10}
30     <dvipdfmx>    {l3backend-dvipdfmx.def}
31     <dvips>       {l3backend-dvips.def}
32     <dvisvgm>     {l3backend-dvisvgm.def}
33     <luatex>      {l3backend-luatex.def}
34     <pdftex>      {l3backend-pdftex.def}
35     <xetex>       {l3backend-xetex.def}
```

```

36 }
37 {
38   \cs_if_exist_use:cF { @latex@error } { \errmessage }
39   {
40     Mismatched-LaTeX-support-files-detected. \MessageBreak
41     Loading~aborted!
42   }
43   { \use:c { @ehd } }
44   \tex_endinput:D
45 }

```

The order of the backend code here is such that we get somewhat logical outcomes in terms of code sharing whilst keeping things readable. (Trying to mix all of the code by concept is almost unmanageable.) The key parts which are shared are

- Color support is either dvips-like or LuaTeX/pdfTeX-like.
- LuaTeX/pdfTeX and dvipdfmx/X_YTeX share drawing routines.
- X_YTeX is the same as dvipdfmx other than image size extraction so takes most of the same code.

`__kernel_backend_literal:e` The one shared function for all backends is access to the basic `\special` primitive: it has slightly odd expansion behavior so a wrapper is provided.

```

46 \cs_new_eq:NN __kernel_backend_literal:e \tex_special:D
47 \cs_new_protected:Npn __kernel_backend_literal:n #1
48 { __kernel_backend_literal:e { \exp_not:n {#1} } }

```

(End of definition for `__kernel_backend_literal:e`.)

`__kernel_backend_first_shipout:n` We need to write at first shipout in a few places. As we want to use the most up-to-date method,

```

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

```

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

1.1 dvips backend

59 `<*dvips>`

`__kernel_backend_literal_postscript:n` Literal PostScript can be included using a few low-level formats. Here, we use the form with no positioning: this is overall more convenient as a wrapper. Note that this does require that where position is important, an appropriate wrapper is included.

```

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

```

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

`_kernel_backend_postscript:n` PostScript data that does have positioning, and also applying a shift to `SDict` (which is not done automatically by `ps:` or `ps::`, in contrast to `!` or `"`).

```

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

```

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

PostScript for the header: a small saving but makes the code clearer. This is held until the start of shipout such that a document with no actual output does not write anything.

```

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

```

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

```

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

```

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

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

```

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

```

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

```

86 </dvips>

```

1.2 LuaTeX and pdfTeX backends

87 `<*luatex | pdftex>`

Both LuaTeX and pdfTeX write PDFs directly rather than via an intermediate file. Although there are similarities, the move of LuaTeX to have more code in Lua means we create two independent files using shared DocStrip code.

This is equivalent to `\special{pdf:}` but the engine can track it. Without the `direct` keyword everything is kept in sync: the transformation matrix is set to the current point automatically. Note that this is still inside the text (BT ...ET block).

```

88 \cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1
89 {
90   <*luatex>
91     \tex_pdfextension:D literal
92   </luatex>
93   <*pdftex>
94     \tex_pdfliteral:D
95   </pdftex>
96     { \exp_not:n {#1} }
97   }
98 \cs_new_protected:Npn \__kernel_backend_literal_pdf:e #1
99 {
100   <*luatex>
101     \tex_pdfextension:D literal
102   </luatex>
103   <*pdftex>
104     \tex_pdfliteral:D
105   </pdftex>
106     {#1}
107   }

```

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

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

```

108 \cs_new_protected:Npn \__kernel_backend_literal_page:n #1
109 {
110   <*luatex>
111     \tex_pdfextension:D literal ~
112   </luatex>
113   <*pdftex>
114     \tex_pdfliteral:D
115   </pdftex>
116     page { \exp_not:n {#1} }
117   }
118 \cs_new_protected:Npn \__kernel_backend_literal_page:e #1
119 {
120   <*luatex>
121     \tex_pdfextension:D literal ~
122   </luatex>
123   <*pdftex>
124     \tex_pdfliteral:D
125   </pdftex>
126     page {#1}
127   }

```

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

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

```

\_kernel_backend_scope_begin:
\_kernel_backend_scope_end:
128 \cs_new_protected:Npn \_kernel_backend_scope_begin:
129 {
130 <*luatex>
131 \tex_pdfextension:D save \scan_stop:
132 </luatex>
133 <*pdftex>
134 \tex_pdfsave:D
135 </pdftex>
136 }
137 \cs_new_protected:Npn \_kernel_backend_scope_end:
138 {
139 <*luatex>
140 \tex_pdfextension:D restore \scan_stop:
141 </luatex>
142 <*pdftex>
143 \tex_pdfrestore:D
144 </pdftex>
145 }

```

(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
146 \cs_new_protected:Npn \_kernel_backend_matrix:n #1
147 {
148 <*luatex>
149 \tex_pdfextension:D setmatrix
150 </luatex>
151 <*pdftex>
152 \tex_pdfsetmatrix:D
153 </pdftex>
154 { \exp_not:n {#1} }
155 }
156 \cs_new_protected:Npn \_kernel_backend_matrix:e #1
157 {
158 <*luatex>
159 \tex_pdfextension:D setmatrix
160 </luatex>
161 <*pdftex>
162 \tex_pdfsetmatrix:D
163 </pdftex>
164 {#1}
165 }

```

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

```

166 </luatex | pdftex>

```

1.3 dvipdfmx backend

167 $\langle *dvipdfmx | xetex \rangle$

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.

`_kernel_backend_literal_pdf:n`
`_kernel_backend_literal_pdf:e`

```
168 \cs_new_protected:Npn \_kernel_backend_literal_pdf:n #1
169 { \_kernel_backend_literal:n { pdf:literal~ #1 } }
170 \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!

```
171 \cs_new_protected:Npn \_kernel_backend_literal_page:n #1
172 { \_kernel_backend_literal:n { pdf:literal~direct~ #1 } }
```

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

`_kernel_backend_scope_begin:`
`_kernel_backend_scope_end:`

Scoping is done using the backend-specific specials. We use the versions originally from `xdvipdfmx (x:)` as these are well-tested “in the wild”.

```
173 \cs_new_protected:Npn \_kernel_backend_scope_begin:
174 { \_kernel_backend_literal:n { x:gsave } }
175 \cs_new_protected:Npn \_kernel_backend_scope_end:
176 { \_kernel_backend_literal:n { x:grestore } }
```

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

177 $\langle /dvipdfmx | xetex \rangle$

1.4 dvisvgm backend

178 $\langle *dvisvgm \rangle$

`_kernel_backend_literal_svg:n`
`_kernel_backend_literal_svg:e`

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.

```
179 \cs_new_protected:Npn \_kernel_backend_literal_svg:n #1
180 { \_kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }
181 \cs_generate_variant:Nn \_kernel_backend_literal_svg:n { e }
```

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

`\g__kernel_backend_scope_int`
`\l__kernel_backend_scope_int`

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

```
182 \int_new:N \g__kernel_backend_scope_int
183 \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.

```

184 \cs_new_protected:Npn \__kernel_backend_scope_begin:
185 {
186   \__kernel_backend_literal_svg:n { <g> }
187   \int_set_eq:NN
188     \l__kernel_backend_scope_int
189     \g__kernel_backend_scope_int
190   \group_begin:
191     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
192 }
193 \cs_new_protected:Npn \__kernel_backend_scope_end:
194 {
195   \prg_replicate:nn
196     { \g__kernel_backend_scope_int }
197     { \__kernel_backend_literal_svg:n { </g> } }
198   \group_end:
199   \int_gset_eq:NN
200     \g__kernel_backend_scope_int
201     \l__kernel_backend_scope_int
202 }
203 \cs_new_protected:Npn \__kernel_backend_scope_begin:n #1
204 {
205   \__kernel_backend_literal_svg:n { <g ~ #1 > }
206   \int_set_eq:NN
207     \l__kernel_backend_scope_int
208     \g__kernel_backend_scope_int
209   \group_begin:
210     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
211 }
212 \cs_generate_variant:Nn \__kernel_backend_scope_begin:n { e }
213 \cs_new_protected:Npn \__kernel_backend_scope:n #1
214 {
215   \__kernel_backend_literal_svg:n { <g ~ #1 > }
216   \int_gincr:N \g__kernel_backend_scope_int
217 }
218 \cs_generate_variant:Nn \__kernel_backend_scope:n { e }

```

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

```

219 </dvisvgm>
220 </package>

```

2 l3backend-box implementation

```

221 <*package>
222 <@@=box>

```

2.1 dvips backend

```

223 <*dvips>

```

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

```

224 \cs_new_protected:Npn \_box_backend_clip:N #1
225 {
226   \_kernel_backend_scope_begin:
227   \_kernel_backend_align_begin:
228   \_kernel_backend_literal_postscript:n { matrix~currentmatrix }
229   \_kernel_backend_literal_postscript:n
230   { Resolution~72~div~VResolution~72~div~scale }
231   \_kernel_backend_literal_postscript:n { DVImag~dup~scale }
232   \_kernel_backend_literal_postscript:e
233   {
234     0 ~
235     \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
236     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
237     \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
238     rectclip
239   }
240   \_kernel_backend_literal_postscript:n { setmatrix }
241   \_kernel_backend_align_end:
242   \hbox_overlap_right:n { \box_use:N #1 }
243   \_kernel_backend_scope_end:
244   \skip_horizontal:n { \box_wd:N #1 }
245 }

```

(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`

```

246 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
247 { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
248 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
249 {
250   \_kernel_backend_scope_begin:
251   \_kernel_backend_align_begin:
252   \_kernel_backend_literal_postscript:e
253   {
254     \fp_compare:nNnTF {#2} = \c_zero_fp
255     { 0 }
256     { \fp_eval:n { round ( -(#2) , 5 ) } } ~
257     rotate
258   }
259   \_kernel_backend_align_end:
260   \box_use:N #1
261   \_kernel_backend_scope_end:
262 }

```

(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.

```

263 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
264 {
265   \__kernel_backend_scope_begin:
266   \__kernel_backend_align_begin:
267   \__kernel_backend_literal_postscript:e
268   {
269     \fp_eval:n { round ( #2 , 5 ) } ~
270     \fp_eval:n { round ( #3 , 5 ) } ~
271     scale
272   }
273   \__kernel_backend_align_end:
274   \hbox_overlap_right:n { \box_use:N #1 }
275   \__kernel_backend_scope_end:
276 }

(End of definition for \__box_backend_scale:Nnn.)

277 </dvips>

```

2.2 LuaTeX and pdfTeX backends

```

278 <*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.

```

279 \cs_new_protected:Npn \__box_backend_clip:N #1
280 {
281   \__kernel_backend_scope_begin:
282   \__kernel_backend_literal_pdf:e
283   {
284     0~
285     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
286     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
287     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
288     re~W~n
289   }
290   \hbox_overlap_right:n { \box_use:N #1 }
291   \__kernel_backend_scope_end:
292   \skip_horizontal:n { \box_wd:N #1 }
293 }

(End of definition for \__box_backend_clip:N.)

```

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

```

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

```

```

295 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
296 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
297 {
298   \__kernel_backend_scope_begin:
299   \box_set_wd:Nn #1 { Opt }
300   \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
301   \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
302     { \fp_zero:N \l__box_backend_cos_fp }
303   \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
304   \__kernel_backend_matrix:e
305   {
306     \fp_use:N \l__box_backend_cos_fp \c_space_tl
307     \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp
308       { 0~0 }
309       {
310         \fp_use:N \l__box_backend_sin_fp
311         \c_space_tl
312         \fp_eval:n { -\l__box_backend_sin_fp }
313       }
314     \c_space_tl
315     \fp_use:N \l__box_backend_cos_fp
316   }
317   \box_use:N #1
318   \__kernel_backend_scope_end:
319 }
320 \fp_new:N \l__box_backend_cos_fp
321 \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.

```

322 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
323 {
324   \__kernel_backend_scope_begin:
325   \__kernel_backend_matrix:e
326   {
327     \fp_eval:n { round ( #2 , 5 ) } ~
328     0~0~
329     \fp_eval:n { round ( #3 , 5 ) }
330   }
331   \hbox_overlap_right:n { \box_use:N #1 }
332   \__kernel_backend_scope_end:
333 }

```

(End of definition for __box_backend_scale:Nnn.)

334 </luatex | pdftex>

2.3 dvipdfmx/X_YTeX backend

335 <*dvipdfmx | xetex>

__box_backend_clip:N The code here is identical to that for Lua_TEX/pdf_TEX: unlike rotation and scaling, there is no higher-level support in the backend for clipping.

```

336 \cs_new_protected:Npn \__box_backend_clip:N #1

```

```

337 {
338   \__kernel_backend_scope_begin:
339   \__kernel_backend_literal_pdf:e
340   {
341     0~
342     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
343     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
344     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
345     re~W~n
346   }
347   \hbox_overlap_right:n { \box_use:N #1 }
348   \__kernel_backend_scope_end:
349   \skip_horizontal:n { \box_wd:N #1 }
350 }

```

(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.

```

351 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
352 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
353 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
354 {
355   \__kernel_backend_scope_begin:
356   \__kernel_backend_literal:e
357   {
358     x:rotate~
359     \fp_compare:nNnTF {#2} = \c_zero_fp
360     { 0 }
361     { \fp_eval:n { round ( #2 , 5 ) } }
362   }
363   \box_use:N #1
364   \__kernel_backend_scope_end:
365 }

```

(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.

```

366 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
367 {
368   \__kernel_backend_scope_begin:
369   \__kernel_backend_literal:e
370   {
371     x:scale~
372     \fp_eval:n { round ( #2 , 5 ) } ~
373     \fp_eval:n { round ( #3 , 5 ) }
374   }
375   \hbox_overlap_right:n { \box_use:N #1 }
376   \__kernel_backend_scope_end:
377 }

```

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

378 `</dviptfm | xetex>`

2.4 dvisvgm backend

379 `<*dvisvgm>`

`__box_backend_clip:N`
`\g__kernel_clip_path_int`

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

```
380 \cs_new_protected:Npn \__box_backend_clip:N #1
381 {
382   \int_gincr:N \g__kernel_clip_path_int
383   \__kernel_backend_literal_svg:e
384   { < clipPath-id = " l3cp \int_use:N \g__kernel_clip_path_int " > }
385   \__kernel_backend_literal_svg:e
386   {
387     <
388       path ~ d =
389         "
390           M ~ 0 ~
391             \dim_to_decimal:n { -\box_dp:N #1 } ~
392             L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
393             \dim_to_decimal:n { -\box_dp:N #1 } ~
394             L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
395             \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
396             L ~ 0 ~
397             \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
398             Z
399           "
400         />
401     }
402   \__kernel_backend_literal_svg:n
403   { < /clipPath > }
```

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

```
404 \__kernel_backend_scope_begin:n
405 {
406   transform =
407     "
408       translate ( { ?x } , { ?y } ) ~
409       scale ( 1 , -1 )
410     "
411 }
412 \__kernel_backend_scope:e
```

```

413     {
414         clip-path =
415             "url ( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int ) "
416     }
417     \__kernel_backend_scope:n
418     {
419         transform =
420             "
421             scale ( -1 , 1 ) ~
422             translate ( { ?x } , { ?y } ) ~
423             scale ( -1 , -1 )
424             "
425     }
426     \box_use:N #1
427     \__kernel_backend_scope_end:
428 }
429 \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.

```

430 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
431 {
432     \__kernel_backend_scope_begin:e
433     {
434         transform =
435             "
436             rotate
437             ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
438             "
439     }
440     \box_use:N #1
441     \__kernel_backend_scope_end:
442 }

```

(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).

```

443 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
444 {
445     \__kernel_backend_scope_begin:e
446     {
447         transform =
448             "
449             translate ( { ?x } , { ?y } ) ~
450             scale
451             (
452                 \fp_eval:n { round ( -#2 , 5 ) } ,
453                 \fp_eval:n { round ( -#3 , 5 ) }
454             ) ~

```

```

455         translate ( { ?x } , { ?y } ) ~
456         scale ( -1 )
457     "
458 }
459 \hbox_overlap_right:n { \box_use:N #1 }
460 \__kernel_backend_scope_end:
461 }

```

(End of definition for __box_backend_scale:Nnn.)

```

462 </dvisvgm>
463 </package>

```

3 I3backend-color implementation

```

464 <*package>
465 <@@=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

```

466 <*luatex | pdftex>

```

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

```

467 \int_new:N \l__color_backend_stack_int

```

(End of definition for \l__color_backend_stack_int.)

```

468 </luatex | pdftex>

```

3.1.2 Lua_YL_AT_EX and pdf_YL_AT_EX

```

469 <*luatex | pdftex>

```

__kernel_color_backend_stack_init:Nnn

```

470 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
471 {
472     \int_const:Nn #1
473     {
474         <*luatex>
475         \tex_pdffeedback:D colorstackinit ~
476         </luatex>

```

```

477 <*pdfTeX>
478     \tex_pdfcolorstackinit:D
479 </pdfTeX>
480     \tl_if_blank:nF {#2} { #2 ~ }
481     {#3}
482   }
483 }

```

(End of definition for __kernel_color_backend_stack_init:Nnn.)

```

\__kernel_color_backend_stack_push:nn
\__kernel_color_backend_stack_pop:n
484 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
485 {
486   <*luatex>
487     \tex_pdfextension:D colorstack ~
488   </luatex>
489   <*pdfTeX>
490     \tex_pdfcolorstack:D
491   </pdfTeX>
492     \int_eval:n {#1} ~ push ~ {#2}
493   }
494 \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
495 {
496   <*luatex>
497     \tex_pdfextension:D colorstack ~
498   </luatex>
499   <*pdfTeX>
500     \tex_pdfcolorstack:D
501   </pdfTeX>
502     \int_eval:n {#1} ~ pop \scan_stop:
503   }

```

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

```

504 </luatex | pdfTeX>

```

3.2 General color

3.2.1 dvips-style

```

505 <*dvips | dvisvgm>

```

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

```

\__color_backend_select_cmyk:nN
\__color_backend_select_gray:nN
\__color_backend_select_named:nN
\__color_backend_select_rgb:nN
\__color_backend_select:nN
\__color_backend_reset:
506 \cs_new_protected:Npn \__color_backend_select_cmyk:nN #1
507   { \__color_backend_select:nN { cmyk ~ #1 } }
508 \cs_new_protected:Npn \__color_backend_select_gray:nN #1
509   { \__color_backend_select:nN { gray ~ #1 } }
510 \cs_new_protected:Npn \__color_backend_select_named:nN #1
511   { \__color_backend_select:nN { ~ #1 } }
512 \cs_new_protected:Npn \__color_backend_select_rgb:nN #1
513   { \__color_backend_select:nN { rgb ~ #1 } }
514 \cs_new_protected:Npn \__color_backend_select:nN #1#2
515   {
516     \tl_set:Nn #2 {#1}

```

```

517 \__kernel_backend_literal:n { color~push~ #1 }
518 <*dvips>
519 \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
520 </dvips>
521 }
522 \cs_new_protected:Npn \__color_backend_reset:
523 { \__kernel_backend_literal:n { color~pop } }

```

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

```

524 </dvips | dvisvgm>

```

3.2.2 LuaTeX and pdfTeX

```

525 <*luatex | pdftex>

```

```

\l__color_backend_fill_tl
\l__color_backend_stroke_tl

```

```

526 \tl_new:N \l__color_backend_fill_tl
527 \tl_new:N \l__color_backend_stroke_tl
528 \tl_set:Nn \l__color_backend_fill_tl { 0 ~ g }
529 \tl_set:Nn \l__color_backend_stroke_tl { 0 ~ G }

```

(End of definition for \l__color_backend_fill_tl and \l__color_backend_stroke_tl.)

```

\__color_backend_select_cmyk:nN
\__color_backend_select_gray:nN
\__color_backend_select_rgb:nN
\__color_backend_select:nnN
\__color_backend_reset:

```

Store the values then pass to the stack.

```

530 \cs_new_protected:Npn \__color_backend_select_cmyk:nN #1
531 { \__color_backend_select:nnN { #1 ~ k } { #1 ~ K } }
532 \cs_new_protected:Npn \__color_backend_select_gray:nN #1
533 { \__color_backend_select:nnN { #1 ~ g } { #1 ~ G } }
534 \cs_new_protected:Npn \__color_backend_select_rgb:nN #1
535 { \__color_backend_select:nnN { #1 ~ rg } { #1 ~ RG } }
536 \cs_new_protected:Npn \__color_backend_select:nnN #1#2#3
537 {
538   \tl_set:Nn \l__color_backend_fill_tl {#1}
539   \tl_set:Nn \l__color_backend_stroke_tl {#2}
540   \tl_set:Nn #3 { #1 ~ #2 }
541   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int { #1 ~ #2 }
542 }
543 \cs_new_protected:Npn \__color_backend_reset:
544 { \__kernel_color_backend_stack_pop:n \l__color_backend_stack_int }

```

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

```

545 </luatex | pdftex>

```

3.2.3 dvipdfmx/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.

```

546 <*dvipdfmx | xetex>

```



```

\__color_backend_select:n
  \__color_backend_select_cmyk:nN
  \__color_backend_select_gray:nN
  \__color_backend_select_rgb:nN
\__color_backend_reset:
547 \cs_new_protected:Npn \__color_backend_select:n #1
548 { \__kernel_backend_literal:n { pdf : bc ~ [ #1 ] } }
549 \cs_new_protected:Npn \__color_backend_select_cmyk:nN #1#2
550 {
551   \tl_set:Nn #2 { cmyk ~ #1 }
552   \__color_backend_select:n {#1}
553 }
554 \cs_new_protected:Npn \__color_backend_select_gray:nN #1#2
555 {
556   \tl_set:Nn #2 { gray ~ #1 }
557   \__color_backend_select:n {#1}
558 }
559 \cs_new_protected:Npn \__color_backend_select_rgb:nN #1#2
560 {
561   \tl_set:Nn #2 { rgb ~ #1 }
562   \__color_backend_select:n {#1}
563 }
564 \cs_new_protected:Npn \__color_backend_reset:
565 { \__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.)

```

\__color_backend_select_named:nN
566 \cs_new_protected:Npn \__color_backend_select_named:nN #1#2
567 {
568   \str_if_eq:nnTF {#1} { Black }
569   { \__color_backend_select_gray:n { 0 } }
570   { \msg_error:nnn { color } { unknown-named-color } {#1} }
571 }
572 \msg_new:nnn { color } { unknown-named-color }
573 { Named-color~'#1'~is~not~known. }

```

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

(End of definition for __color_backend_select_named:nN.)

574 </dviptfm | xetex>

3.3 Separations

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

575 <*dviptfm | luatex | pdftex | xetex | dvips>

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

```

\g__color_backend_colorant_prop
576 \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
577 \cs_new:Npe \__color_backend_devicen_colorants:n #1
578 {
579   \exp_not:N \tl_if_blank:nF {#1}
580   {

```

```

581     \c_space_tl
582     << ~
583     /Colorants ~
584     << ~
585         \exp_not:N \__color_backend_devicen_colorants:w #1 ~
586         \exp_not:N \q_recursion_tail \c_space_tl
587         \exp_not:N \q_recursion_stop
588     >> ~
589 >>
590 }
591 }
592 \cs_new:Npn \__color_backend_devicen_colorants:w #1 ~
593 {
594     \quark_if_recursion_tail_stop:n {#1}
595     \prop_if_in:NnT \g__color_backend_colorant_prop {#1}
596     {
597         #1 ~
598         \prop_item:Nn \g__color_backend_colorant_prop {#1} ~
599     }
600     \__color_backend_devicen_colorants:w
601 }

```

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

```

602 </dvipdfmx | luatex | pdfTeX | xetex | dvips>
603 <*dvips>

```

```

\__color_backend_select_separation:nnN
\__color_backend_select_devicen:nnN
604 \cs_new_protected:Npn \__color_backend_select_separation:nnN #1#2#3
605 {
606     \tl_set:Nn #2 { ~ #1 ~ #2 }
607     \__color_backend_select:nN { separation ~ #1 ~ #2 } \l__color_tmp_tl
608 }
609 \cs_new_eq:NN \__color_backend_select_devicen:nnN \__color_backend_select_separation:nnN

```

(End of definition for __color_backend_select_separation:nnN and __color_backend_select_devicen:nnN.)

```

\__color_backend_select_iccbased:nnN
No support.
610 \cs_new_protected:Npn \__color_backend_select_iccbased:nnN #1#2#3 { }

```

(End of definition for __color_backend_select_iccbased: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.

```

611 \cs_new_protected:Npe \__color_backend_separation_init:nnnnn #1#2#3#4#5
612 {
613     \bool_if:NT \g__kernel_backend_header_bool
614     {
615         \exp_not:N \exp_args:Ne \__kernel_backend_first_shipout:n
616         {
617             \exp_not:N \__color_backend_separation_init_aux:nnnnnn
618             { \exp_not:N \int_use:N \g__color_model_int }

```

```

619         {#1} {#2} {#3} {#4} {#5}
620     }
621     \prop_gput:Nee \exp_not:N \g__color_backend_colorant_prop
622     { / \exp_not:N \str_convert_pdfname:n {#1} }
623     {
624         << ~
625         /setcolorspace ~ {} ~
626         >> ~ begin ~
627         color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
628         end
629     }
630 }
631 }
632 \cs_generate_variant:Nn \__color_backend_separation_init:nnnnn { nee }
633 \cs_new_protected:Npn \__color_backend_separation_init_aux:nnnnnn #1#2#3#4#5#6
634 {
635     \__kernel_backend_literal:e
636     {
637         !
638         TeXDict ~ begin ~
639         /color #1
640         {
641             [ ~
642             /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
643             [ ~ #3 ~ ] ~
644             {
645                 \cs_if_exist_use:cF { __color_backend_separation_init_ #3 :nnn }
646                 { \__color_backend_separation_init:nnn }
647                 {#4} {#5} {#6}
648             }
649             ] ~ setcolorspace
650         } ~ def ~
651         end
652     }
653 }
654 \cs_new:cpn { __color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
655 { \__color_backend_separation_init_Device:Nn 4 {#3} }
656 \cs_new:cpn { __color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
657 { \__color_backend_separation_init_Device:Nn 1 {#3} }
658 \cs_new:cpn { __color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
659 { \__color_backend_separation_init_Device:Nn 2 {#3} }
660 \cs_new:Npn \__color_backend_separation_init_Device:Nn #1#2
661 {
662     #2 ~
663     \prg_replicate:nn {#1}
664     { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
665     \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
666 }

```

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.

```

667 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3

```

```

668 {
669   \exp_args:Ne \__color_backend_separation_init:nnnn
670   { \__color_backend_separation_init_count:n {#2} }
671   {#1} {#2} {#3}
672 }
673 \cs_new:Npn \__color_backend_separation_init_count:n #1
674 { \int_eval:n { 0 \__color_backend_separation_init_count:w #1 ~ \s__color_stop } }
675 \cs_new:Npn \__color_backend_separation_init_count:w #1 ~ #2 \s__color_stop
676 {
677   +1
678   \tl_if_blank:nF {#2}
679   { \__color_backend_separation_init_count:w #2 \s__color_stop }
680 }

```

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

```

681 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
682 {
683   \__color_backend_separation_init:w #3 ~ \s__color_stop #4 ~ \s__color_stop
684   \prg_replicate:nn {#1}
685   {
686     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
687     \int_eval:n { 3 * #1 } ~ index ~ mul ~
688     2 ~ index ~ add ~
689     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
690   }
691   \int_step_function:nnnN {#1} { -1 } { 1 }
692   \__color_backend_separation_init:n
693   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
694   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
695   \tl_if_blank:nF {#2}
696   { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
697 }
698 \cs_new:Npn \__color_backend_separation_init:w
699 #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
700 {
701   #1 ~ #3 ~ 0 ~
702   \tl_if_blank:nF {#2}
703   { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }
704 }
705 \cs_new:Npn \__color_backend_separation_init:n #1
706 { \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.

```

707 \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s_color_stop
708 {
709   #2 ~ #3 ~
710   2 ~ index ~ 2 ~ index ~ lt ~
711   { ~ pop ~ exch ~ pop ~ } ~
712   { ~
713     2 ~ index ~ 1 ~ index ~ gt ~
714     { ~ exch ~ pop ~ exch ~ pop ~ } ~
715     { ~ pop ~ pop ~ } ~
716     ifelse ~
717   }
718   ifelse ~
719   #1 ~ 1 ~ roll ~
720   \tl_if_blank:nF {#4}
721   { \__color_backend_separation_init:nw {#1} #4 \s_color_stop }
722 }

```

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.

```

723 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
724 {
725   \__color_backend_separation_init:neenn
726   {#2}
727   {
728     /CIEBasedABC ~
729     << ~
730     /RangeABC ~ [ ~ \c__color_model_range_CIELAB_tl \c_space_tl ] ~
731     /DecodeABC ~
732     [ ~
733       { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
734       { ~ 500 ~ div ~ } ~ bind ~
735       { ~ 200 ~ div ~ } ~ bind ~
736     ] ~
737     /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
738     /DecodeLMN ~
739     [ ~
740     { ~
741       dup ~ 6 ~ 29 ~ div ~ ge ~
742       { ~ dup ~ dup ~ mul ~ mul ~ } ~
743       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
744       ifelse ~
745       0.9505 ~ mul ~
746     } ~ bind ~
747     { ~
748       dup ~ 6 ~ 29 ~ div ~ ge ~
749       { ~ dup ~ dup ~ mul ~ mul ~ } ~
750       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
751       ifelse ~
752     } ~ bind ~
753     { ~
754       dup ~ 6 ~ 29 ~ div ~ ge ~
755       { ~ dup ~ dup ~ mul ~ mul ~ } ~
756       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~

```

```

757         ifelse ~
758         1.0890 ~ mul ~
759         } ~ bind
760     ] ~
761     /WhitePoint ~
762     [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
763     >>
764 }
765 { \c__color_model_range_CIELAB_tl }
766 { 100 ~ 0 ~ 0 }
767 {#3}
768 }

```

(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.

```

769 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
770 {
771     \__kernel_backend_literal:e
772     {
773         !
774         TeXDict ~ begin ~
775         /color \int_use:N \g__color_model_int
776         {
777             [ ~
778             /DeviceN ~
779             [ ~ #1 ~ ] ~
780             #2 ~
781             { ~ #3 ~ } ~
782             \__color_backend_devicen_colorants:n {#1}
783             ] ~ setcolorspace
784             } ~ def ~
785         end
786     }
787 }

```

(End of definition for __color_backend_devicen_init:nnn.)

__color_backend_iccbased_init:nnn No support at present.

```

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

```

(End of definition for __color_backend_iccbased_init:nnn.)

```

789 </dvips>
790 <*dvisvgm>

```

__color_backend_select_separation:nnN No support at present.

```

791 \cs_new_protected:Npn \__color_backend_select_separation:nnN #1#2#3 { }
792 \cs_new_eq:NN \__color_backend_select_devicen:nnN \__color_backend_select_separation:nnN

```

(End of definition for __color_backend_select_separation:nnN and __color_backend_select_devicen:nnN.)

__color_backend_separation_init:nnnnn No support at present.

```

\__color_backend_separation_init_CIELAB:nnn
793 \cs_new_protected:Npn \__color_backend_separation_init:nnnnn #1#2#3#4#5 { }
794 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnnnnn #1#2#3 { }

```

(End of definition for `_color_backend_separation_init:nnnn` and `_color_backend_separation_init_CIELAB:nnn`.)

`_color_backend_select_iccbased:nn` As detailed in <https://www.w3.org/TR/css-color-4/#at-profile>, we can apply a color profile using CSS. As we have a local file, we use a relative URL.

```

795 \cs_new_protected:Npn \_color_backend_select_iccbased:nn #1#2
796 {
797   \_kernel_backend_literal_svg:e
798   {
799     <style>
800       @color-profile ~
801       \str_if_eq:nnTF {#2} { cmyk }
802       { device-cmyk }
803       { --color \int_use:N \g_color_model_int }
804       \c_space_tl
805       {
806         src:("#1")
807       }
808     </style>
809   }
810 }

```

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

```

811 </divisvgm>
812 <*dvipdfmx | luatex | pdftex | xetex>

```

```

\_color_backend_select_separation:nnN
\_color_backend_select_devicen:nnN
\_color_backend_select_iccbased:nnN
813 <*dvipdfmx | xetex>
814 \cs_new_protected:Npn \_color_backend_select_separation:nnN #1#2#3
815 { \_kernel_backend_literal:e { pdf : bc ~ \pdf_object_ref:n {#1} ~ [ #2 ] } }
816 </dvipdfmx | xetex>
817 <*luatex | pdftex>
818 \cs_new_protected:Npn \_color_backend_select_separation:nnN #1#2
819 { \_color_backend_select:nnN { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
820 </luatex | pdftex>
821 \cs_new_eq:NN \_color_backend_select_devicen:nnN \_color_backend_select_separation:nnN
822 \cs_new_eq:NN \_color_backend_select_iccbased:nnN \_color_backend_select_separation:nnN

```

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

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

```

823 \cs_new_protected:Npn \_color_backend_init_resource:n #1
824 {
825   <*luatex | pdftex>
826   \bool_lazy_and:nnT
827   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
828   { \pdfmanagement_if_active_p: }
829   {
830     \use:e
831     {
832       \pdfmanagement_add:nnn
833       { Page / Resources / ColorSpace }

```

```

834         { #1 }
835         { \pdf_object_ref_last: }
836     }
837 }
838 </luatex | pdftex>
839 }

```

(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.

```

840 \cs_new_protected:Npn __color_backend_separation_init:nnnnn #1#2#3#4#5
841 {
842     \pdf_object_unnamed_write:ne { dict }
843     {
844         /FunctionType ~ 2
845         /Domain ~ [0 ~ 1]
846         \tl_if_blank:nF {#3} { /Range ~ [#3] }
847         /C0 ~ [#4] ~
848         /C1 ~ [#5] /N ~ 1
849     }
850     \exp_args:Ne __color_backend_separation_init:nn
851     { \str_convert_pdfname:n {#1} } {#2}
852     __color_backend_init_resource:n { color \int_use:N \g__color_model_int }
853 }
854 \cs_new_protected:Npn __color_backend_separation_init:nn #1#2
855 {
856     \use:e
857     {
858         \pdf_object_new:n { color \int_use:N \g__color_model_int }
859         \pdf_object_write:nnn { color \int_use:N \g__color_model_int } { array }
860         { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
861     }
862     \prop_gput:Nne \g__color_backend_colorant_prop { /#1 }
863     { \pdf_object_ref_last: }
864 }

```

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

```

865 \cs_new_protected:Npn __color_backend_separation_init_CIELAB:nnn #1#2#3
866 {
867     \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
868     {
869         \pdf_object_new:n { __color_illuminant_CIELAB_ #1 }
870         \pdf_object_write:nne { __color_illuminant_CIELAB_ #1 } { array }
871         {
872             /Lab ~
873             <<
874             /WhitePoint ~
875             [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
876             /Range ~ [ \c__color_model_range_CIELAB_tl ]
877             >>

```



```

878     }
879   }
880   \_color_backend_separation_init:nnnnn
881   {#2}
882   { \pdf_object_ref:n { \_color_illuminant_CIELAB_ #1 } }
883   { \c\_color_model_range_CIELAB_t1 }
884   { 100 ~ 0 ~ 0 }
885   {#3}
886 }

```

(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.

```

887 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
888 {
889   \pdf_object_unnamed_write:ne { stream }
890   {
891     {
892       /FunctionType ~ 4 ~
893       /Domain ~
894       [ ~
895         \prg_replicate:nn
896         { 0 \_color_backend_devicen_init:w #1 ~ \s_color_stop }
897         { 0 ~ 1 ~ }
898       ] ~
899       /Range ~
900       [ ~
901         \str_case:nn {#2}
902         {
903           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
904           { /DeviceGray } { 0 ~ 1 }
905           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
906         } ~
907       ]
908     }
909     { {#3} }
910   }
911   \use:e
912   {
913     \pdf_object_new:n { color \int_use:N \g_color_model_int }
914     \pdf_object_write:nnn { color \int_use:N \g_color_model_int } { array }
915     {
916       /DeviceN ~
917       [ ~ #1 ~ ] ~
918       #2 ~
919       \pdf_object_ref_last:
920       \_color_backend_devicen_colorants:n {#1}
921     }
922   }
923   \_color_backend_init_resource:n { color \int_use:N \g_color_model_int }
924 }
925 \cs_new:Npn \_color_backend_devicen_init:w #1 ~ #2 \s_color_stop

```

```

926 {
927   + 1
928   \tl_if_blank:nF {#2}
929   { \__color_backend_devicen_init:w #2 \s__color_stop }
930 }

```

(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.

```

931 \cs_new_protected:Npn \__color_backend_iccbased_init:nnn #1#2#3
932 {
933   \pdf_object_if_exist:nF { __color_icc_ #1 }
934   {
935     \pdf_object_new:n { __color_icc_ #1 }
936     \pdf_object_write:nne { __color_icc_ #1 } { fstream }
937     {
938       {
939         /N ~ \exp_not:n { #2 } ~
940         \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
941       }
942       {#1}
943     }
944   }
945   \pdf_object_unnamed_write:ne { array }
946   { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
947   \__color_backend_init_resource:n { color \int_use:N \g__color_model_int }
948 }

```

(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.

```

949 \cs_new_protected:Npn \__color_backend_iccbased_device:nnn #1#2#3
950 {
951   \pdf_object_if_exist:nF { __color_icc_ #1 }
952   {
953     \pdf_object_new:n { __color_icc_ #1 }
954     \pdf_object_write:nnn { __color_icc_ #1 } { fstream }
955     {
956       { /N ~ #3 }
957       {#1}
958     }
959   }
960   \pdf_object_unnamed_write:ne { array }
961   { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
962   \__color_backend_init_resource:n { Default #2 }
963 }

```

(End of definition for __color_backend_iccbased_device:nnn.)

```

964 </dviPDFmx | luatex | pdftex | xetex>

```

3.4 Fill and stroke color

Here, `dvipdfmx/XYTeX` 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.

965 `<*dvipdfmx | xetex>`

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

```
966 \cs_new_protected:Npn \__color_backend_fill:nN #1#2
967 { \__kernel_backend_literal:n { pdf : bc ~ fill ~ [ #1 ] } }
968 \cs_new_eq:NN \__color_backend_fill_cmyk:nN \__color_backend_fill:nN
969 \cs_new_eq:NN \__color_backend_fill_gray:nN \__color_backend_fill:nN
970 \cs_new_eq:NN \__color_backend_fill_rgb:nN \__color_backend_fill:nN
971 \cs_new_protected:Npn \__color_backend_stroke:nN #1#2
972 { \__kernel_backend_literal:n { pdf : bc ~ stroke ~ [ #1 ] } }
973 \cs_new_eq:NN \__color_backend_stroke_cmyk:nN \__color_backend_stroke:nN
974 \cs_new_eq:NN \__color_backend_stroke_gray:nN \__color_backend_stroke:nN
975 \cs_new_eq:NN \__color_backend_stroke_rgb:nN \__color_backend_stroke:nN
```

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

```
\__color_backend_fill_separation:nnN
\__color_backend_stroke_separation:nnN
\__color_backend_fill_devicen:nnN
\__color_backend_stroke_devicen:nnN
976 \cs_new_protected:Npn \__color_backend_fill_separation:nnN #1#2#3
977 {
978   \__kernel_backend_literal:e
979   { pdf : bc ~ fill ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
980 }
981 \cs_new_protected:Npn \__color_backend_stroke_separation:nnN #1#2#3
982 {
983   \__kernel_backend_literal:e
984   { pdf : bc ~ stroke ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
985 }
986 \cs_new_eq:NN \__color_backend_fill_devicen:nnN \__color_backend_fill_separation:nnN
987 \cs_new_eq:NN \__color_backend_stroke_devicen:nnN \__color_backend_stroke_separation:nnN
```

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

```
\__color_backend_fill_reset:
\__color_backend_stroke_reset:
988 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
989 \cs_new_eq:NN \__color_backend_stroke_reset: \__color_backend_reset:
```

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

990 `</dvipdfmx | xetex>`

991 `<*luatex | pdftex>`

Drawing (fill/stroke) color is handled in `dvipdfmx/XYTeX` in the same way as LuaTeX/pdfTeX. We use the same approach as earlier, except the color stack is not involved so the generic direct PDF operation is used. There is no worry about the nature of strokes: everything is handled automatically. Here, we can set the classical data at the same time.

```
992 \cs_new_protected:Npn \__color_backend_fill_cmyk:nN #1
```

```
\__color_backend_fill_gray:nN
```

```
\__color_backend_fill_rgb:nN
```

```
\__color_backend_stroke_cmyk:nN
```

```
\__color_backend_stroke_gray:nN
```

```
\__color_backend_stroke_rgb:nN
```

```
\__color_backend_stroke:nN
```

```

993 { \_color_backend_fill:nN { #1 ~ k } }
994 \cs_new_protected:Npn \_color_backend_fill_gray:nN #1
995 { \_color_backend_fill:nN { #1 ~ g } }
996 \cs_new_protected:Npn \_color_backend_fill_rgb:nN #1
997 { \_color_backend_fill:nN { #1 ~ rg } }
998 \cs_new_protected:Npn \_color_backend_fill:nN #1#2
999 {
1000   \tl_set:Nn \l__color_backend_fill_tl {#1}
1001   \tl_set:Ne #2 { #1 ~ \l__color_backend_stroke_tl }
1002   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int {#2}
1003 }
1004 \cs_new_protected:Npn \_color_backend_stroke_cmyk:nN #1
1005 { \_color_backend_stroke:nN { #1 ~ K } }
1006 \cs_new_protected:Npn \_color_backend_stroke_gray:nN #1
1007 { \_color_backend_stroke:nN { #1 ~ G } }
1008 \cs_new_protected:Npn \_color_backend_stroke_rgb:nN #1
1009 { \_color_backend_stroke:nN { #1 ~ RG } }
1010 \cs_new_protected:Npn \_color_backend_stroke:nN #1#2
1011 {
1012   \tl_set:Nn \l__color_backend_stroke_tl {#1}
1013   \tl_set:Ne #2 { \l__color_backend_fill_tl \c_space_tl #1 }
1014   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int {#2}
1015 }

```

(End of definition for _color_backend_fill_cmyk:nN and others.)

```

\_color_backend_fill_separation:nnN
\_color_backend_stroke_separation:nnN
\_color_backend_fill_devicen:nnN
\_color_backend_stroke_devicen:nnN
1016 \cs_new_protected:Npn \_color_backend_fill_separation:nnN #1#2#3
1017 { \_color_backend_fill:nN { /#1 ~ cs ~ #2 ~ scn } }
1018 \cs_new_protected:Npn \_color_backend_stroke_separation:nnN #1#2#3
1019 { \_color_backend_stroke:nN { /#1 ~ CS ~ #2 ~ SCN } }
1020 \cs_new_eq:NN \_color_backend_fill_devicen:nnN \_color_backend_fill_separation:nnN
1021 \cs_new_eq:NN \_color_backend_stroke_devicen:nnN \_color_backend_stroke_separation:nnN

```

(End of definition for _color_backend_fill_separation:nnN and others.)

```

\_color_backend_fill_reset:
\_color_backend_stroke_reset:
1022 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
1023 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:

```

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

```

1024 </luatex | pdftex>
1025 <*dvips>

```

```

\_color_backend_fill_cmyk:nN
\_color_backend_fill_gray:nN
\_color_backend_fill_rgb:nN
\_color_backend_fill:nN
\_color_backend_stroke_cmyk:nN
\_color_backend_stroke_gray:nN
\_color_backend_stroke_rgb:nN
1026 \cs_new_protected:Npn \_color_backend_fill_cmyk:nN #1
1027 { \_color_backend_fill:nN { cmyk ~ #1 } }
1028 \cs_new_protected:Npn \_color_backend_fill_gray:nN #1
1029 { \_color_backend_fill:nN { gray ~ #1 } }
1030 \cs_new_protected:Npn \_color_backend_fill_rgb:nN #1
1031 { \_color_backend_fill:nN { rgb ~ #1 } }
1032 \cs_new_protected:Npn \_color_backend_fill:nN #1#2
1033 {

```

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.

```

1034 \tl_set:Nn #2 {#1}
1035 \__kernel_backend_literal:n { color-push~ #1 }
1036 }
1037 \cs_new_protected:Npn \__color_backend_stroke_cmyk:nN #1#2
1038 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1039 \cs_new_protected:Npn \__color_backend_stroke_gray:nN #1#2
1040 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1041 \cs_new_protected:Npn \__color_backend_stroke_rgb:nN #1#2
1042 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }

```

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

```

\__color_backend_fill_separation:nnN
\__color_backend_stroke_separation:nnN
\__color_backend_fill_devicen:nnN
\__color_backend_stroke_devicen:nnN
1043 \cs_new_protected:Npn \__color_backend_fill_separation:nnN #1#2#3
1044 { \__color_backend_fill:n { separation ~ #1 ~ #2 } }
1045 \cs_new_protected:Npn \__color_backend_stroke_separation:nnN #1#2#3
1046 { \__kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1047 \cs_new_eq:NN \__color_backend_fill_devicen:nnN \__color_backend_fill_separation:nnN
1048 \cs_new_eq:NN \__color_backend_stroke_devicen:nnN \__color_backend_stroke_separation:nnN

```

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

```

\__color_backend_fill_reset:
\__color_backend_stroke_reset:
1049 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1050 \cs_new_protected:Npn \__color_backend_stroke_reset: { }

```

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

```

1051 </dvips>
1052 <*dvisvgm>

```

```

\__color_backend_fill_cmyk:nN Fill color here is the same as general color.
\__color_backend_fill_gray:nN
\__color_backend_fill_rgb:nN
\__color_backend_fill:nN
1053 \cs_new_protected:Npn \__color_backend_fill_cmyk:nN #1
1054 { \__color_backend_fill:nN { cmyk ~ #1 } }
1055 \cs_new_protected:Npn \__color_backend_fill_gray:nN #1
1056 { \__color_backend_fill:nN { gray ~ #1 } }
1057 \cs_new_protected:Npn \__color_backend_fill_rgb:nN #1
1058 { \__color_backend_fill:nN { rgb ~ #1 } }
1059 \cs_new_protected:Npn \__color_backend_fill:nN #1#2
1060 {
1061 \tl_set:Nn #2 {#1}
1062 \__kernel_backend_literal:n { color~push~ #1 }
1063 }

```

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

__color_backend_stroke_cmyk:nN 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.

```

\__color_backend_stroke_gray:nN
\__color_backend_stroke_gray_aux:n
\__color_backend_stroke_rgb:nN
\__color_backend_stroke_rgb:w
\__color_backend:nnn
1064 \cs_new_protected:Npn \__color_backend_stroke_cmyk:nN #1#2
1065 {
1066 \__color_backend_fill_cmyk:n {#1}
1067 \__kernel_backend_scope:n { stroke = "{?color}" }
1068 \__color_backend_reset:
1069 }

```

```

1070 \cs_new_protected:Npn \__color_backend_stroke_gray:nN #1#2
1071 {
1072   \use:e
1073   {
1074     \__color_backend_stroke_gray_aux:n
1075     { \fp_eval:n { 100 * (#1) } }
1076   }
1077 }
1078 \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1079 { \__color_backend:nnn {#1} {#1} {#1} }
1080 \cs_new_protected:Npn \__color_backend_stroke_rgb:nN #1#2
1081 { \__color_backend_rgb:w #1 \s__color_stop }
1082 \cs_new_protected:Npn \__color_backend_stroke_rgb:w
1083 #1 ~ #2 ~ #3 \s__color_stop
1084 {
1085   \use:e
1086   {
1087     \__color_backend:nnn
1088     { \fp_eval:n { 100 * (#1) } }
1089     { \fp_eval:n { 100 * (#2) } }
1090     { \fp_eval:n { 100 * (#3) } }
1091   }
1092 }
1093 \cs_new_protected:Npe \__color_backend:nnn #1#2#3
1094 {
1095   \__kernel_backend_scope:n
1096   {
1097     stroke =
1098     "
1099     rgb
1100     (
1101       #1 \c_percent_str ,
1102       #2 \c_percent_str ,
1103       #3 \c_percent_str
1104     )
1105     "
1106   }
1107 }

```

(End of definition for __color_backend_stroke_cmyk:nN and others.)

At present, these are no-ops.

```

\__color_backend_fill_separation:nnN
\__color_backend_stroke_separation:nnN
\__color_backend_fill_devicen:nnN
\__color_backend_stroke_devicen:nnN
1108 \cs_new_protected:Npn \__color_backend_fill_separation:nnN #1#2#3 { }
1109 \cs_new_protected:Npn \__color_backend_stroke_separation:nnN #1#2#3 { }
1110 \cs_new_eq:NN \__color_backend_fill_devicen:nnN \__color_backend_fill_separation:nnN
1111 \cs_new_eq:NN \__color_backend_stroke_devicen:nnN \__color_backend_stroke_separation:nnN

```

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

```

\__color_backend_fill_reset:
\__color_backend_stroke_reset:
1112 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1113 \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
1114 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3 { }
1115 \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.)

1116 </dvisvgm>
1117 </package>

```

3.5 Font handling integration

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

```

1118 <*lua>
1119 local l = lpeg
1120 local spaces = l.P' '^0
1121 local digit16 = l.R('09', 'af', 'AF')
1122
1123 local octet = digit16 * digit16 / function(s)
1124   return string.format('%.3g ', tonumber(s, 16) / 255)
1125 end
1126
1127 if luaotfload and luaotfload.set_transparent_colorstack then
1128   local htmlcolor = l.Cs(octet * octet * octet * -1 * l.Cc'rg')
1129   local color_export = {
1130     token.create'tex_endlocalcontrol:D',
1131     token.create'tex_hpack:D',
1132     token.new(0, 1),
1133     token.create'color_export:nnN',
1134     token.new(0, 1),
1135     '',
1136     token.new(0, 2),
1137     token.new(0, 1),
1138     'backend',
1139     token.new(0, 2),
1140     token.create'l__color_tmp_tl',
1141     token.create'exp_after:wN',
1142     token.create'__color_select_aux:nnN',
1143     token.create'l__color_tmp_tl',
1144     token.create'l__color_tmp_tl',
1145     token.new(0, 2),
1146   }
1147   local group_end = token.create'group_end:'
1148   local value = (1 - l.P'}')^0
1149   luatexbase.add_to_callback('luaotfload.parse_color', function (value)
1150 % Also allow HTML colors to preserve compatibility
1151     local html = htmlcolor:match(value)
1152     if html then return html end
1153
1154 % If no l3color named color with this name is known, check for defined xcolor colors
1155     local l3color_prop = token.get_macro(string.format('l__color_named_%s_prop', value))
1156     if l3color_prop == nil or l3color_prop == '' then
1157       local legacy_color_macro = token.create(string.format('\\color@%s', value))

```

```

1158         if legacy_color_macro.cmdname ~= 'undefined_cs' then
1159             token.put_next(legacy_color_macro)
1160             return token.scan_argument()
1161         end
1162     end
1163
1164     tex.runtoks(function()
1165         token.get_next()
1166         color_export[6] = value
1167         tex.sprint(-2, color_export)
1168     end)
1169     local list = token.scan_list()
1170     if not list.head or list.head.next
1171         or list.head.subtype ~= node.subtype'pdf_colorstack' then
1172         error'Unexpected backend behavior'
1173     end
1174     local cmd = list.head.data
1175     node.free(list)
1176     return cmd
1177 end, 'l3color')
1178 end
1179  $\langle$ /lua $\rangle$ 
1180  $\langle$ *luatex $\rangle$ 
1181  $\langle$ *package $\rangle$ 
1182 \lua_load_module:n {l3backend-luatex}
1183  $\langle$ /package $\rangle$ 
1184  $\langle$ /luatex $\rangle$ 

```

4 l3backend-draw implementation

```

1185  $\langle$ *package $\rangle$ 
1186  $\langle$ @@=draw $\rangle$ 

```

4.1 dvips backend

```

1187  $\langle$ *dvips $\rangle$ 

```

$\backslash_draw_backend_literal:n$ The same as literal PostScript: same arguments about positioning apply here.

$\backslash_draw_backend_literal:e$ 1188 \cs_new_eq:NN _draw_backend_literal:n _kernel_backend_literal_postscript:n
1189 \cs_generate_variant:Nn _draw_backend_literal:n { e }

(End of definition for $\backslash_draw_backend_literal:n$.)

$\backslash_draw_backend_begin:$ The $\text{ps}::[\text{begin}]$ special here deals with positioning but allows us to continue on to a
 $\backslash_draw_backend_end:$ matching $\text{ps}::[\text{end}]$: contrast with $\text{ps}:$, which positions but where we can't split material
between separate calls. The $\text{@beginspecial}/\text{@endspecial}$ 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 $\text{@beginspecial}/\text{@endspecial}$ forms a
backend scope.)

```

1190 \cs_new_protected:Npn \_draw_backend_begin:
1191 {
1192     \_draw_backend_literal:n { [begin] }

```



```

1193     \__draw_backend_literal:n { /draw.x~currentpoint~/draw.y~exch~def~def }
1194     \__draw_backend_literal:n { @beginspecial }
1195   }
1196   \cs_new_protected:Npn \__draw_backend_end:
1197   {
1198     \__draw_backend_literal:n { @endspecial }
1199     \__draw_backend_literal:n { [end] }
1200   }

```

(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
 __draw_backend_scope_end: the graphic state has to be sent to the stack.

```

1201   \cs_new_protected:Npn \__draw_backend_scope_begin:
1202   { \__draw_backend_literal:n { save } }
1203   \cs_new_protected:Npn \__draw_backend_scope_end:
1204   { \__draw_backend_literal:n { restore } }

```

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

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

```

1205   \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1206   {
1207     \__draw_backend_literal:e
1208     {
1209       \dim_to_decimal_in_bp:n {#1} ~
1210       \dim_to_decimal_in_bp:n {#2} ~ moveto
1211     }
1212   }
1213   \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1214   {
1215     \__draw_backend_literal:e
1216     {
1217       \dim_to_decimal_in_bp:n {#1} ~
1218       \dim_to_decimal_in_bp:n {#2} ~ lineto
1219     }
1220   }
1221   \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1222   {
1223     \__draw_backend_literal:e
1224     {
1225       \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1226       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1227       moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1228     }
1229   }
1230   \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1231   {
1232     \__draw_backend_literal:e
1233     {

```

```

1234         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1235         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1236         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1237         curveto
1238     }
1239 }

```

(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
1240 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1241 { \bool_gset_true:N \g__draw_draw_eor_bool }
1242 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1243 { \bool_gset_false:N \g__draw_draw_eor_bool }
1244 \bool_new:N \g__draw_draw_eor_bool

```

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

```

\__draw_backend_closepath: Unlike PDF, PostScript doesn't track separate colors for strokes and other elements. It is
\__draw_backend_stroke: also desirable to have the clip keyword after a stroke or fill. To achieve those outcomes,
\__draw_backend_closestroke: there is some work to do. For color, the stroke color is simple but the fill one has to be
\__draw_backend_fill: inserted by hand. For clipping, the required ordering is achieved using a TEX switch.
\__draw_backend_fillstroke: All of the operations end with a new path instruction as they do not terminate (again in
\__draw_backend_clip: contrast to PDF).
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
1245 \cs_new_protected:Npn \__draw_backend_closepath:
1246 { \__draw_backend_literal:n { closepath } }
1247 \cs_new_protected:Npn \__draw_backend_stroke:
1248 {
1249     \__draw_backend_literal:n { gsave }
1250     \__draw_backend_literal:n { color.sc }
1251     \__draw_backend_literal:n { stroke }
1252     \__draw_backend_literal:n { grestore }
1253     \bool_if:NT \g__draw_draw_clip_bool
1254     {
1255         \__draw_backend_literal:e
1256         {
1257             \bool_if:NT \g__draw_draw_eor_bool { eo }
1258             clip
1259         }
1260     }
1261     \__draw_backend_literal:n { newpath }
1262     \bool_gset_false:N \g__draw_draw_clip_bool
1263 }
1264 \cs_new_protected:Npn \__draw_backend_closestroke:
1265 {
1266     \__draw_backend_closepath:
1267     \__draw_backend_stroke:
1268 }
1269 \cs_new_protected:Npn \__draw_backend_fill:
1270 {
1271     \__draw_backend_literal:e
1272     {
1273         \bool_if:NT \g__draw_draw_eor_bool { eo }

```

```

1274         fill
1275     }
1276     \bool_if:NT \g__draw_draw_clip_bool
1277     {
1278         \__draw_backend_literal:e
1279         {
1280             \bool_if:NT \g__draw_draw_eor_bool { eo }
1281             clip
1282         }
1283     }
1284     \__draw_backend_literal:n { newpath }
1285     \bool_gset_false:N \g__draw_draw_clip_bool
1286 }
1287 \cs_new_protected:Npn \__draw_backend_fillstroke:
1288 {
1289     \__draw_backend_literal:e
1290     {
1291         \bool_if:NT \g__draw_draw_eor_bool { eo }
1292         fill
1293     }
1294     \__draw_backend_literal:n { gsave }
1295     \__draw_backend_literal:n { color.sc }
1296     \__draw_backend_literal:n { stroke }
1297     \__draw_backend_literal:n { grestore }
1298     \bool_if:NT \g__draw_draw_clip_bool
1299     {
1300         \__draw_backend_literal:e
1301         {
1302             \bool_if:NT \g__draw_draw_eor_bool { eo }
1303             clip
1304         }
1305     }
1306     \__draw_backend_literal:n { newpath }
1307     \bool_gset_false:N \g__draw_draw_clip_bool
1308 }
1309 \cs_new_protected:Npn \__draw_backend_clip:
1310 { \bool_gset_true:N \g__draw_draw_clip_bool }
1311 \bool_new:N \g__draw_draw_clip_bool
1312 \cs_new_protected:Npn \__draw_backend_discardpath:
1313 {
1314     \bool_if:NT \g__draw_draw_clip_bool
1315     {
1316         \__draw_backend_literal:e
1317         {
1318             \bool_if:NT \g__draw_draw_eor_bool { eo }
1319             clip
1320         }
1321     }
1322     \__draw_backend_literal:n { newpath }
1323     \bool_gset_false:N \g__draw_draw_clip_bool
1324 }

```

(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:
1325 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1326 {
1327   \__draw_backend_literal:e
1328   {
1329     [
1330       \exp_args:Nf \use:n
1331       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1332     ] ~
1333     \dim_to_decimal_in_bp:n {#2} ~ setdash
1334   }
1335 }
1336 \cs_new:Npn \__draw_backend_dash:n #1
1337 { ~ \dim_to_decimal_in_bp:n {#1} }
1338 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1339 {
1340   \__draw_backend_literal:e
1341   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1342 }
1343 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1344 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1345 \cs_new_protected:Npn \__draw_backend_cap_but:
1346 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1347 \cs_new_protected:Npn \__draw_backend_cap_round:
1348 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1349 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1350 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1351 \cs_new_protected:Npn \__draw_backend_join_miter:
1352 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1353 \cs_new_protected:Npn \__draw_backend_join_round:
1354 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1355 \cs_new_protected:Npn \__draw_backend_join_bevel:
1356 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

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

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

```

1357 \cs_new_protected:Npn \__draw_backend_transform:nnnn #1#2#3#4
1358 {
1359   \__draw_backend_literal:n
1360   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1361 }
1362 \cs_new_protected:Npn \__draw_backend_shift:nn #1#2
1363 {
1364   \__draw_backend_literal:n
1365   { [ 1 ~ 0 ~ 0 ~ 1 ~ #1 ~ #2 ] ~ concat }
1366 }

```

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

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`.

```

1367 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1368 {
1369   \__draw_backend_literal:n { save }
1370   \__draw_backend_literal:n { 72~Resolution~div~72~VResolution~div~neg~scale }
1371   \__draw_backend_literal:n { magscale { 1~DVImag~div~dup~scale } if }
1372   \__draw_backend_literal:n { draw.x~neg~draw.y~neg~translate }
1373   \__draw_backend_literal:n { [end] }
1374   \__draw_backend_literal:n { [begin] }
1375   \__draw_backend_literal:n { save }
1376   \__draw_backend_literal:n { currentpoint }
1377   \__draw_backend_literal:n { currentpoint~translate }
1378   \__draw_backend_transform:nnnn { 1 } { 0 } { 0 } { -1 }
1379   \__draw_backend_transform:nnnn { #2 } { #3 } { #4 } { #5 }
1380   \__draw_backend_transform:nnnn { 1 } { 0 } { 0 } { -1 }
1381   \__draw_backend_literal:n { neg~exch~neg~exch~translate }
1382   \__draw_backend_literal:n { [end] }
1383   \hbox_overlap_right:n { \box_use:N #1 }
1384   \__draw_backend_literal:n { [begin] }
1385   \__draw_backend_literal:n { restore }
1386   \__draw_backend_literal:n { [end] }
1387   \__draw_backend_literal:n { [begin] }
1388   \__draw_backend_literal:n { restore }
1389 }

```

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

```

1390 </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.

```

1391 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

<code>__draw_backend_literal:n</code> <code>__draw_backend_literal:e</code>	Pass data through using a dedicated interface. <pre> 1392 \cs_new_eq:NN __draw_backend_literal:n __kernel_backend_literal_pdf:n 1393 \cs_new_eq:NN __draw_backend_literal:e __kernel_backend_literal_pdf:e </pre>
--	--

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

<code>__draw_backend_begin:</code> <code>__draw_backend_end:</code>	No special requirements here, so simply set up a drawing scope. <pre> 1394 \cs_new_protected:Npn __draw_backend_begin: 1395 { __draw_backend_scope_begin: } </pre>
--	---

```

1396 \cs_new_protected:Npn \__draw_backend_end:
1397 { \__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: 1398 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1399 \cs_new_eq:NN \__draw_backend_scope_end: \__kernel_backend_scope_end:

```

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

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

```

\__draw_backend_lineto:nn 1400 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
\__draw_backend_curveto:nnnnnn 1401 {
1402   \__draw_backend_literal:e
1403   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1404 }
1405 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1406 {
1407   \__draw_backend_literal:e
1408   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1409 }
1410 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1411 {
1412   \__draw_backend_literal:e
1413   {
1414     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1415     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1416     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1417     c
1418   }
1419 }
1420 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1421 {
1422   \__draw_backend_literal:e
1423   {
1424     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1425     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1426     re
1427   }
1428 }

```

(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: 1429 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
\g__draw_draw_eor_bool 1430 { \bool_gset_true:N \g__draw_draw_eor_bool }
1431 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1432 { \bool_gset_false:N \g__draw_draw_eor_bool }
1433 \bool_new:N \g__draw_draw_eor_bool

```

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

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

```

\__draw_backend_closepath:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
1434 \cs_new_protected:Npn \__draw_backend_closepath:
1435 { \__draw_backend_literal:n { h } }
1436 \cs_new_protected:Npn \__draw_backend_stroke:
1437 { \__draw_backend_literal:n { S } }
1438 \cs_new_protected:Npn \__draw_backend_closestroke:
1439 { \__draw_backend_literal:n { s } }
1440 \cs_new_protected:Npn \__draw_backend_fill:
1441 {
1442   \__draw_backend_literal:e
1443   { f \bool_if:NT \g__draw_draw_eor_bool * }
1444 }
1445 \cs_new_protected:Npn \__draw_backend_fillstroke:
1446 {
1447   \__draw_backend_literal:e
1448   { B \bool_if:NT \g__draw_draw_eor_bool * }
1449 }
1450 \cs_new_protected:Npn \__draw_backend_clip:
1451 {
1452   \__draw_backend_literal:e
1453   { W \bool_if:NT \g__draw_draw_eor_bool * }
1454 }
1455 \cs_new_protected:Npn \__draw_backend_discardpath:
1456 { \__draw_backend_literal:n { n } }

```

(End of definition for __draw_backend_closepath: and others.)

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1457 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1458 {
1459   \__draw_backend_literal:e
1460   {
1461     [
1462       \exp_args:Nf \use:n
1463       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1464     ] ~
1465     \dim_to_decimal_in_bp:n {#2} ~ d
1466   }
1467 }
1468 \cs_new:Npn \__draw_backend_dash:n #1
1469 { ~ \dim_to_decimal_in_bp:n {#1} }
1470 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1471 {
1472   \__draw_backend_literal:e
1473   { \dim_to_decimal_in_bp:n {#1} ~ w }
1474 }
1475 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1476 { \__draw_backend_literal:e { #1 ~ M } }
1477 \cs_new_protected:Npn \__draw_backend_cap_butt:
1478 { \__draw_backend_literal:n { 0 ~ J } }
1479 \cs_new_protected:Npn \__draw_backend_cap_round:
1480 { \__draw_backend_literal:n { 1 ~ J } }
1481 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1482 { \__draw_backend_literal:n { 2 ~ J } }

```

```

1483 \cs_new_protected:Npn \__draw_backend_join_miter:
1484 { \__draw_backend_literal:n { 0 ~ j } }
1485 \cs_new_protected:Npn \__draw_backend_join_round:
1486 { \__draw_backend_literal:n { 1 ~ j } }
1487 \cs_new_protected:Npn \__draw_backend_join_bevel:
1488 { \__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 to decompose the matrix into rotations and a scaling, then use those operations as they are handled by the backend. (There is backend support for matrix operations in dvipdfmx/X_YTeX, but as a matched pair so not suitable for the “stand alone” transformation set up here.) The specials used here are from xdvipdfmx originally: they are well-tested, but probably equivalent to the pdf₊ versions! As working out the rotation is relatively expensive, we optimize for the case where there is only a scaling.

```

1489 \cs_new_protected:Npn \__draw_backend_transform:nnnn #1#2#3#4
1490 {
1491   < *luatex | pdftex >
1492   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1493   < /luatex | pdftex >
1494   < *dvipdfmx | xetex >
1495   \str_if_eq:nnTF { #2 ~ #3 } { 0 ~ 0 }
1496   {
1497     \__kernel_backend_literal:n { x:rotate~0 }
1498     \__kernel_backend_literal:n { x:scale~#1~#4 }
1499     \__kernel_backend_literal:n { x:rotate~0 }
1500   }
1501   {
1502     \__draw_backend_transform_decompose:nnnnN {#1} {#2} {#3} {#4}
1503     \__draw_backend_transform_aux:nnnn
1504   }
1505   < /dvipdfmx | xetex >
1506 }
1507 < *dvipdfmx | xetex >
1508 \cs_new_protected:Npn \__draw_backend_transform_aux:nnnn #1#2#3#4
1509 {
1510   \__kernel_backend_literal:e
1511   {
1512     x:rotate~
1513     \fp_compare:nNnTF {#1} = \c_zero_fp
1514     { 0 }
1515     { \fp_eval:n { round ( -#1 , 5 ) } }
1516   }
1517   \__kernel_backend_literal:e
1518   {
1519     x:scale~
1520     \fp_eval:n { round ( #2 , 5 ) } ~
1521     \fp_eval:n { round ( #3 , 5 ) }
1522   }
1523   \__kernel_backend_literal:e
1524   {

```



```

1525         x:rotate~
1526         \fp_compare:nNnTF {#4} = \c_zero_fp
1527         { 0 }
1528         { \fp_eval:n { round ( -#4 , 5 ) } }
1529     }
1530 }
1531 </dviptdmtx | 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.

```

1532 \cs_new_protected:Npn \__draw_backend_shift:nn #1#2
1533 {
1534     \__draw_backend_literal:n
1535     { 1 ~ 0 ~ 0 ~ 1 ~ #1 ~ #2 ~ cm }
1536 }

```

(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.

```

1537 <*dviptdmtx | xetex>
1538 \cs_new_protected:Npn \__draw_backend_transform_decompose:nnnnN #1#2#3#4#5
1539 {
1540     \use:e
1541     {
1542         \__draw_backend_transform_decompose_auxi:nnnnN
1543         { \fp_eval:n { (#1 + #4) / 2 } }
1544         { \fp_eval:n { (#1 - #4) / 2 } }

```

```

1545         { \fp_eval:n { (#3 + #2) / 2 } }
1546         { \fp_eval:n { (#3 - #2) / 2 } }
1547     }
1548     #5
1549 }
1550 \cs_new_protected:Npn \__draw_backend_transform_decompose_auxii:nnnnN #1#2#3#4#5
1551 {
1552     \use:e
1553     {
1554         \__draw_backend_transform_decompose_auxiii:nnnnN
1555         { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1556         { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1557         { \fp_eval:n { atand ( #3 , #2 ) } }
1558         { \fp_eval:n { atand ( #4 , #1 ) } }
1559     }
1560     #5
1561 }
1562 \cs_new_protected:Npn \__draw_backend_transform_decompose_auxiii:nnnnN #1#2#3#4#5
1563 {
1564     \use:e
1565     {
1566         \__draw_backend_transform_decompose_auxiiii:nnnnN
1567         { \fp_eval:n { ( #4 - #3 ) / 2 } }
1568         { \fp_eval:n { ( #1 + #2 ) / 2 } }
1569         { \fp_eval:n { ( #1 - #2 ) / 2 } }
1570         { \fp_eval:n { ( #4 + #3 ) / 2 } }
1571     }
1572     #5
1573 }
1574 \cs_new_protected:Npn \__draw_backend_transform_decompose_auxiiii:nnnnN #1#2#3#4#5
1575 {
1576     \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1577     { #5 {#1} {#2} {#3} {#4} }
1578     { #5 {#1} {#3} {#2} {#4} }
1579 }
1580 \</dvi\pdfmx | xetex>

```

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

__draw_backend_box_use:Nnnnn Inserting a T_EX box transformed to the requested position and using the current matrix is done using a mixture of T_EX and low-level manipulation. The offset can be handled by T_EX, so only any rotation/skew/scaling component needs to be done using the matrix operation. As this operation can never be cached, the scope is set directly not using the draw version.

```

1581 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1582 {
1583     \__kernel_backend_scope_begin:
1584     < *luatex | pdftex >
1585     \__kernel_backend_matrix:n { #2 ~ #3 ~ #4 ~ #5 }
1586     < /luatex | pdftex >
1587     < *dvi\pdfmx | xetex >
1588     \__kernel_backend_literal:n
1589     { pdf:btrans-matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1590     < /dvi\pdfmx | xetex >

```

```

1591 \hbox_overlap_right:n { \box_use:N #1 }
1592 <*dvipdfmx | xetex>
1593 \__kernel_backend_literal:n { pdf:ettrans }
1594 </dvipdfmx | xetex>
1595 \__kernel_backend_scope_end:
1596 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1597 </dvipdfmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1598 <*dvisvgm>

```

The same as the more general literal call.

```

\__draw_backend_literal:n
\__draw_backend_literal:e
1599 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1600 \cs_generate_variant:Nn \__draw_backend_literal:n { e }

```

(End of definition for __draw_backend_literal:n.)

Use the backend-level scope mechanisms.

```

\__draw_backend_scope_begin:
\__draw_backend_scope_end:
1601 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1602 \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:.)

A drawing needs to be set up such that the coordinate system is translated. That is done inside a scope, which as described below

```

1603 \cs_new_protected:Npn \__draw_backend_begin:
1604 {
1605   \__kernel_backend_scope_begin:
1606   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1607 }
1608 \cs_new_eq:NN \__draw_backend_end: \__kernel_backend_scope_end:

```

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

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

```

\__draw_backend_moveto:nn
\__draw_backend_lineto:nn
\__draw_backend_rectangle:nnnn
\__draw_backend_curveto:nnnnnn
\__draw_backend_add_to_path:n
\g__draw_backend_path_tl
1609 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1610 {
1611   \__draw_backend_add_to_path:n
1612   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1613 }
1614 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1615 {
1616   \__draw_backend_add_to_path:n
1617   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1618 }
1619 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1620 {

```

```

1621 \__draw_backend_add_to_path:n
1622 {
1623     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1624     h ~ \dim_to_decimal:n {#3} ~
1625     v ~ \dim_to_decimal:n {#4} ~
1626     h ~ \dim_to_decimal:n { -#3 } ~
1627     Z
1628 }
1629 }
1630 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1631 {
1632     \__draw_backend_add_to_path:n
1633     {
1634         C ~
1635         \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1636         \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1637         \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1638     }
1639 }
1640 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1641 {
1642     \tl_gset:Nx \g__draw_backend_path_tl
1643     {
1644         \g__draw_backend_path_tl
1645         \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1646         #1
1647     }
1648 }
1649 \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:
1650 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1651 { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1652 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1653 { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

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

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

```

\__draw_backend_fillstroke:
1654 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_clip:
1655 { \__draw_backend_add_to_path:n { Z } }
\__draw_backend_discardpath:
1656 \cs_new_protected:Npn \__draw_backend_path:n #1
\g__draw_draw_clip_bool
1657 {
\g__draw_draw_path_int
1658     \bool_if:NTF \g__draw_draw_clip_bool
1659     {
1660         \int_gincr:N \g__kernel_clip_path_int
1661         \__draw_backend_literal:e
1662         {

```

```

1663         < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1664             { ?nl }
1665         <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1666         < /clipPath > { ? nl }
1667         <
1668             use~xlink:href =
1669                 "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1670                 #1
1671             />
1672         }
1673         \__kernel_backend_scope:e
1674         {
1675             clip-path =
1676                 "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1677         }
1678     }
1679     {
1680         \__draw_backend_literal:e
1681         { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1682     }
1683     \tl_gclear:N \g__draw_backend_path_tl
1684     \bool_gset_false:N \g__draw_draw_clip_bool
1685 }
1686 \int_new:N \g__draw_backend_path_int
1687 \cs_new_protected:Npn \__draw_backend_stroke:
1688 { \__draw_backend_path:n { style="fill:none" } }
1689 \cs_new_protected:Npn \__draw_backend_closestroke:
1690 {
1691     \__draw_backend_closepath:
1692     \__draw_backend_stroke:
1693 }
1694 \cs_new_protected:Npn \__draw_backend_fill:
1695 { \__draw_backend_path:n { style="stroke:none" } }
1696 \cs_new_protected:Npn \__draw_backend_fillstroke:
1697 { \__draw_backend_path:n { } }
1698 \cs_new_protected:Npn \__draw_backend_clip:
1699 { \bool_gset_true:N \g__draw_draw_clip_bool }
1700 \bool_new:N \g__draw_draw_clip_bool
1701 \cs_new_protected:Npn \__draw_backend_discardpath:
1702 {
1703     \bool_if:NT \g__draw_draw_clip_bool
1704     {
1705         \int_gincr:N \g__kernel_clip_path_int
1706         \__draw_backend_literal:e
1707         {
1708             < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1709                 { ?nl }
1710             <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1711             < /clipPath >
1712         }
1713         \__kernel_backend_scope:e
1714         {
1715             clip-path =
1716                 "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"

```

```

1717     }
1718   }
1719   \tl_gclear:N \g__draw_backend_path_tl
1720   \bool_gset_false:N \g__draw_draw_clip_bool
1721 }

```

(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).

```

1722 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1723 {
1724   \use:e
1725   {
1726     \__draw_backend_dash_aux:nn
1727     { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1728     { \dim_to_decimal:n {#2} }
1729   }
1730 }
1731 \cs_new:Npn \__draw_backend_dash:n #1
1732 { , \dim_to_decimal_in_bp:n {#1} }
1733 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1734 {
1735   \__kernel_backend_scope:e
1736   {
1737     stroke-dasharray =
1738     "
1739     \tl_if_empty:nTF {#1}
1740     { none }
1741     { \use_none:n #1 }
1742     " ~
1743     stroke-offset=" #2 "
1744   }
1745 }
1746 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1747 { \__kernel_backend_scope:e { stroke-width=" \dim_to_decimal:n {#1} " } }
1748 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1749 { \__kernel_backend_scope:e { stroke-miterlimit=" #1 " } }
1750 \cs_new_protected:Npn \__draw_backend_cap_but:
1751 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1752 \cs_new_protected:Npn \__draw_backend_cap_round:
1753 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1754 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1755 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1756 \cs_new_protected:Npn \__draw_backend_join_miter:
1757 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1758 \cs_new_protected:Npn \__draw_backend_join_round:
1759 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1760 \cs_new_protected:Npn \__draw_backend_join_bevel:
1761 { \__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.

```

1762 \cs_new_protected:Npn \__draw_backend_transform:nnnn #1#2#3#4
1763 {
1764   \__kernel_backend_scope:n
1765   {
1766     transform =
1767     " matrix ( #1 , #2 , #3 , #4 , 0pt , 0pt ) "
1768   }
1769 }
1770 \cs_new_protected:Npn \__draw_backend_shift:nn #1#2
1771 {
1772   \__kernel_backend_scope:n
1773   {
1774     transform =
1775     " matrix ( 1 , 0 , 0 , 1 , #1pt , #2pt ) "
1776   }
1777 }

```

(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.

```

1778 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1779 {
1780   \__kernel_backend_scope_begin:
1781   \__draw_backend_transform:nnnn {#2} {#3} {#4} {#5}
1782   \__kernel_backend_literal_svg:n
1783   {
1784     < g~
1785     stroke="none"~
1786     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1787     >
1788   }
1789   \box_set_wd:Nn #1 { 0pt }
1790   \box_set_ht:Nn #1 { 0pt }
1791   \box_set_dp:Nn #1 { 0pt }
1792   \box_use:N #1
1793   \__kernel_backend_literal_svg:n { </g> }
1794   \__kernel_backend_scope_end:
1795 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1796 </dvisvgm>
1797 </package>

```

5 l3backend-graphics implementation

```

1798 <*package>
1799 <@@=graphics>

```

5.1 dvips backend

```

1800 <*dvips>

```

\l_graphics_search_ext_seq

```

1801 \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.

```
1802 \cs_new_eq:NN \_graphics_backend_getbb_eps:n \_graphics_read_bb:n
1803 \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.

```
1804 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1805 {
1806   \_kernel_backend_literal:e
1807   {
1808     PSfile = #1 \c_space_tl
1809     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1810     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1811     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1812     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1813   }
1814 }
1815 \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`

```
1816 \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n
```

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

```
1817 </dvips>
```

5.2 LuaTeX and pdfTeX backends

```
1818 <*\luatex | pdftex>
```

`\l_graphics_search_ext_seq`

```
1819 \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1820 { .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.

```
1821 \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.

```
1822 \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.

```

1823 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1824 {
1825   \int_zero:N \l__graphics_page_int
1826   \tl_clear:N \l__graphics_pagebox_tl
1827   \bool_set_false:N \l__graphics_transgroup_bool
1828   \tl_set:Ne \l__graphics_attr_tl
1829   {
1830     \tl_if_empty:NF \l__graphics_decodearray_str
1831     { :D \l__graphics_decodearray_str }
1832     \bool_if:NT \l__graphics_interpolate_bool
1833     { :I }
1834     \str_if_empty:NF \l__graphics_pdf_str
1835     { :X \l__graphics_pdf_str }
1836   }
1837   \__graphics_backend_getbb_auxi:n {#1}
1838 }
1839 \cs_new_eq:NN \_graphics_backend_getbb_jpeg:n \_graphics_backend_getbb_jpg:n
1840 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1841 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1842 {
1843   \tl_clear:N \l__graphics_decodearray_str
1844   \bool_set_true:N \l__graphics_transgroup_bool
1845   \bool_set_false:N \l__graphics_interpolate_bool
1846   \tl_set:Ne \l__graphics_attr_tl
1847   {
1848     : \l__graphics_pagebox_tl
1849     \int_compare:nNnT \l__graphics_page_int > 1
1850     { :P \int_use:N \l__graphics_page_int }
1851     \str_if_empty:NF \l__graphics_pdf_str
1852     { :X \l__graphics_pdf_str }
1853   }
1854   \__graphics_backend_getbb_auxi:n {#1}
1855 }
1856 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:n #1
1857 {
1858   \__graphics_bb_restore:eF { #1 \l__graphics_attr_tl }
1859   { \__graphics_backend_getbb_auxiii:n {#1} }
1860 }

```

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.

```

1861 \cs_new_protected:Npn \_graphics_backend_getbb_auxiii:n #1
1862 {
1863   \exp_args:Ne \_graphics_backend_getbb_auxiii:n
1864   { \_graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1865   \int_const:cn { c__graphics_ #1 \l__graphics_attr_tl _int }

```

```

1866     { \tex_the:D \tex_pdflastximage:D }
1867     \__graphics_bb_save:e { #1 \l__graphics_attr_tl }
1868   }
1869   \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1870   {
1871     \tex_immediate:D \tex_pdfximage:D
1872     \bool_lazy_any:nT
1873     {
1874       { \l__graphics_interpolate_bool }
1875       { \l__graphics_transgroup_bool }
1876       { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1877       { ! \str_if_empty_p:N \l__graphics_pdf_str }
1878     }
1879     {
1880       attr
1881       {
1882         \tl_if_empty:NF \l__graphics_decodearray_str
1883         { /Decode~[ \l__graphics_decodearray_str ] }
1884         \bool_if:NT \l__graphics_transgroup_bool
1885         { /Group << /S /Transparency /K ~ false /I ~ false >> }
1886         \bool_if:NT \l__graphics_interpolate_bool
1887         { /Interpolate~true }
1888         \l__graphics_pdf_str
1889       }
1890     }
1891     \int_compare:nNnT \l__graphics_page_int > 0
1892     { page ~ \int_use:N \l__graphics_page_int }
1893     \tl_if_empty:NF \l__graphics_pagebox_tl
1894     { \l__graphics_pagebox_tl }
1895     {#1}
1896     \hbox_set:Nn \l__graphics_tmp_box
1897     { \tex_pdfrefximage:D \tex_pdflastximage:D }
1898     \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_tmp_box }
1899     \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_tmp_box }
1900   }
1901   \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.

```

1902   \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1903   {
1904     \tex_pdfrefximage:D
1905     \int_use:c { c__graphics_ #1 \l__graphics_attr_tl_int }
1906   }
1907   \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1908   \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1909   \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.)

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.

```

\__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
1910 \sys_if_shell:T
1911 {
1912   \str_new:N \l__graphics_backend_dir_str
1913   \str_new:N \l__graphics_backend_name_str
1914   \str_new:N \l__graphics_backend_ext_str
1915   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1916   {
1917     \file_parse_full_name:nNNN {#1}
1918     \l__graphics_backend_dir_str
1919     \l__graphics_backend_name_str
1920     \l__graphics_backend_ext_str
1921     \exp_args:Ne \__graphics_backend_getbb_eps:nn
1922     {
1923       \exp_args:Ne \__kernel_file_name_quote:n
1924       {
1925         \l__graphics_backend_name_str
1926         - \str_tail:N \l__graphics_backend_ext_str
1927         -converted-to.pdf
1928       }
1929     }
1930     {#1}
1931   }
1932   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_backend_getbb_eps:n
1933   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1934   {
1935     \file_compare_timestamp:nNnT {#2} > {#1}
1936     {
1937       \sys_shell_now:n
1938       { repstopdf ~ #2 ~ #1 }
1939     }
1940     \tl_set:Nn \l__graphics_final_name_str {#1}
1941     \__graphics_backend_getbb_pdf:n {#1}
1942   }
1943   \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1944   {
1945     \file_parse_full_name:nNNN {#1}
1946     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1947     \exp_args:Ne \__graphics_backend_include_pdf:n
1948     {
1949       \exp_args:Ne \__kernel_file_name_quote:n
1950       {
1951         \l__graphics_backend_name_str
1952         - \str_tail:N \l__graphics_backend_ext_str
1953         -converted-to.pdf
1954       }
1955     }
1956   }
1957   \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1958 }

```

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

`_graphics_backend_get_pagecount:n`

Simply load and store.

```
1959 \cs_new_protected:Npn \_graphics_backend_get_pagecount:n #1
1960 {
1961   \tex_pdfximage:D {#1}
1962   \int_const:cn { c__graphics_ #1 _pages_int }
1963   { \int_use:N \tex_pdflastximagepages:D }
1964 }

(End of definition for \_graphics_backend_get_pagecount:n.)

1965 </luatex | pdftex>
```

5.3 dvipdfmx backend

```
1966 <*dvipdfmx | xetex>
```

`\l_graphics_search_ext_seq`

```
1967 \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1968 { .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`

```
1969 \cs_new_eq:NN \_graphics_backend_getbb_eps:n \_graphics_read_bb:n
```

`_graphics_backend_getbb_jpg:n`

```
1970 \cs_new_eq:NN \_graphics_backend_getbb_ps:n \_graphics_read_bb:n
```

`_graphics_backend_getbb_jpeg:n`

```
1971 <*dvipdfmx>
```

`_graphics_backend_getbb_pdf:n`

```
1972 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
```

`_graphics_backend_getbb_png:n`

```
1973 {
```

`_graphics_backend_getbb_bmp:n`

```
1974   \int_zero:N \l__graphics_page_int
```

```
1975   \tl_clear:N \l__graphics_pagebox_tl
```

```
1976   \_graphics_extract_bb:n {#1}
```

```
1977 }
```

```
1978 \cs_new_eq:NN \_graphics_backend_getbb_jpeg:n \_graphics_backend_getbb_jpg:n
```

```
1979 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
```

```
1980 \cs_new_eq:NN \_graphics_backend_getbb_bmp:n \_graphics_backend_getbb_jpg:n
```

```
1981 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
```

```
1982 {
```

```
1983   \tl_clear:N \l__graphics_decodearray_str
```

```
1984   \bool_set_false:N \l__graphics_interpolate_bool
```

```
1985   \_graphics_extract_bb:n {#1}
```

```
1986 }
```

```
1987 </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.

```
1988 \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.

```
1989 \int_new:N \g__graphics_track_int
```

(End of definition for `\g__graphics_track_int`.)

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

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

```

1990 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1991 {
1992   \__kernel_backend_literal:e
1993   {
1994     PSfile = #1 \c_space_tl
1995     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1996     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1997     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1998     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1999   }
2000 }
2001 \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.

```

2002 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
2003 {
2004   \bool_set_false:N \l__graphics_transgroup_bool
2005   \_graphics_backend_include_auxi:n {#1}
2006 }
2007 \cs_new_eq:NN \_graphics_backend_include_jpeg:n \_graphics_backend_include_jpg:n
2008 \cs_new_eq:NN \_graphics_backend_include_bmp:n \_graphics_backend_include_jpg:n
2009 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
2010 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
2011 {
2012   \bool_set_true:N \l__graphics_transgroup_bool
2013   \_graphics_backend_include_auxi:n {#1}
2014 }
2015 \cs_new_protected:Npn \_graphics_backend_include_auxi:n #1
2016 {
2017   \_graphics_backend_include_auxii:en
2018   {
2019     \tl_if_empty:NF \l__graphics_pagebox_tl
2020     { : \l__graphics_pagebox_tl }
2021     \int_compare:nNnT \l__graphics_page_int > 1
2022     { :P \int_use:N \l__graphics_page_int }
2023     \tl_if_empty:NF \l__graphics_decodearray_str
2024     { :D \l__graphics_decodearray_str }
2025     \bool_if:NT \l__graphics_interpolate_bool
2026     { :I }
2027   }
2028   {#1}
2029 }
2030 \cs_new_protected:Npn \_graphics_backend_include_auxii:nn #1#2
2031 {
2032   \int_if_exist:cTF { c__graphics_ #2#1 _int }
2033   {
2034     \__kernel_backend_literal:e
2035     { pdf:usexobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }

```

```

2036     }
2037     { \_graphics_backend_include_auxiii:nn {#2} {#1} }
2038   }
2039   \cs_generate_variant:Nn \_graphics_backend_include_auxii:nn { e }

2040   \cs_new_protected:Npn \_graphics_backend_include_auxiii:nn #1#2
2041   {
2042     \int_gincr:N \g__graphics_track_int
2043     \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
2044     \__kernel_backend_literal:e
2045     {
2046       pdf:image ~
2047       @graphic \int_use:c { c__graphics_ #1#2 _int } ~
2048       \int_compare:nNnT \l__graphics_page_int > 1
2049       { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2050       \tl_if_empty:NF \l__graphics_pagebox_tl
2051       {
2052         pagebox ~ \l__graphics_pagebox_tl \c_space_tl
2053         bbox ~
2054         \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2055         \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2056         \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2057         \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
2058       }
2059       (#1)
2060       \bool_lazy_any:nT
2061       {
2062         { \l__graphics_interpolate_bool }
2063         { \l__graphics_transgroup_bool }
2064         { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
2065       }
2066       {
2067         <<
2068         \tl_if_empty:NF \l__graphics_decodearray_str
2069         { /Decode~[ \l__graphics_decodearray_str ] }
2070         \bool_if:NT \l__graphics_transgroup_bool
2071         { /Group << /S /Transparency /K ~ false /I ~ false >> }
2072         \bool_if:NT \l__graphics_interpolate_bool
2073         { /Interpolate~true }
2074       >>
2075     }
2076   }
2077 }

```

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

_graphics_backend_get_pagecount:n

```

2078 <*\dvipdfmx>
2079 \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n
2080 </dvipdfmx>

```

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

2081 `</dvipdfmx | xetex>`

5.4 X_YT_EX backend

2082 `<*xetex>`

For X_YT_EX, 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_YT_EX primitive omits the text box from the page box specification, so there is also some “trimming” to do here.

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_jpeg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
\__graphics_backend_getbb_bmp:n
\__graphics_backend_getbb_auxi:nN
\__graphics_backend_getbb_auxii:nnN
\__graphics_backend_getbb_auxii:VnN
\__graphics_backend_getbb_auxiii:nNnn
\__graphics_backend_getbb_auxiv:nnNnn
\__graphics_backend_getbb_auxiv:VnNnn
\__graphics_backend_getbb_auxv:nNnn
\__graphics_backend_getbb_auxv:nNnn
\__graphics_backend_getbb_pagebox:w

2083 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2084 {
2085   \int_zero:N \l__graphics_page_int
2086   \tl_clear:N \l__graphics_pagebox_tl
2087   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2088 }
2089 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2090 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
2091 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2092 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2093 {
2094   \tl_clear:N \l__graphics_decodearray_str
2095   \bool_set_false:N \l__graphics_interpolate_bool
2096   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2097 }
2098 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2099 {
2100   \int_compare:nNnTF \l__graphics_page_int > 1
2101     { \__graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2102     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2103 }
2104 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2105 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2106 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2107 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2108 {
2109   \tl_if_empty:NTF \l__graphics_pagebox_tl
2110     { \__graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2111     { \__graphics_backend_getbb_auxv:nNnn {
2112       {#1} #2 {#3} {#4}
2113     } }
2114 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2115 {
2116   \use:e
2117   {
2118     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2119     {
2120       #5
2121       \tl_if_blank:nF {#1}
2122         { \c_space_tl \__graphics_backend_getbb_pagebox:w #1 }
2123     }
2124   }

```

```

2125 }
2126 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2127 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2128 {
2129   \__graphics_bb_restore:nF {#1#3}
2130   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2131 }
2132 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2133 {
2134   \hbox_set:Nn \l__graphics_tmp_box { #2 #1 ~ #4 }
2135   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_tmp_box }
2136   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_tmp_box }
2137   \__graphics_bb_save:n {#1#3}
2138 }
2139 \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.

```

2140 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
2141 {
2142   \int_const:cn { c__graphics_ #1 _pages_int }
2143   {
2144     \int_max:nn
2145     { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2146     { 1 }
2147   }
2148 }

```

(End of definition for __graphics_backend_get_pagecount:n.)

2149 \langle /xetex \rangle

5.5 dvisvgm backend

2150 \langle *dvisvgm \rangle

\l_graphics_search_ext_seq

```

2151 \seq_set_from_clist:Nn \l_graphics_search_ext_seq
2152 { .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.

```

2153 \cs_new_protected:Npn \__graphics_backend_getbb_svg:n #1
2154 {
2155   \__graphics_bb_restore:nF {#1}
2156   {
2157     \ior_open:Nn \l__graphics_tmp_ior {#1}
2158     \ior_if_eof:NTF \l__graphics_tmp_ior
2159     { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2160     {

```



```

2161 \dim_zero:N \l__graphics_llx_dim
2162 \dim_zero:N \l__graphics_lly_dim
2163 \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2164 \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2165 \ior_str_map_inline:Nn \l__graphics_tmp_ior
2166 {
2167   \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2168   {
2169     \__graphics_backend_getbb_svg_auxi:nNn
2170     { width } \l__graphics_urx_dim {##1}
2171   }
2172   \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2173   {
2174     \__graphics_backend_getbb_svg_auxi:nNn
2175     { height } \l__graphics_ury_dim {##1}
2176   }
2177   \bool_lazy_and:nnF
2178   { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2179   { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2180   { \ior_map_break: }
2181 }
2182 \__graphics_bb_save:n {#1}
2183 }
2184 \ior_close:N \l__graphics_tmp_ior
2185 }
2186 }
2187 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2188 {
2189   \use:e
2190   {
2191     \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxii:w
2192     ##1 \tl_to_str:n {#1} = ##2 \tl_to_str:n {#1} = ##3
2193     \s__graphics_stop
2194   }
2195   {
2196     \tl_if_blank:nF {##2}
2197     {
2198       \peek_remove_spaces:n
2199       {
2200         \peek_meaning:NTF ' % '
2201         { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2202         {
2203           \peek_meaning:NTF " % "
2204           { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2205           { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2206         }
2207       }
2208       ##2 \s__graphics_stop
2209     }
2210   }
2211   \use:e
2212   {
2213     \__graphics_backend_getbb_svg_auxii:w #3
2214     \tl_to_str:n {#1} = \tl_to_str:n {#1} =

```

```

2215         \s__graphics_stop
2216     }
2217 }
2218 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2219 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop
2220 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2221 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2222 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2223 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2224 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2225 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2226 {
2227     \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2228     \l__graphics_tmp_dim #2 bp \scan_stop:
2229     \dim_set_eq:NN #1 \l__graphics_tmp_dim
2230 }
2231 \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.

```

2232 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2233 \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.

```

2234 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2235 {
2236     \int_zero:N \l__graphics_page_int
2237     \tl_clear:N \l__graphics_pagebox_tl
2238     \__graphics_extract_bb:n {#1}
2239 }
2240 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2241 \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

```

2242 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2243 {
2244     \tl_clear:N \l__graphics_decodearray_str
2245     \bool_set_false:N \l__graphics_interpolate_bool
2246     \__graphics_extract_bb:n {#1}
2247 }

```

(End of definition for __graphics_backend_getbb_pdf:n.)

__graphics_backend_include_eps:n
__graphics_backend_include_ps:n
__graphics_backend_include_pdf:n
__graphics_backend_include_nn

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

```

2248 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2249 { \__graphics_backend_include:nn { PSfile } {#1} }
2250 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
2251 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1

```

```

2252 { \_graphics_backend_include:nn { pdffile } {#1} }
2253 \cs_new_protected:Npn \_graphics_backend_include:nn #1#2
2254 {
2255   \_kernel_backend_literal:e
2256   {
2257     #1 = #2 \c_space_tl
2258     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2259     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2260     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2261     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2262   }
2263 }

```

(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.

```

2264 \cs_new_protected:Npn \_graphics_backend_include_svg:n #1
2265 {
2266   \box_move_up:nn { \l__graphics_ury_dim }
2267   {
2268     \hbox:n
2269     {
2270       \_kernel_backend_literal:e
2271       {
2272         dvisvgm:img~
2273         \dim_to_decimal:n { \l__graphics_urx_dim } ~
2274         \dim_to_decimal:n { \l__graphics_ury_dim } ~
2275         \_graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2276       }
2277     }
2278   }
2279 }
2280 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_svg:n
2281 \cs_new_eq:NN \_graphics_backend_include_jpeg:n \_graphics_backend_include_svg:n
2282 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_svg:n
2283 \cs_new:Npn \_graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2284 {#2}

```

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

_graphics_backend_get_pagecount:n

```

2285 \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n

```

(End of definition for _graphics_backend_get_pagecount:n.)

```

2286 </dvisvgm>
2287 </package>

```

6 l3backend-pdf implementation

2288 $\langle *package \rangle$
 2289 $\langle @@=pdf \rangle$

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

2290 $\langle *dvips \rangle$

`__pdf_backend_pdfmark:n`
`__pdf_backend_pdfmark:e`

Used often enough it should be a separate function.

2291 `\cs_new_protected:Npn __pdf_backend_pdfmark:n #1`
 2292 `{ __kernel_backend_postscript:n { mark #1 ~ pdfmark } }`
 2293 `\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`

2294 `\cs_new_protected:Npn __pdf_backend_catalog_gput:nn #1#2`
 2295 `{ __pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }`
 2296 `\cs_new_protected:Npn __pdf_backend_info_gput:nn #1#2`
 2297 `{ __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`

2298 `\cs_new_protected:Npn __pdf_backend_object_new:`
 2299 `{ \int_gincr:N \g__pdf_backend_object_int }`
 2300 `\cs_new:Npn __pdf_backend_object_ref:n #1 { { pdf.obj #1 } }`
 2301 `\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.

2302 `\cs_new_protected:Npn __pdf_backend_object_write:nnn #1#2#3`
 2303 `{`
 2304 `__pdf_backend_object_write_aux:nnn`
 2305 `{ __pdf_backend_object_ref:n {#1} }`
 2306 `{#2} {#3}`
 2307 `}`
 2308 `\cs_generate_variant:Nn __pdf_backend_object_write:nnn { nne }`
 2309 `\cs_new_protected:Npn __pdf_backend_object_write_aux:nnn #1#2#3`
 2310 `{`
 2311 `__pdf_backend_pdfmark:e`
 2312 `{`
 2313 `/_objdef ~ #1`

```

2314         /type
2315         \str_case:nn {#2}
2316         {
2317             { array }    { /array }
2318             { dict }     { /dict }
2319             { fstream }  { /stream }
2320             { stream }   { /stream }
2321         }
2322     /OBJ
2323 }
2324 \use:c { __pdf_backend_object_write_ #2 :nn } {#1} {#3}
2325 }
2326 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2327 {
2328     \__pdf_backend_pdfmark:e
2329     { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2330 }
2331 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2332 {
2333     \__pdf_backend_pdfmark:e
2334     { #1 << \exp_not:n {#2} >> /PUT }
2335 }
2336 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2337 {
2338     \exp_args:Ne
2339     \__pdf_backend_object_write_fstream:nnn {#1} #2
2340 }
2341 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2342 {
2343     \__kernel_backend_postscript:n
2344     {
2345         SDict ~ begin ~
2346         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2347         mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2348         end
2349     }
2350 }
2351 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2352 {
2353     \exp_args:Ne
2354     \__pdf_backend_object_write_stream:nnn {#1} #2
2355 }
2356 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2357 {
2358     \__kernel_backend_postscript:n
2359     {
2360         mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2361         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2362     }
2363 }

```

(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

```

2364 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2365 {
2366   \int_gincr:N \g__pdf_backend_object_int
2367   \__pdf_backend_object_write_aux:nnn
2368   { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2369   {#1} {#2}
2370 }
2371 \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.

```

2372 \cs_new:Npn \__pdf_backend_object_last:
2373 { { 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.

```

2374 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2375 { { Page #1 } }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.1.3 Destinations

__pdf_backend_destination:nn
 __pdf_backend_destination:nnnn
 __pdf_backend_destination_aux:nnnn

Here, we need to turn the zoom into a scale. 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.

```

2376 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2377 {
2378   \__kernel_backend_postscript:n { pdf.dest.anchor }
2379   \__pdf_backend_pdfmark:e
2380   {
2381     /View
2382     [
2383       \str_case:nnF {#2}
2384       {
2385         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2386         { fit } { /Fit }
2387         { fitb } { /FitB }
2388         { fitbh } { /FitBH ~ pdf.dest.y }
2389         { fitbv } { /FitBV ~ pdf.dest.x }
2390         { fith } { /FitH ~ pdf.dest.y }
2391         { fitv } { /FitV ~ pdf.dest.x }
2392         { fitr } { /Fit }
2393       }
2394       {
2395         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2396       }
2397     ]
2398     /Dest ( \exp_not:n {#1} ) cvn
2399     /DEST
2400   }

```

```

2401 }
2402 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2403 {
2404   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2405   { \dim_eval:n {#2} } {#1} {#3} {#4}
2406 }
2407 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2408 {
2409   \vbox_to_zero:n
2410   {
2411     \dim_vertical:n {#4}
2412     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2413     \tex_vss:D
2414   }
2415   \dim_horizontal:n {#1}
2416   \vbox_to_zero:n
2417   {
2418     \dim_vertical:n { -#3 }
2419     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2420     \tex_vss:D
2421   }
2422   \dim_horizontal:n { -#1 }
2423   \__pdf_backend_pdfmark:n
2424   {
2425     /View
2426     [
2427       /FitR ~
2428       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2429       pdf.urx ~ pdf.ury ~ pdf.dest2device
2430     ]
2431     /Dest ( #2 ) cvn
2432     /DEST
2433   }
2434 }

```

(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

Doable for the usual ps2pdf method.

```

2435 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2436 {
2437   \int_compare:nNnT {#1} = 0
2438   {
2439     \__kernel_backend_literal_postscript:n
2440     {
2441       /setdistillerparams ~ where
2442       { pop << /CompressPages ~ false >> setdistillerparams }
2443       if
2444     }
2445   }
2446 }
2447 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1

```

```

2448 {
2449   \bool_if:nF {#1}
2450   {
2451     \__kernel_backend_literal_postscript:n
2452     {
2453       /setdistillerparams ~ where
2454       { pop << /CompressStreams ~ false >> setdistillerparams }
2455       if
2456     }
2457   }
2458 }

```

(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

```

2459 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2460 {
2461   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2462 }
2463 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2464 {
2465   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2466 }

```

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

__pdf_backend_version_major:

Data not available!

__pdf_backend_version_minor:

```

2467 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2468 \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

Simple wrappers.

__pdf_backend_emc:

```

2469 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2470 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2471 \cs_new_protected:Npn \__pdf_backend_emc:
2472 { \__pdf_backend_pdfmark:n { /EMC } }

```

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

2473 </dvips>

6.2 LuaTeX and pdfTeX backend

2474 < *luatex | pdftex >

6.2.1 Destinations

__pdf_backend_destination:nn

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

__pdf_backend_destination:nnnn

```

2475 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2476 {

```



```

2477 <*luatex>
2478   \tex_pdfextension:D dest ~
2479 </luatex>
2480 <*pdftex>
2481   \tex_pdfdest:D
2482 </pdftex>
2483   name {#1}
2484   \str_case:nnF {#2}
2485   {
2486     { xyz } { xyz }
2487     { fit } { fit }
2488     { fitb } { fitb }
2489     { fitbh } { fitbh }
2490     { fitbv } { fitbv }
2491     { fith } { fith }
2492     { fitv } { fitv }
2493     { fitr } { fitr }
2494   }
2495   { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2496   \scan_stop:
2497 }
2498 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2499 {
2500 <*luatex>
2501   \tex_pdfextension:D dest ~
2502 </luatex>
2503 <*pdftex>
2504   \tex_pdfdest:D
2505 </pdftex>
2506   name {#1}
2507   fitr ~
2508   width \dim_eval:n {#2} ~
2509   height \dim_eval:n {#3} ~
2510   depth \dim_eval:n {#4} \scan_stop:
2511 }

```

(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
2512 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2513 {
2514 <*luatex>
2515   \tex_pdfextension:D catalog
2516 </luatex>
2517 <*pdftex>
2518   \tex_pdfcatalog:D
2519 </pdftex>
2520   { / #1 ~ #2 }
2521 }
2522 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2523 {
2524 <*luatex>

```

```

2525     \tex_pdfextension:D info
2526 </luatex>
2527 <*pdftex>
2528     \tex_pdfinfo:D
2529 </pdftex>
2530     { / #1 ~ #2 }
2531 }

```

(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.

```

2532 \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 2533 \cs_new_protected:Npn \_pdf_backend_object_new:
\_pdf_backend_object_id:n 2534 {
2535 <*luatex>
2536     \tex_pdfextension:D obj ~
2537 </luatex>
2538 <*pdftex>
2539     \tex_pdfobj:D
2540 </pdftex>
2541     reserveobjnum ~
2542     \int_gset:Nn \g__pdf_backend_object_int
2543 <*luatex>
2544     { \tex_pdffeedback:D lastobj }
2545 </luatex>
2546 <*pdftex>
2547     { \tex_pdflastobj:D }
2548 </pdftex>
2549 }
2550 \cs_new:Npn \_pdf_backend_object_ref:n #1 { #1 ~ 0 ~ R }
2551 \cs_new:Npn \_pdf_backend_object_id:n #1 {#1}

```

(End of definition for _pdf_backend_object_new:, _pdf_backend_object_ref:n, and _pdf_backend_object_id:n.)

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

```

\_pdf_backend_object_write:nne 2552 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
\_pdf_backend_object_write:nn 2553 {
    \_pdf_exp_not_i:nn 2554 <*luatex>
    \_pdf_exp_not_ii:nn 2555     \tex_immediate:D \tex_pdfextension:D obj ~
2556 </luatex>
2557 <*pdftex>
2558     \tex_immediate:D \tex_pdfobj:D
2559 </pdftex>
2560     useobjnum ~ #1
2561     \_pdf_backend_object_write:nn {#2} {#3}
2562 }
2563 \cs_new:Npn \_pdf_backend_object_write:nn #1#2
2564 {

```

```

2565 \str_case:nn {#1}
2566 {
2567   { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2568   { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2569   { fstream }
2570   {
2571     stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2572     file ~ { \__pdf_exp_not_ii:nn #2 }
2573   }
2574   { stream }
2575   {
2576     stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2577     { \__pdf_exp_not_ii:nn #2 }
2578   }
2579 }
2580 }
2581 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2582 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2583 \cs_new:Npn \__pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

```

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

__pdf_backend_object_now:nn Much like writing, but direct creation.

```

\__pdf_backend_object_now:ne
2584 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2585 {
2586   <*luatex>
2587   \tex_immediate:D \tex_pdfextension:D obj ~
2588   </luatex>
2589   <*pdftex>
2590   \tex_immediate:D \tex_pdfobj:D
2591   </pdftex>
2592   \__pdf_backend_object_write:nn {#1} {#2}
2593 }
2594 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last: Much like annotation.

```

2595 \cs_new:Npe \__pdf_backend_object_last:
2596 {
2597   \exp_not:N \int_value:w
2598   <*luatex>
2599   \exp_not:N \tex_pdffeedback:D lastobj ~
2600   </luatex>
2601   <*pdftex>
2602   \exp_not:N \tex_pdflastobj:D
2603   </pdftex>
2604   \c_space_tl 0 ~ R
2605 }

```

(End of definition for __pdf_backend_object_last:.)

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

```

2606 \cs_new:Npe \__pdf_backend_pageobject_ref:n #1
2607 {

```

```

2608     \exp_not:N \int_value:w
2609 <*\luatex>
2610     \exp_not:N \tex_pdffeedback:D pageref
2611 </\luatex>
2612 <*\pdfTeX>
2613     \exp_not:N \tex_pdfpageref:D
2614 </\pdfTeX>
2615     \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2616 }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.2.4 Structure

Simply pass data to the engine.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
\__pdf_backend_objcompresslevel:n
2617 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2618 {
2619     \tex_global:D
2620 <*\luatex>
2621     \tex_pdfvariable:D compresslevel
2622 </\luatex>
2623 <*\pdfTeX>
2624     \tex_pdfcompresslevel:D
2625 </\pdfTeX>
2626     \int_value:w \int_eval:n {#1} \scan_stop:
2627 }
2628 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2629 {
2630     \bool_if:nTF {#1}
2631     { \__pdf_backend_objcompresslevel:n { 2 } }
2632     { \__pdf_backend_objcompresslevel:n { 0 } }
2633 }
2634 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2635 {
2636     \tex_global:D
2637 <*\luatex>
2638     \tex_pdfvariable:D objcompresslevel
2639 </\luatex>
2640 <*\pdfTeX>
2641     \tex_pdfobjcompresslevel:D
2642 </\pdfTeX>
2643     #1 \scan_stop:
2644 }

```

(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
2645 \cs_new_protected:Npe \__pdf_backend_version_major_gset:n #1
2646 {
2647 <*\luatex>
2648     \int_compare:nNnT \tex_luatexversion:D > { 106 }
2649     {
2650         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2651         \exp_not:N \int_eval:n {#1} \scan_stop:

```

```

2652     }
2653 </luatex>
2654 <*pdftex>
2655     \cs_if_exist:NT \tex_pdfmajorversion:D
2656     {
2657         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2658         \exp_not:N \int_eval:n {#1} \scan_stop:
2659     }
2660 </pdftex>
2661 }
2662 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2663 {
2664     \tex_global:D
2665 <*luatex>
2666     \tex_pdfvariable:D minorversion
2667 </luatex>
2668 <*pdftex>
2669     \tex_pdfminorversion:D
2670 </pdftex>
2671     \int_eval:n {#1} \scan_stop:
2672 }

```

(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:
2673 \cs_new:Npe \__pdf_backend_version_major:
2674 {
2675 <*luatex>
2676     \int_compare:nNnTF \tex luatexversion:D > { 106 }
2677     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2678     { 1 }
2679 </luatex>
2680 <*pdftex>
2681     \cs_if_exist:NTF \tex_pdfmajorversion:D
2682     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2683     { 1 }
2684 </pdftex>
2685 }
2686 \cs_new:Npn \__pdf_backend_version_minor:
2687 {
2688     \tex_the:D
2689 <*luatex>
2690     \tex_pdfvariable:D minorversion
2691 </luatex>
2692 <*pdftex>
2693     \tex_pdfminorversion:D
2694 </pdftex>
2695 }

```

(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>.
`__pdf_backend_emc:`

```
2696 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2697 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2698 \cs_new_protected:Npn \__pdf_backend_emc:
2699 { \__kernel_backend_literal_page:n { EMC } }

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

2700 </!latex | pdftex>
```

6.3 dvipdfmx backend

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

```
2701 <*dvipdfmx | xetex>

2702 \cs_new_protected:Npe \__pdf_backend:n #1
2703 { \__kernel_backend_literal:n { pdf: #1 } }
2704 \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`

```
2705 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2706 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2707 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2708 { \__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.

```
2709 \prop_new:N \g__pdf_backend_object_prop

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

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

```
2710 \cs_new_protected:Npn \__pdf_backend_object_new:
2711 { \int_gincr:N \g__pdf_backend_object_int }
2712 \cs_new:Npn \__pdf_backend_object_ref:n #1 { @pdf.obj #1 }
2713 \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_array:nn
\__pdf_backend_object_write_dict:nn
\__pdf_backend_object_write_fstream:nn
\__pdf_backend_object_write_stream:nn
\__pdf_backend_object_write_stream:nnnn

2714 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2715 {
2716   \use:c { __pdf_backend_object_write_ #2 :nn }
2717   { \__pdf_backend_object_ref:n {#1} } {#3}
2718 }
2719 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2720 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2721 {
2722   \__pdf_backend:e
2723   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2724 }
2725 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2726 {
2727   \__pdf_backend:e
2728   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2729 }
2730 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2731 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2732 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2733 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2734 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2735 {
2736   \__pdf_backend:e
2737   {
2738     #1 stream ~ #2 ~
2739     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2740   }
2741 }

```

This is where we choose the actual type.

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

```

\__pdf_backend_object_now:nn
\__pdf_backend_object_now:ne

2742 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2743 {
2744   \int_gincr:N \g__pdf_backend_object_int
2745   \exp_args:Nne \use:c { __pdf_backend_object_write_ #1 :nn }
2746   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2747   {#2}
2748 }
2749 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2750 \cs_new:Npn \__pdf_backend_object_last:
2751 { @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_TTeX.

```

2752 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2753 { @page #1 }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.3.3 Destinations

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 $\text{T}_{\text{E}}\text{X}$ by using the backend data for `@xpos` and `@ypos`. `/FitR` without rule spec doesn't work, so it falls back to `/Fit` here.

```

2754 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2755 {
2756   \__pdf_backend:e
2757   {
2758     dest ~ ( \exp_not:n {#1} )
2759     [
2760       @thispage
2761       \str_case:nnF {#2}
2762       {
2763         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2764         { fit } { /Fit }
2765         { fitb } { /FitB }
2766         { fitbh } { /FitBH }
2767         { fitbv } { /FitBV ~ @xpos }
2768         { fith } { /FitH ~ @ypos }
2769         { fitv } { /FitV ~ @xpos }
2770         { fitr } { /Fit }
2771       }
2772       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2773     ]
2774   }
2775 }
2776 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2777 {
2778   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2779   { \dim_eval:n {#2} } {#1} {#3} {#4}
2780 }
2781 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2782 {
2783   \vbox_to_zero:n
2784   {
2785     \dim_vertical:n {#4}
2786     \hbox:n
2787     {
2788       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2789       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2790     }
2791     \tex_vss:D
2792   }
2793   \dim_horizontal:n {#1}
2794   \vbox_to_zero:n
2795   {
2796     \dim_vertical:n { -#3 }
2797     \hbox:n
2798     {
2799       \__pdf_backend:n
2800       {
2801         dest ~ (#2)

```



```

2802         [
2803         @thispage
2804         /FitR ~
2805         @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2806         @xpos ~ @ypos
2807         ]
2808     }
2809 }
2810 \tex_vss:D
2811 }
2812 \dim_horizontal:n { -#1 }
2813 }

```

(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.

```

2814 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2815 { \__kernel_backend_literal:e { dvipdfmx:config~z~ \int_eval:n {#1} } }
2816 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2817 {
2818   \bool_if:nF {#1}
2819   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2820 }

```

(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.

```

2821 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2822 {
2823   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2824   \__kernel_backend_literal:e { pdf:majorversion~ \__pdf_backend_version_major: }
2825 }
2826 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2827 {
2828   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2829   \__kernel_backend_literal:e { pdf:minorversion~ \__pdf_backend_version_minor: }
2830 }

```

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

`__pdf_backend_version_major:`
`__pdf_backend_version_minor:`

We start with the assumption that the default is active.

```

2831 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2832 \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:`

```
2833 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2834 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2835 \cs_new_protected:Npn \__pdf_backend_emc:
2836 { \__kernel_backend_literal_page:n { EMC } }
```

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

```
2837 </dviptfm | xetex>
```

6.4 dvisvgm backend

```
2838 <*dvisvgm>
```

6.4.1 Destinations

```
\__pdf_backend_destination:nn
\__pdf_backend_destination:nnnn
2839 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2 { }
2840 \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
\__pdf_backend_info_gput:nn
2841 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2 { }
2842 \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:
\__pdf_backend_object_ref:n
\__pdf_backend_object_id:n
\__pdf_backend_object_write:nnn
\__pdf_backend_object_write:ne
\__pdf_backend_object_now:nn
\__pdf_backend_object_now:ne
\__pdf_backend_object_last:
\__pdf_backend_pageobject_ref:n
2843 \cs_new_protected:Npn \__pdf_backend_object_new: { }
2844 \cs_new:Npn \__pdf_backend_object_ref:n #1 { }
2845 \cs_new:Npn \__pdf_backend_object_id:n #1 { }
2846 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3 { }
2847 \cs_new_protected:Npn \__pdf_backend_object_write:ne #1#2#3 { }
2848 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2 { }
2849 \cs_new_protected:Npn \__pdf_backend_object_now:ne #1#2 { }
2850 \cs_new:Npn \__pdf_backend_object_last: { }
2851 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1 { }
```

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

6.4.4 Structure

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2852 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2853 \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
2854 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2855 \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:
2856 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2857 \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:
2858 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2859 \cs_new_protected:Npn \__pdf_backend_emc: { }

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

2860 </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.

```

2861 <*dvipdfmx | dvips>

\__pdf_backend_pagesize_gset:nn
2862 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2
2863 {
2864   \__kernel_backend_first_shipout:n
2865   {
2866     \__kernel_backend_literal:e
2867     {
2868       <*dvipdfmx>
2869         pdf:pagesize ~
2870         width ~ \dim_eval:n {#1} ~
2871         height ~ \dim_eval:n {#2}
2872       </dvipdfmx>
2873       <*dvips>
2874         papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
2875       </dvips>
2876     }
2877   }
2878 }

```

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

2879 `</dvipdfmx | dvips>`

2880 `<*luatex | pdftex | xetex>`

`__pdf_backend_pagesize_gset:nn` Pass to the primitives.

2881 `\cs_new_protected:Npn __pdf_backend_pagesize_gset:nn #1#2`

2882 `{`

2883 `\dim_gset:Nn \tex_pagewidth:D {#1}`

2884 `\dim_gset:Nn \tex_pageheight:D {#2}`

2885 `}`

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

2886 `</luatex | pdftex | xetex>`

2887 `<*dvisvgm>`

`__pdf_backend_pagesize_gset:nn` A no-op.

2888 `\cs_new_protected:Npn __pdf_backend_pagesize_gset:nn #1#2 { }`

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

2889 `</dvisvgm>`

2890 `</package>`

7 l3backend-pdfannot implementation

2891 `<*package>`

2892 `<@@=pdfannot>`

7.1 dvips backend

2893 `<*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.

2894 `\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.

2895 `\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.

2896 `\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.

```

2897 \cs_new_protected:Npn \__pdfannot_backend_generic:nnnn #1#2#3#4
2898 {
2899   \exp_args:Nf \__pdfannot_backend_generic_aux:nnnn
2900   { \dim_eval:n {#1} } {#2} {#3} {#4}
2901 }
2902 \cs_new_protected:Npn \__pdfannot_backend_generic_aux:nnnn #1#2#3#4
2903 {
2904   \box_move_down:nn {#3}
2905   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2906   \box_move_up:nn {#2}
2907   {
2908     \hbox:n
2909     {
2910       \dim_horizontal:n {#1}
2911       \__kernel_backend_postscript:n { pdf.save.ur }
2912       \dim_horizontal:n { -#1 }
2913     }
2914   }
2915   \int_gincr:N \g__pdfannot_backend_int
2916   \__kernel_backend_postscript:e
2917   {
2918     mark
2919     /_objdef { pdf.annot \int_use:N \g__pdfannot_backend_int }
2920     pdf.rect
2921     #4 ~
2922     /ANN ~
2923     pdfmark
2924   }
2925 }
```

(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.

```

2926 \cs_new:Npn \__pdfannot_backend_last:
2927 { { 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.

```

2928 \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.

```

2929 \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.

```

2930 \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.

2931 `\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.

2932 `\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.

2933 `\tl_new:N \l__pdfannot_backend_breaklink_pdfmark_tl`

2934 `\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.

2935 `\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.

2936 `\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 `hydpvips`, 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 `hydpvips` 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 `hydpvips`.

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.

2937 `\cs_new_protected:Npn _pdfannot_backend_link_begin_goto:nnw #1#2`

2938 `{`

2939 `_pdfannot_backend_link_begin:nw`

2940 `{ #1 /Subtype /Link /Action << /S /GoTo /D (#2) >> }`

2941 `}`

2942 `\cs_new_protected:Npn _pdfannot_backend_link_begin_user:nnw #1#2`

2943 `{ _pdfannot_backend_link_begin:nw {#1#2} }`

2944 `\cs_new_protected:Npn _pdfannot_backend_link_begin:nw #1`

2945 `{`

```

2946 \bool_if:NF \g__pdfannot_backend_link_bool
2947 { \__pdfannot_backend_link_begin_aux:nw {#1} }
2948 }

```

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.

```

2949 \cs_new_protected:Npn \__pdfannot_backend_link_begin_aux:nw #1
2950 {
2951   \bool_gset_true:N \g__pdfannot_backend_link_bool
2952   \__kernel_backend_postscript:n
2953   { /pdf.link.dict ( #1 ) def }
2954   \tl_gset:Nn \g__pdfannot_backend_link_dict_tl {#1}
2955   \__pdfannot_backend_link_sf_save:
2956   \mode_if_math:TF
2957   { \bool_gset_true:N \g__pdfannot_backend_link_math_bool }
2958   { \bool_gset_false:N \g__pdfannot_backend_link_math_bool }
2959   \hbox_set:Nw \l__pdfannot_backend_content_box
2960   \__pdfannot_backend_link_sf_restore:
2961   \bool_if:NT \g__pdfannot_backend_link_math_bool
2962   { \c_math_toggle_token }
2963 }
2964 \cs_new_protected:Npn \__pdfannot_backend_link_end:
2965 {
2966   \bool_if:NT \g__pdfannot_backend_link_bool
2967   { \__pdfannot_backend_link_end_aux: }
2968 }
2969 \cs_new_protected:Npn \__pdfannot_backend_link_end_aux:
2970 {
2971   \bool_if:NT \g__pdfannot_backend_link_math_bool
2972   { \c_math_toggle_token }
2973   \__pdfannot_backend_link_sf_save:
2974   \hbox_set_end:
2975   \__pdfannot_backend_link_minima:
2976   \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
2977   \exp_args:Ne \__pdfannot_backend_link_outerbox:n
2978   {
2979     \int_if_odd:nTF { \value { page } }
2980     { \oddsidemargin }
2981     { \evensidemargin }
2982   }
2983   \box_move_down:nn { \box_dp:N \l__pdfannot_backend_content_box }
2984   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2985   \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.begin }
2986   \__pdfannot_backend_breaklink_usebox:N \l__pdfannot_backend_content_box
2987   \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.end }
2988   \box_move_up:nn { \box_ht:N \l__pdfannot_backend_content_box }
2989   {
2990     \hbox:n
2991     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2992   }
2993   \int_gincr:N \g__pdfannot_backend_int
2994   \int_gset_eq:NN \g__pdfannot_backend_link_int \g__pdfannot_backend_int
2995   \__kernel_backend_postscript:e
2996   {

```

```

2997         mark
2998         /_objdef { pdf.annot \int_use:N \g__pdfannot_backend_link_int }
2999         \g__pdfannot_backend_link_dict_tl \c_space_tl
3000         pdf.rect
3001         /ANN ~ \l__pdfannot_backend_breaklink_pdfmark_tl
3002     }
3003     \__pdfannot_backend_link_sf_restore:
3004     \bool_gset_false:N \g__pdfannot_backend_link_bool
3005 }
3006 \cs_new_protected:Npn \__pdfannot_backend_link_minima:
3007 {
3008     \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
3009     \__kernel_backend_postscript:e
3010     {
3011         /pdf.linkdp.pad ~
3012         \dim_to_decimal:n
3013         {
3014             \dim_max:nn
3015             {
3016                 \box_dp:N \l__pdfannot_backend_model_box
3017                 - \box_dp:N \l__pdfannot_backend_content_box
3018             }
3019             { Opt }
3020         } ~
3021         pdf.pt.dvi ~ def
3022     /pdf.linkht.pad ~
3023     \dim_to_decimal:n
3024     {
3025         \dim_max:nn
3026         {
3027             \box_ht:N \l__pdfannot_backend_model_box
3028             - \box_ht:N \l__pdfannot_backend_content_box
3029         }
3030         { Opt }
3031     } ~
3032     pdf.pt.dvi ~ def
3033 }
3034 }
3035 \cs_new_protected:Npn \__pdfannot_backend_link_outerbox:n #1
3036 {
3037     \__kernel_backend_postscript:e
3038     {
3039         /pdf.outerbox
3040         [
3041             \dim_to_decimal:n {#1} ~
3042             \dim_to_decimal:n { -\box_dp:N \l__pdfannot_backend_model_box } ~
3043             \dim_to_decimal:n { #1 + \textwidth } ~
3044             \dim_to_decimal:n { \box_ht:N \l__pdfannot_backend_model_box }
3045         ]
3046         [ exch { pdf.pt.dvi } forall ] def
3047     /pdf.baselineskip ~
3048     \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
3049     { pdf.pt.dvi ~ def }
3050     { pop ~ pop }

```



```

3051         ifelse
3052     }
3053 }
3054 \cs_new_protected:Npn \__pdfannot_backend_link_sf_save:
3055 {
3056     \int_gset:Nn \g__pdfannot_backend_link_sf_int
3057     {
3058         \mode_if_horizontal:TF
3059         { \tex_spacefactor:D }
3060         { 0 }
3061     }
3062 }
3063 \cs_new_protected:Npn \__pdfannot_backend_link_sf_restore:
3064 {
3065     \mode_if_horizontal:T
3066     {
3067         \int_compare:nNnT \g__pdfannot_backend_link_sf_int > { 0 }
3068         { \int_set:Nn \tex_spacefactor:D \g__pdfannot_backend_link_sf_int }
3069     }
3070 }

```

(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.

```

3071 \use_none:nnn
3072 \cs_if_exist:NT \hook_gput_code:nnn
3073 {
3074     \hook_gput_code:nnn { build/column/after } { backend }
3075     {
3076         \box_if_empty:NF \l_shipout_box
3077         {
3078             \vbox_set:Nn \l_shipout_box
3079             {
3080                 \__kernel_backend_postscript:n
3081                 {
3082                     pdf.globaldict /pdf.brokenlink.rect ~ known
3083                     { pdf.bordertracking.continue }
3084                     if
3085                 }
3086                 \vbox_unpack_drop:N \l_shipout_box
3087                 \__kernel_backend_postscript:n
3088                 { pdf.bordertracking.endpage }
3089             }
3090         }
3091     }
3092     \tl_set:Nn \l__pdfannot_backend_breaklink_pdfmark_tl { pdf.pdfmark }
3093     \cs_set_eq:NN \__pdfannot_backend_breaklink_postscript:n
3094     \__kernel_backend_postscript:n
3095     \cs_set_eq:NN \__pdfannot_backend_breaklink_usebox:N \hbox_unpack:N
3096 }

```

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

```

3097 \cs_new:Npn \__pdfannot_backend_link_last:
3098 { { 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.

```

3099 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1
3100 {
3101   \__kernel_backend_postscript:e
3102   {
3103     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
3104   }
3105 }
```

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

`__pdfannot_backend_link_on:`

```

\__pdfannot_backend_link_off: 3106 \cs_new_protected:Npn \__pdfannot_backend_link_on: { }
3107 \cs_new_protected:Npn \__pdfannot_backend_link_off: { }
```

(End of definition for `__pdfannot_backend_link_on:` and `__pdfannot_backend_link_off:`.)

```

3108 </dvips>
```

7.2 LuaTeX and pdfTeX backend

```

3109 <*luatex | pdftex>
```

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

```

3110 \cs_new_protected:Npn \__pdfannot_backend_generic:nnnn #1#2#3#4
3111 {
3112   <*luatex>
3113   \tex_pdfextension:D annot ~
3114   </luatex>
3115   <*pdftex>
3116   \tex_pdfannot:D
3117   </pdftex>
3118   width ~ \dim_eval:n {#1} ~
3119   height ~ \dim_eval:n {#2} ~
3120   depth ~ \dim_eval:n {#3} ~
3121   {#4}
3122 }
```

(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.

```

3123 \cs_new:Npe \__pdfannot_backend_last:
3124 {
3125   \exp_not:N \int_value:w
3126   <*luatex>
3127   \exp_not:N \tex_pdffeedback:D lastannot ~
3128   </luatex>
3129   <*pdftex>
3130   \exp_not:N \tex_pdflastannot:D
3131   </pdftex>
3132   \c_space_tl 0 ~ R
3133 }
```

(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.

```

3134 \cs_new_protected:Npn \__pdfannot_backend_link_begin_goto:nnw #1#2
3135 { \__pdfannot_backend_link_begin:nnnw {#1} { goto~name } {#2} }
3136 \cs_new_protected:Npn \__pdfannot_backend_link_begin_user:nnw #1#2
3137 { \__pdfannot_backend_link_begin:nnnw {#1} { user } {#2} }
3138 \cs_new_protected:Npn \__pdfannot_backend_link_begin:nnnw #1#2#3
3139 {
3140   \luatex
3141     \tex_pdfextension:D startlink ~
3142   \luatex
3143   \pdfTeX
3144     \tex_pdfstartlink:D
3145   \pdfTeX
3146     attr {#1}
3147     #2 {#3}
3148   }
3149 \cs_new_protected:Npn \__pdfannot_backend_link_end:
3150 {
3151   \luatex
3152     \tex_pdfextension:D endlink \scan_stop:
3153   \luatex
3154   \pdfTeX
3155     \tex_pdfendlink:D
3156   \pdfTeX
3157   }

```

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

`__pdfannot_backend_link_last:`

Formatted for direct use.

```

3158 \cs_new:Npe \__pdfannot_backend_link_last:
3159 {
3160   \exp_not:N \int_value:w
3161   \luatex
3162     \exp_not:N \tex_pdffeedback:D lastlink ~
3163   \luatex
3164   \pdfTeX
3165     \exp_not:N \tex_pdflastlink:D
3166   \pdfTeX
3167     \c_space_tl 0 ~ R
3168   }

```

(End of definition for `__pdfannot_backend_link_last:`)

`__pdfannot_backend_link_margin:n`

A simple task: pass the data to the primitive.

```

3169 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1
3170 {
3171   \luatex
3172     \tex_pdfvariable:D linkmargin
3173   \luatex
3174   \pdfTeX
3175     \tex_pdflinkmargin:D
3176   \pdfTeX

```

```

3177     \dim_eval:n {#1} \scan_stop:
3178   }

(End of definition for \__pdfannot_backend_link_margin:n.)

```

__pdfannot_backend_link_on: Separate definitions for the two engines.

```

\__pdfannot_backend_link_off: 3179 \cs_new_protected:Npn \__pdfannot_backend_link_on:
3180   < *luatex >
3181   { \tex_pdfextension:D linkstate 0 ~ }
3182 < /luatex >
3183 < *pdftex >
3184   { \tex_pdfrunninglinkon:D }
3185 < /pdftex >
3186 \cs_new_protected:Npn \__pdfannot_backend_link_off:
3187   < *luatex >
3188   { \tex_pdfextension:D linkstate 1 ~ }
3189 < /luatex >
3190 < *pdftex >
3191   { \tex_pdfrunninglinkoff:D }
3192 < /pdftex >

(End of definition for \__pdfannot_backend_link_on: and \__pdfannot_backend_link_off:.)
3193 < /luatex | pdftex >

```

7.3 dvipdfmx backend

```

3194 < *dvipdfmx | xetex >

\__pdfannot_backend:n A generic function for the backend PDF specials
\__pdfannot_backend:e 3195 \cs_new_protected:Npe \__pdfannot_backend:n #1
3196   { \__kernel_backend_literal:n { pdf: #1 } }
3197 \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.
3198 \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.
3199 \cs_new_protected:Npn \__pdfannot_backend_generic:nnnn #1#2#3#4
3200   {
3201     \int_gincr:N \g__pdfannot_backend_int
3202     \__pdfannot_backend:e
3203     {
3204       ann ~ @pdfannot \int_use:N \g__pdfannot_backend_int \c_space_tl
3205       width ~ \dim_eval:n {#1} ~
3206       height ~ \dim_eval:n {#2} ~
3207       depth ~ \dim_eval:n {#3} ~
3208       << /Type /Annot #4 >>
3209     }
3210   }

(End of definition for \__pdfannot_backend_generic:nnnn.)

```

_pdfannot_backend_last:

```
3211 \cs_new:Npn \_pdfannot_backend_last:
3212 { @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.

```
3213 \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

```
3214 \cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nnw #1#2
```

_pdfannot_backend_link_begin:n

```
3215 {
```

_pdfannot_backend_link_end:

```
3216 \_pdfannot_backend_link_begin:n
```

```
3217 { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> }
```

```
3218 }
```

```
3219 \cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nnw #1#2
```

```
3220 { \_pdfannot_backend_link_begin:n {#1#2} }
```

```
3221 \cs_new_protected:Npe \_pdfannot_backend_link_begin:n #1
```

```
3222 {
```

```
3223 \int_gincr:N \exp_not:N \g__pdfannot_backend_int
```

```
3224 \int_gset_eq:NN \exp_not:N \g__pdfannot_backend_link_int
```

```
3225 \exp_not:N \g__pdfannot_backend_int
```

```
3226 \_pdfannot_backend:e
```

```
3227 {
```

```
3228 bann ~
```

```
3229 @pdfannot
```

```
3230 \exp_not:N \int_use:N \exp_not:N \g__pdfannot_backend_link_int
```

```
3231 \c_space_tl
```

```
3232 <<
```

```
3233 /Type /Annot
```

```
3234 #1
```

```
3235 >>
```

```
3236 }
```

```
3237 }
```

```
3238 \cs_new_protected:Npn \_pdfannot_backend_link_end:
```

```
3239 { \_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.

```
3240 \cs_new:Npn \_pdfannot_backend_link_last:
```

```
3241 { @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.

```
3242 \cs_new_protected:Npn \_pdfannot_backend_link_margin:n #1
```

```
3243 { \_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:

```
3244 \cs_new_protected:Npn \_pdfannot_backend_link_on: { \_pdfannot_backend:n { link } }
```

```
3245 \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:.`)

3246 \langle /dvipdfmx | xetex \rangle

7.4 dvisvgm backend

3247 \langle *dvisvgm \rangle

`_pdfannot_backend_generic:nnnn`

3248 `\cs_new_protected:Npn _pdfannot_backend_generic:nnnn #1#2#3#4 { }`

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

`_pdfannot_backend_last:`

3249 `\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:`

3250 `\cs_new_protected:Npn _pdfannot_backend_link_begin_goto:nnw #1#2 { }`

3251 `\cs_new_protected:Npn _pdfannot_backend_link_begin_user:nnw #1#2 { }`

3252 `\cs_new_protected:Npn _pdfannot_backend_link_begin:nnnw #1#2#3 { }`

3253 `\cs_new_protected:Npn _pdfannot_backend_link_end: { }`

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

`_pdfannot_backend_link_last:`

3254 `\cs_new:Npe _pdfannot_backend_link_last: { }`

(End of definition for `_pdfannot_backend_link_last:.`)

`_pdfannot_backend_link_margin:n`

3255 `\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:`

3256 `\cs_new_protected:Npn _pdfannot_backend_link_on: { }`

3257 `\cs_new_protected:Npn _pdfannot_backend_link_off: { }`

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

3258 \langle /dvisvgm \rangle

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.

```

3259 <@@=)
3260 \cs_new_eq:NN \__pdf_backend_annotation:nnnn \__pdfannot_backend_generic:nnnn
3261 \cs_new_eq:NN \__pdf_backend_annotation_last: \__pdfannot_backend_last:
3262 \clist_map_inline:nn
3263 {
3264   begin_goto:nnw ,
3265   begin_user:nnw ,
3266   begin:nnnw ,
3267   end: ,
3268   last: ,
3269   margin:n
3270 }
3271 { \cs_new_eq:cc { __pdf_backend_link_ #1 } { __pdfannot_backend_link_ #1 } }
3272 </package>

```

8 l3backend-opacity implementation

```

3273 <*package>
3274 <@@=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.

```

3275 <*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.

```

3276 \cs_new_protected:Npn \__opacity_backend_select:n #1
3277 {
3278   \__opacity_backend:nnn {#1} { fill } { ca }
3279   \__opacity_backend:nnn {#1} { stroke } { CA }
3280 }
3281 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3282 {
3283   \__opacity_backend:nnn
3284     { #1 }
3285     { fill }
3286     { ca }
3287 }
3288 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3289 {
3290   \__opacity_backend:nnn
3291     { #1 }

```

```

3292     { stroke }
3293     { CA }
3294 }
3295 \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3296 {
3297     \__kernel_backend_postscript:n
3298     {
3299         product ~ (Ghostscript) ~ search
3300         {
3301             pop ~ pop ~ pop ~
3302             #1 ~ .set #2 constantalpha
3303         }
3304         {
3305             pop ~
3306             mark ~
3307             /#3 ~ #1
3308             /SetTransparency ~
3309             pdfmark
3310         }
3311         ifelse
3312     }
3313 }
3314 \cs_new_protected:Npn \__opacity_backend_reset:
3315 {
3316     \__opacity_backend_reset_fill:
3317     \__opacity_backend_reset_stroke:
3318 }
3319 \cs_new_protected:Npn \__opacity_backend_reset_fill:
3320 {
3321     \__opacity_backend:nnn
3322     { 1 }
3323     { fill }
3324     { ca }
3325 }
3326 \cs_new_protected:Npn \__opacity_backend_reset_stroke:
3327 {
3328     \__opacity_backend:nnn
3329     { 1 }
3330     { stroke }
3331     { CA }
3332 }

```

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

```
3333 </dvips>
```

```
3334 < *dvipdfmx | luatex | pdftex | xetex >
```

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

```

3335 \bool_lazy_and:nnT
3336 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3337 { \pdfmanagement_if_active_p: }
3338 {
3339 < *luatex | pdftex >
3340     \__kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int

```



```

3341         { page ~ direct } { /opacity 1 ~ gs }
3342 </luatex | pdftex>
3343     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3344     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3345 }

```

(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.

```

3346 \tl_new:N \l_opacity_backend_fill_tl
3347 \tl_new:N \l_opacity_backend_stroke_tl
3348 \tl_set:Nn \l_opacity_backend_fill_tl { 1 }
3349 \tl_set:Nn \l_opacity_backend_stroke_tl { 1 }

```

(End of definition for \l_opacity_backend_fill_tl and \l_opacity_backend_stroke_tl.)

_opacity_backend_select:n Much the same as color.

```

\_opacity_backend_reset: 3350 \cs_new_protected:Npn \_opacity_backend_select:n #1
\_opacity_backend_reset_fill: 3351 {
\_opacity_backend_reset_stroke: 3352     \tl_set:Nn \l_opacity_backend_fill_tl {#1}
3353     \tl_set:Nn \l_opacity_backend_stroke_tl {#1}
3354     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3355     { opacity #1 }
3356     { << /ca ~ #1 /CA ~ #1 >> }
3357 < *dvipdfmx | xetex >
3358     \_kernel_backend_literal_pdf:n
3359 < /dvipdfmx | xetex >
3360 < *luatex | pdftex >
3361     \_kernel_color_backend_stack_push:nn \c_opacity_backend_stack_int
3362 < /luatex | pdftex >
3363     { /opacity #1 ~ gs }
3364 }
3365 \cs_new_protected:Npn \_opacity_backend_reset:
3366 {
3367 < *dvipdfmx | xetex >
3368     \_kernel_backend_literal_pdf:n
3369     { /opacity 1 ~ gs }
3370 < /dvipdfmx | xetex >
3371 < *luatex | pdftex >
3372     \_kernel_color_backend_stack_pop:n \c_opacity_backend_stack_int
3373 < /luatex | pdftex >
3374 }
3375 \cs_new_eq:NN \_opacity_backend_reset_fill: \_opacity_backend_reset:
3376 \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.

```

\_opacity_backend_fill:n 3377 \cs_new_protected:Npn \_opacity_backend_fill:n #1
\_opacity_backend_fill_stroke:n 3378 {
3379     \exp_args:Nno \_opacity_backend_fill_stroke:nn
3380     { #1 }
3381     { \l_opacity_backend_stroke_tl }

```

```

3382 }
3383 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3384 {
3385   \exp_args:No \__opacity_backend_fill_stroke:nn
3386   { \l__opacity_backend_fill_tl }
3387   { #1 }
3388 }
3389 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3390 {
3391   \str_if_eq:nnTF {#1} {#2}
3392   { \__opacity_backend_select:n {#1} }
3393   {
3394     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3395     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3396     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3397     { opacity.fill #1 }
3398     { << /ca ~ #1 >> }
3399     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3400     { opacity.stroke #2 }
3401     { << /CA ~ #2 >> }
3402     <*/dvipdfmx|xetex>
3403     \__kernel_backend_literal_pdf:n
3404     </dvipdfmx|xetex>
3405     <*/luatex|pdfTeX>
3406     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3407     </luatex|pdfTeX>
3408     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3409   }
3410 }

```

(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.

```

3411 \bool_lazy_and:nnF
3412 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3413 { \pdfmanagement_if_active_p: }
3414 {
3415   \cs_gset_protected:Npn \__opacity_backend_select:n #1 { }
3416   \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2 { }
3417   \cs_gset_protected:Npn \__opacity_backend_reset: { }
3418   \cs_gset_eq:NN \__opacity_backend_reset_fill: \__opacity_backend_reset:
3419   \cs_gset_eq:NN \__opacity_backend_reset_stroke: \__opacity_backend_reset:
3420 }

```

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

```

3421 </dvipdfmx|luatex|pdfTeX|xetex>
3422 <*/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.

```

3423 \cs_new_protected:Npn \__opacity_backend_select:n #1
3424 { \__opacity_backend:nn {#1} { } }
3425 \cs_new_protected:Npn \__opacity_backend_fill:n #1

```

```

3426 { \_opacity_backend:nn {#1} { fill- } }
3427 \cs_new_protected:Npn \_opacity_backend_stroke:n #1
3428 { \_opacity_backend:nn {#1} { stroke- } }
3429 \cs_new_protected:Npn \_opacity_backend:nn #1#2
3430 { \_kernel_backend_scope:e { #2 opacity = " #1 " } }
3431 \cs_new_protected:Npn \_opacity_backend_reset: { }
3432 \cs_new_eq:NN \_opacity_backend_reset_fill: \_opacity_backend_reset:
3433 \cs_new_eq:NN \_opacity_backend_reset_stroke: \_opacity_backend_reset:

```

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

```

3434 </divisvgm>
3435 </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.

```

3436 <*lua>

```

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

```

3437 local pdfmanagement_active do
3438   local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3439   local cmd = pdfmanagement_if_active_p.cmdname
3440   if cmd == 'undefined_cs' then
3441     pdfmanagement_active = false
3442   else
3443     token.put_next(pdfmanagement_if_active_p)
3444     pdfmanagement_active = token.scan_int() ~= 0
3445   end
3446 end
3447
3448 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3449   luaotfload.set_transparent_colorstack(function() return token.create'c_opacity_backend_st
3450
3451   local transparent_register = {
3452     token.create'pdfmanagement_add:nnn',
3453     token.new(0, 1),
3454     'Page/Resources/ExtGState',
3455     token.new(0, 2),
3456     token.new(0, 1),
3457     '',
3458     token.new(0, 2),
3459     token.new(0, 1),
3460     '<</ca ',
3461     '',
3462     '/CA ',
3463     '',
3464     '>>',
3465     token.new(0, 2),
3466   }
3467   luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)
3468     value = (octet * -1):match(value)
3469     if not value then

```

```

3470         tex.error'Invalid transparency value'
3471     return
3472 end
3473 value = value:sub(1, -2)
3474 local result = 'opacity' .. value
3475 tex.runtoks(function()
3476     transparent_register[6], transparent_register[10], transparent_register[12] = result,
3477     tex.sprint(-2, transparent_register)
3478 end)
3479 return '/' .. result .. ' gs'
3480 end, 'l3opacity')
3481 end
3482  $\langle$ /lua $\rangle$ 

```

9 l3backend-header implementation

```

3483  $\langle$ *dvips & header $\rangle$ 

```

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

```

3484 /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.

```

3485 TeXDict begin
3486 /TeXcolorseparation { setcolor } def
3487 end

```

(End of definition for TeXcolorseparation and separation.)

pdf.globaldict A small global dictionary for backend use.

```

3488 true setglobal
3489 /pdf.globaldict 4 dict def
3490 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.

```

3491 /pdf.cvs { 65534 string cvs } def
3492 /pdf.dvi.pt { 72.27 mul Resolution div } def
3493 /pdf.pt.dvi { 72.27 div Resolution mul } def
3494 /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.

```

3495 /pdf.linkmargin { 1 pdf.pt.dvi } def
3496 /pdf.linkdp.pad { 0 } def
3497 /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	3498 /pdf.rect
pdf.save.linkur	3499 { /Rect [pdf.llx pdf.lly pdf.urx pdf.ury] } def
pdf.llx	3500 /pdf.save.ll
pdf.lly	3501 {
pdf.urx	3502 currentpoint
pdf.ury	3503 /pdf.lly exch def
	3504 /pdf.llx exch def
	3505 }
	3506 def
	3507 /pdf.save.ur
	3508 {
	3509 currentpoint
	3510 /pdf.ury exch def
	3511 /pdf.urx exch def
	3512 }
	3513 def
	3514 /pdf.save.linkll
	3515 {
	3516 currentpoint
	3517 pdf.linkmargin add
	3518 pdf.linkdp.pad add
	3519 /pdf.lly exch def
	3520 pdf.linkmargin sub
	3521 /pdf.llx exch def
	3522 }
	3523 def
	3524 /pdf.save.linkur
	3525 {
	3526 currentpoint
	3527 pdf.linkmargin sub
	3528 pdf.linkht.pad sub
	3529 /pdf.ury exch def
	3530 pdf.linkmargin add
	3531 /pdf.urx exch def
	3532 }
	3533 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	3534 /pdf.dest.anchor
pdf.dev.y	3535 {
pdf.tmpa	3536 currentpoint exch
pdf.tmpb	3537 pdf.dvi.pt 72 add
pdf.tmpc	3538 /pdf.dest.x exch def
pdf.tmpd	3539 pdf.dvi.pt
	3540 vsize 72 sub exch sub

```

3541     /pdf.dest.y exch def
3542   }
3543   def
3544 /pdf.dest.point
3545   { pdf.dest.x pdf.dest.y } def
3546 /pdf.dest2device
3547   {
3548     /pdf.dest.y exch def
3549     /pdf.dest.x exch def
3550     matrix currentmatrix
3551     matrix defaultmatrix
3552     matrix invertmatrix
3553     matrix concatmatrix
3554     cvx exec
3555     /pdf.dev.y exch def
3556     /pdf.dev.x exch def
3557     /pdf.tmpd exch def
3558     /pdf.tmpc exch def
3559     /pdf.tmpb exch def
3560     /pdf.tmpa exch def
3561     pdf.dest.x pdf.tmpa mul
3562     pdf.dest.y pdf.tmpc mul add
3563     pdf.dev.x add
3564     pdf.dest.x pdf.tmpb mul
3565     pdf.dest.y pdf.tmpd mul add
3566     pdf.dev.y add
3567   }
3568   def

```

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

```

pdf.bordertracking
pdf.bordertracking.begin
pdf.bordertracking.end
pdf.leftboundary
pdf.rightboundary
pdf.brokenlink.rect
pdf.brokenlink.skip
pdf.brokenlink.dict
pdf.bordertracking.endpage
pdf.bordertracking.continue
pdf.originx
pdf.originy

```

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

```

3569 /pdf.bordertracking false def
3570 /pdf.bordertracking.begin
3571   {
3572     SDict /pdf.bordertracking true put
3573     SDict /pdf.leftboundary undef
3574     SDict /pdf.rightboundary undef
3575     /a where
3576     {
3577       /a
3578       {
3579         currentpoint pop
3580         SDict /pdf.rightboundary known dup
3581         {
3582           SDict /pdf.rightboundary get 2 index lt
3583           { not }
3584           if
3585         }
3586         if
3587         { pop }

```

```

3588         { SDict exch /pdf.rightboundary exch put }
3589     ifelse
3590     moveto
3591     currentpoint pop
3592     SDict /pdf.leftboundary known dup
3593     {
3594         SDict /pdf.leftboundary get 2 index gt
3595         { not }
3596         if
3597     }
3598     if
3599     { pop }
3600     { SDict exch /pdf.leftboundary exch put }
3601     ifelse
3602 }
3603 put
3604 }
3605 if
3606 }
3607 def
3608 /pdf.bordertracking.end
3609 {
3610     /a where { /a { moveto } put } if
3611     /x where { /x { 0 exch rmoveto } put } if
3612     SDict /pdf.leftboundary known
3613     { pdf.outerbox 0 pdf.leftboundary put }
3614     if
3615     SDict /pdf.rightboundary known
3616     { pdf.outerbox 2 pdf.rightboundary put }
3617     if
3618     SDict /pdf.bordertracking false put
3619 }
3620 def
3621 /pdf.bordertracking.endpage
3622 {
3623     pdf.bordertracking
3624     {
3625         pdf.bordertracking.end
3626         true setglobal
3627         pdf.globaldict
3628         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3629         pdf.globaldict
3630         /pdf.brokenlink.skip pdf.baselineskip put
3631         pdf.globaldict
3632         /pdf.brokenlink.dict
3633         pdf.link.dict pdf.cvs put
3634         false setglobal
3635         mark pdf.link.dict cvx exec /Rect
3636         [
3637             pdf.llx
3638             pdf.lly
3639             pdf.outerbox 2 get pdf.linkmargin add
3640             currentpoint exch pop
3641             pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub

```

```

3642     ]
3643     /ANN pdf.pdfmark
3644   }
3645   if
3646 }
3647   def
3648 /pdf.bordertracking.continue
3649   {
3650     /pdf.link.dict pdf.globaldict
3651     /pdf.brokenlink.dict get def
3652     /pdf.outerbox pdf.globaldict
3653     /pdf.brokenlink.rect get def
3654     /pdf.baselineskip pdf.globaldict
3655     /pdf.brokenlink.skip get def
3656     pdf.globaldict dup dup
3657     /pdf.brokenlink.dict undef
3658     /pdf.brokenlink.skip undef
3659     /pdf.brokenlink.rect undef
3660     currentpoint
3661     /pdf.originy exch def
3662     /pdf.originx exch def
3663     /a where
3664     {
3665       /a
3666       {
3667         moveto
3668         SDict
3669         begin
3670         currentpoint pdf.originy ne exch
3671         pdf.originx ne or
3672         {
3673           pdf.save.linkll
3674           /pdf.lly
3675           pdf.lly pdf.outerbox 1 get sub def
3676           pdf.bordertracking.begin
3677         }
3678         if
3679         end
3680       }
3681       put
3682     }
3683   if
3684   /x where
3685   {
3686     /x
3687     {
3688       0 exch rmoveto
3689       SDict
3690       begin
3691       currentpoint
3692       pdf.originy ne exch pdf.originx ne or
3693       {
3694         pdf.save.linkll
3695         /pdf.lly

```



```

3696         pdf.lly pdf.outerbox 1 get sub def
3697         pdf.bordertracking.begin
3698     }
3699     if
3700     end
3701 }
3702 put
3703 }
3704 if
3705 }
3706 def

```

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

```

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

```

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

```

3707 /pdf.breaklink
3708 {
3709     pop
3710     counttomark 2 mod 0 eq
3711     {
3712         counttomark /pdf.count exch def
3713         {
3714             pdf.count 0 eq { exit } if
3715             counttomark 2 roll
3716             1 index /Rect eq
3717             {
3718                 dup 4 array copy
3719                 dup dup
3720                 1 get
3721                 pdf.outerbox pdf.rect.ht
3722                 pdf.linkmargin 2 mul add sub
3723                 3 exch put
3724                 dup
3725                 pdf.outerbox 2 get
3726                 pdf.linkmargin add
3727                 2 exch put
3728                 dup dup
3729                 3 get
3730                 pdf.outerbox pdf.rect.ht
3731                 pdf.linkmargin 2 mul add add
3732                 1 exch put
3733             } /pdf.currentrect exch def
3734             pdf.breaklink.write
3735             {
3736                 pdf.currentrect
3737                 dup
3738                 pdf.outerbox 0 get
3739                 pdf.linkmargin sub
3740                 0 exch put
3741             }

```

```

3742         pdf.outerbox 2 get
3743         pdf.linkmargin add
3744         2 exch put
3745     dup dup
3746     1 get
3747     pdf.baselineskip add
3748     1 exch put
3749     dup dup
3750     3 get
3751     pdf.baselineskip add
3752     3 exch put
3753     /pdf.currentrect exch def
3754     pdf.breaklink.write
3755 }
3756 1 index 3 get
3757 pdf.linkmargin 2 mul add
3758 pdf.outerbox pdf.rect.ht add
3759 2 index 1 get sub
3760 pdf.baselineskip div round cvi 1 sub
3761     exch
3762     repeat
3763     pdf.currentrect
3764     dup
3765         pdf.outerbox 0 get
3766         pdf.linkmargin sub
3767         0 exch put
3768     dup dup
3769     1 get
3770     pdf.baselineskip add
3771     1 exch put
3772     dup dup
3773     3 get
3774     pdf.baselineskip add
3775     3 exch put
3776     dup 2 index 2 get 2 exch put
3777     /pdf.currentrect exch def
3778     pdf.breaklink.write
3779     SDict /pdf.pdfmark.good false put
3780     exit
3781 }
3782 { pdf.count 2 sub /pdf.count exch def }
3783 ifelse
3784 }
3785 loop
3786 }
3787 if
3788 /ANN
3789 }
3790 def
3791 /pdf.breaklink.write
3792 {
3793     counttomark 1 sub
3794     index /_objdef eq
3795     {

```

```

3796         counttomark -2 roll
3797         dup wcheck
3798         {
3799             readonly
3800             counttomark 2 roll
3801         }
3802         { pop pop }
3803         ifelse
3804     }
3805     if
3806     counttomark 1 add copy
3807     pop pdf.currentrect
3808     /ANN pdfmark
3809 }
3810 def

```

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

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

```

3811 /pdf.pdfmark
3812 {
3813     SDict /pdf.pdfmark.good true put
3814     dup /ANN eq
3815     {
3816         pdf.pdfmark.store
3817         pdf.pdfmark.dict
3818         begin
3819             Subtype /Link eq
3820             currentdict /Rect known and
3821             SDict /pdf.outerbox known and
3822             SDict /pdf.baselineskip known and
3823             {
3824                 Rect 3 get
3825                 pdf.linkmargin 2 mul add
3826                 pdf.outerbox pdf.rect.ht add
3827                 Rect 1 get sub
3828                 pdf.baselineskip div round cvi 0 gt
3829                 { pdf.breaklink }
3830                 if
3831             }
3832             if
3833         end
3834         SDict /pdf.outerbox undef
3835         SDict /pdf.baselineskip undef
3836         currentdict /pdf.pdfmark.dict undef
3837     }
3838     if
3839     pdf.pdfmark.good
3840     { pdfmark }
3841     { cleartomark }

```

```

3842     ifelse
3843   }
3844   def
3845 /pdf.pdfmark.store
3846   {
3847     /pdf.pdfmark.dict 65534 dict def
3848     counttomark 1 add copy
3849     pop
3850     {
3851       dup mark eq
3852       {
3853         pop
3854         exit
3855       }
3856       {
3857         pdf.pdfmark.dict
3858         begin def end
3859       }
3860     } ifelse
3861   }
3862   loop
3863 }
3864 def

```

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

```

3865 </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>	1157
A	
<code>\AtBeginDvi</code>	56
B	
bool commands:	
<code>\bool_gset_false:N</code>	1243, 1262, 1285, 1307, 1323, 1432, 1684, 1720, 2958, 3004
<code>\bool_gset_true:N</code>	1241, 1310, 1430, 1699, 2951, 2957
<code>\bool_if:NTF</code>	66, 613, 1253, 1257, 1273, 1276, 1280, 1291, 1298, 1302, 1314, 1318, 1443, 1448, 1453, 1658, 1703, 1832, 1884, 1886, 2025, 2070, 2072, 2946, 2961, 2966, 2971
<code>\bool_if:nTF</code>	2449, 2630, 2818
<code>\bool_lazy_and:nnTF</code>	826, 2177, 3335, 3411
<code>\bool_lazy_any:nTF</code>	1872, 2060
<code>\bool_new:N</code>	1244, 1311, 1433, 1700, 1822, 1988, 2931, 2932
<code>\bool_set_false:N</code>	1827, 1845, 1984, 2004, 2095, 2245
<code>\bool_set_true:N</code>	1844, 2012
box commands:	
<code>\box_dp:N</code>	235, 237, 285, 287, 342, 344, 391, 393, 395, 397, 2983, 3016, 3017, 3042
<code>\box_ht:N</code>	237, 287, 344, 395, 397, 1899, 2136, 2988, 3027, 3028, 3044
<code>\box_if_empty:N</code>	3076
<code>\box_move_down:nn</code>	2904, 2983
<code>\box_move_up:nn</code>	2266, 2906, 2988
<code>\box_new:N</code>	2894, 2895
<code>\box_set_dp:Nn</code>	1791
<code>\box_set_ht:Nn</code>	1790
<code>\box_set_wd:Nn</code>	299, 1789
<code>\box_use:N</code>	242, 260, 274, 290, 317, 331, 347, 363, 375, 426, 440, 459, 1383, 1591, 1792, 2936
<code>\box_wd:N</code>	236, 244, 286, 292, 343, 349, 392, 394, 1898, 2135
box internal commands:	
<code>__box_backend_clip:N</code>	224, 224, 279, 279, 336, 336, 380, 380
<code>\l__box_backend_cos_fp</code>	294
<code>__box_backend_rotate:Nn</code>	246, 246, 294, 294, 351, 351, 430, 430
<code>__box_backend_rotate_aux:Nn</code>	246, 247, 248, 294, 295, 296, 351, 352, 353
<code>__box_backend_scale:Nnn</code>	263, 263, 322, 322, 366, 366, 443, 443
<code>\l__box_backend_sin_fp</code>	294
C	
clist commands:	
<code>\clist_map_function:nN</code>	1331, 1463, 1727
<code>\clist_map_inline:nn</code>	3262
color internal commands:	
<code>__color_backend:nnn</code>	1064, 1079, 1087, 1093
<code>\g__color_backend_colorant_prop</code>	576, 595, 598, 621, 862
<code>__color_backend_devicen_colorants:n</code>	577, 577, 782, 920
<code>__color_backend_devicen_colorants:w</code>	577, 585, 592, 600
<code>__color_backend_devicen_init:nnn</code>	769, 769, 887, 887, 1114, 1114
<code>__color_backend_devicen_init:w</code>	887, 896, 925, 929
<code>__color_backend_fill:n</code>	1044
<code>__color_backend_fill:nN</code>	966, 966, 968, 969, 970, 992, 993, 995, 997, 998, 1017, 1026, 1027, 1029, 1031, 1032, 1053, 1054, 1056, 1058, 1059
<code>__color_backend_fill_cmyk:n</code>	1066
<code>__color_backend_fill_cmyk:nN</code>	966, 968, 992, 992, 1026, 1026, 1053, 1053
<code>__color_backend_fill_devicen:nnN</code>	976, 986, 1016, 1020, 1043, 1047, 1108, 1110
<code>__color_backend_fill_gray:nN</code>	966, 969, 992, 994, 1026, 1028, 1053, 1055
<code>__color_backend_fill_reset:</code>	988, 988, 1022, 1022, 1049, 1049, 1112, 1112
<code>__color_backend_fill_rgb:nN</code>	966, 970, 992, 996, 1026, 1030, 1053, 1057

```

\__color_backend_fill_separation:nnN
.. 976, 976, 986, 1016, 1016, 1020,
1043, 1043, 1047, 1108, 1108, 1110
\l_color_backend_fill_tl .....
..... 526, 538, 1000, 1013
\__color_backend_iccbased_-
device:nnn ..... 949, 949
\__color_backend_iccbased_-
init:nnn .....
..... 788, 788, 931, 931, 1114, 1115
\__color_backend_init_resource:n
..... 823, 823, 852, 923, 947, 962
\__color_backend_reset: .....
.... 506, 522, 530, 543, 547, 564,
988, 989, 1022, 1023, 1049, 1068, 1112
\__color_backend_rgb:w ..... 1081
\__color_backend_select:n .....
..... 547, 547, 552, 557, 562
\__color_backend_select:nN .....
.... 506, 507, 509, 511, 513, 514, 607
\__color_backend_select:nnN ....
..... 530, 531, 533, 535, 536, 819
\__color_backend_select_cmyk:nN .
..... 506, 506, 530, 530, 547, 549
\__color_backend_select_devicen:nnN
..... 604, 609, 791, 792, 813, 821
\__color_backend_select_gray:n . 569
\__color_backend_select_gray:nN .
..... 506, 508, 530, 532, 547, 554
\__color_backend_select_iccbased:nn
..... 795, 795
\__color_backend_select_iccbased:nnN
..... 610, 610, 813, 822
\__color_backend_select_named:nN
..... 506, 510, 566, 566
\__color_backend_select_rgb:nN ..
..... 506, 512, 530, 534, 547, 559
\__color_backend_select_separation:nnN
..... 604, 604, 609,
791, 791, 792, 813, 814, 818, 821, 822
\__color_backend_separation_-
init:n ..... 611, 692, 705
\__color_backend_separation_-
init:nn ..... 840, 850, 854
\__color_backend_separation_-
init:nnn ..... 611, 646, 667
\__color_backend_separation_-
init:nnnn ..... 611, 669, 681
\__color_backend_separation_-
init:nnnnn ..... 611,
611, 632, 725, 793, 793, 840, 840, 880
\__color_backend_separation_-
init:nw ..... 611, 696, 707, 721
\__color_backend_separation_-
init:w ..... 611, 683, 698, 703
\__color_backend_separation_-
init_/DeviceCMYK:nnn ..... 611
\__color_backend_separation_-
init_/DeviceGray:nnn ..... 611
\__color_backend_separation_-
init_/DeviceRGB:nnn ..... 611
\__color_backend_separation_-
init_aux:nnnnn .... 611, 617, 633
\__color_backend_separation_-
init_CIELAB:nnn .....
..... 611, 723, 793, 840, 865
\__color_backend_separation_-
init_CIELAB:nnnnn ..... 794
\__color_backend_separation_-
init_count:n ..... 611, 670, 673
\__color_backend_separation_-
init_count:w ... 611, 674, 675, 679
\__color_backend_separation_-
init_Device:Nn .....
..... 611, 655, 657, 659, 660
\l_color_backend_stack_int ....
..... 467, 541, 544, 1002, 1014
\__color_backend_stroke:nN .....
..... 966, 971, 973, 974,
975, 992, 1005, 1007, 1009, 1010, 1019
\__color_backend_stroke_cmyk:nN .
..... 966,
973, 992, 1004, 1026, 1037, 1064, 1064
\__color_backend_stroke_devicen:nn
..... 1043
\__color_backend_stroke_devicen:nnN
976, 987, 1016, 1021, 1048, 1108, 1111
\__color_backend_stroke_gray:nN .
..... 966,
974, 992, 1006, 1026, 1039, 1064, 1070
\__color_backend_stroke_gray_-
aux:n ..... 1064, 1074, 1078
\__color_backend_stroke_reset: ..
..... 988,
989, 1022, 1023, 1049, 1050, 1112, 1113
\__color_backend_stroke_rgb:nN ..
..... 966,
975, 992, 1008, 1026, 1041, 1064, 1080
\__color_backend_stroke_rgb:w ...
..... 1064, 1082
\__color_backend_stroke_separation:nnN
.. 976, 981, 987, 1016, 1018, 1021,
1043, 1045, 1048, 1108, 1109, 1111
\l_color_backend_stroke_tl ....
..... 526, 539, 1001, 1012
\g_color_model_int 618, 627, 775,
803, 852, 858, 859, 913, 914, 923, 947

```

\c__color_model_range_CIELAB_tl .	981, 992, 994, 996, 998, 1004, 1006,
..... 730, 765, 876, 883	1008, 1010, 1016, 1018, 1026, 1028,
\l__color_tmp_tl 607	1030, 1032, 1037, 1039, 1041, 1043,
color.sc 3484	1045, 1050, 1053, 1055, 1057, 1059,
cs commands:	1064, 1070, 1078, 1080, 1082, 1108,
\cs_generate_variant:Nn .. 62, 65,	1109, 1113, 1114, 1115, 1190, 1196,
170, 181, 212, 218, 632, 1189, 1600,	1201, 1203, 1205, 1213, 1221, 1230,
2039, 2106, 2126, 2293, 2308, 2371,	1240, 1242, 1245, 1247, 1264, 1269,
2581, 2594, 2704, 2719, 2749, 3197	1287, 1309, 1312, 1325, 1338, 1343,
\cs_gset:Npe .. 2461, 2465, 2823, 2828	1345, 1347, 1349, 1351, 1353, 1355,
\cs_gset_eq:NN 3418, 3419	1357, 1362, 1367, 1394, 1396, 1400,
\cs_gset_protected:Npn 3415, 3416, 3417	1405, 1410, 1420, 1429, 1431, 1434,
\cs_if_exist:NTF 27, 49, 2655, 2681, 3072	1436, 1438, 1440, 1445, 1450, 1455,
\cs_if_exist_p:N 827, 3336, 3412	1457, 1470, 1475, 1477, 1479, 1481,
\cs_if_exist_use:NTF 38, 645	1483, 1485, 1487, 1489, 1508, 1532,
\cs_new:Npe 577, 2595, 2606, 2673, 3123, 3158, 3254	1538, 1550, 1562, 1574, 1581, 1603,
\cs_new:Npn 592, 654, 656,	1609, 1614, 1619, 1630, 1640, 1650,
658, 660, 667, 673, 675, 681, 698,	1652, 1654, 1656, 1687, 1689, 1694,
705, 707, 925, 1336, 1468, 1731,	1696, 1698, 1701, 1722, 1733, 1746,
1901, 2139, 2283, 2300, 2372, 2374,	1748, 1750, 1752, 1754, 1756, 1758,
2467, 2468, 2550, 2551, 2563, 2582,	1760, 1762, 1770, 1778, 1804, 1823,
2583, 2686, 2712, 2750, 2752, 2831,	1841, 1856, 1861, 1869, 1902, 1915,
2832, 2844, 2845, 2850, 2851, 2856,	1933, 1943, 1959, 1972, 1981, 1990,
2857, 2926, 3097, 3211, 3240, 3249	2002, 2010, 2015, 2030, 2040, 2083,
\cs_new_eq:NN 46, 56, 58, 609, 792,	2092, 2098, 2104, 2107, 2114, 2127,
821, 822, 968, 969, 970, 973, 974,	2132, 2140, 2153, 2187, 2218, 2219,
975, 986, 987, 988, 989, 1020, 1021,	2221, 2223, 2225, 2231, 2234, 2242,
1022, 1023, 1047, 1048, 1049, 1110,	2248, 2251, 2253, 2264, 2291, 2294,
1111, 1112, 1188, 1392, 1393, 1398,	2296, 2298, 2302, 2309, 2326, 2331,
1399, 1599, 1601, 1602, 1608, 1802,	2336, 2341, 2351, 2356, 2364, 2376,
1803, 1815, 1816, 1839, 1840, 1907,	2402, 2407, 2435, 2447, 2459, 2463,
1908, 1909, 1932, 1957, 1969, 1970,	2469, 2471, 2475, 2498, 2512, 2522,
1978, 1979, 1980, 2001, 2007, 2008,	2533, 2552, 2584, 2617, 2628, 2634,
2009, 2079, 2089, 2090, 2091, 2232,	2662, 2696, 2698, 2705, 2707, 2710,
2233, 2240, 2241, 2250, 2280, 2281,	2714, 2720, 2725, 2730, 2732, 2734,
2282, 2285, 2301, 2713, 2936, 3260,	2742, 2754, 2776, 2781, 2814, 2816,
3261, 3271, 3375, 3376, 3432, 3433	2821, 2826, 2833, 2835, 2839, 2840,
\cs_new_protected:Npe 611, 1093, 2645, 2702, 3195, 3221	2841, 2842, 2843, 2846, 2847, 2848,
\cs_new_protected:Npn . 47, 53, 60,	2849, 2852, 2853, 2854, 2855, 2858,
63, 71, 77, 82, 84, 88, 98, 108, 118,	2859, 2862, 2881, 2888, 2897, 2902,
128, 137, 146, 156, 168, 171, 173,	2935, 2937, 2942, 2944, 2949, 2964,
175, 179, 184, 193, 203, 213, 224,	2969, 3006, 3035, 3054, 3063, 3099,
246, 248, 263, 279, 294, 296, 322,	3106, 3107, 3110, 3134, 3136, 3138,
336, 351, 353, 366, 380, 430, 443,	3149, 3169, 3179, 3186, 3199, 3214,
470, 484, 494, 506, 508, 510, 512,	3219, 3238, 3242, 3244, 3245, 3248,
514, 522, 530, 532, 534, 536, 543,	3250, 3251, 3252, 3253, 3255, 3256,
547, 549, 554, 559, 564, 566, 604,	3257, 3276, 3281, 3288, 3295, 3314,
610, 633, 723, 769, 788, 791, 793,	3319, 3326, 3350, 3365, 3377, 3383,
794, 795, 814, 818, 823, 840, 854,	3389, 3423, 3425, 3427, 3429, 3431
865, 887, 931, 949, 966, 971, 976,	
	\cs_set_eq:NN 3093, 3095
	\cs_set_protected:Npn 2191

D

dim commands:

```
\dim_compare:nNnTF ..... 2167, 2172
\dim_compare_p:nNn ..... 2178, 2179
\dim_eval:n .....
... 2405, 2508, 2509, 2510, 2779,
2870, 2871, 2874, 2900, 3118, 3119,
3120, 3177, 3205, 3206, 3207, 3243
\dim_gset:Nn ..... 2883, 2884
\dim_horizontal:n .....
.. 2415, 2422, 2793, 2812, 2910, 2912
\dim_max:nn ..... 3014, 3025
\dim_set:Nn .....
.. 1898, 1899, 2135, 2136, 2163, 2164
\dim_set_eq:NN ..... 2229
\dim_to_decimal:n .. 391, 392, 393,
394, 395, 397, 1612, 1617, 1623,
1624, 1625, 1626, 1635, 1636, 1637,
1728, 1747, 2273, 2274, 3012, 3023,
3041, 3042, 3043, 3044, 3048, 3103
\dim_to_decimal_in_bp:n .....
.... 235, 236, 237, 285, 286, 287,
342, 343, 344, 1209, 1210, 1217,
1218, 1225, 1226, 1234, 1235, 1236,
1333, 1337, 1341, 1403, 1408, 1414,
1415, 1416, 1424, 1425, 1465, 1469,
1473, 1732, 1809, 1810, 1811, 1812,
1995, 1996, 1997, 1998, 2054, 2055,
2056, 2057, 2258, 2259, 2260, 2261
\dim_vertical:n 2411, 2418, 2785, 2796
\dim_zero:N ..... 2161, 2162
\c_max_dim .....
.. 2163, 2164, 2167, 2172, 2178, 2179
```

draw internal commands:

```
\__draw_backend_add_to_path:n ...
..... 1609,
1611, 1616, 1621, 1632, 1640, 1655
\__draw_backend_begin: .....
.. 1190, 1190, 1394, 1394, 1603, 1603
\__draw_backend_box_use:Nnnnn ...
.. 1367, 1367, 1581, 1581, 1778, 1778
\__draw_backend_cap_but: .....
.. 1325, 1345, 1457, 1477, 1722, 1750
\__draw_backend_cap_rectangle: ..
.. 1325, 1349, 1457, 1481, 1722, 1754
\__draw_backend_cap_round: .....
.. 1325, 1347, 1457, 1479, 1722, 1752
\__draw_backend_clip: .....
.. 1245, 1309, 1434, 1450, 1654, 1698
\__draw_backend_closepath: .....
..... 1245, 1245,
1266, 1434, 1434, 1654, 1654, 1691
\__draw_backend_closestroke: ...
.. 1245, 1264, 1434, 1438, 1654, 1689
```

```
\__draw_backend_curveto:nnnnnn ..
.. 1205, 1230, 1400, 1410, 1609, 1630
\__draw_backend_dash:n .....
..... 1325, 1331, 1336,
1457, 1463, 1468, 1722, 1727, 1731
\__draw_backend_dash_aux:nn ....
..... 1722, 1726, 1733
\__draw_backend_dash_pattern:nn .
.. 1325, 1325, 1457, 1457, 1722, 1722
\__draw_backend_discardpath: ...
.. 1245, 1312, 1434, 1455, 1654, 1701
\__draw_backend_end: .....
.. 1190, 1196, 1394, 1396, 1603, 1608
\__draw_backend_evenodd_rule: ...
.. 1240, 1240, 1429, 1429, 1650, 1650
\__draw_backend_fill: .....
.. 1245, 1269, 1434, 1440, 1654, 1694
\__draw_backend_fillstroke: ....
.. 1245, 1287, 1434, 1445, 1654, 1696
\__draw_backend_join_bevel: ....
.. 1325, 1355, 1457, 1487, 1722, 1760
\__draw_backend_join_miter: ....
.. 1325, 1351, 1457, 1483, 1722, 1756
\__draw_backend_join_round: ....
.. 1325, 1353, 1457, 1485, 1722, 1758
\__draw_backend_lineto:nn .....
.. 1205, 1213, 1400, 1405, 1609, 1614
\__draw_backend_linewidth:n ....
.. 1325, 1338, 1457, 1470, 1722, 1746
\__draw_backend_literal:n .....
1188, 1188, 1189, 1192, 1193, 1194,
1198, 1199, 1202, 1204, 1207, 1215,
1223, 1232, 1246, 1249, 1250, 1251,
1252, 1255, 1261, 1271, 1278, 1284,
1289, 1294, 1295, 1296, 1297, 1300,
1306, 1316, 1322, 1327, 1340, 1344,
1346, 1348, 1350, 1352, 1354, 1356,
1359, 1364, 1369, 1370, 1371, 1372,
1373, 1374, 1375, 1376, 1377, 1381,
1382, 1384, 1385, 1386, 1387, 1388,
1392, 1392, 1393, 1402, 1407, 1412,
1422, 1435, 1437, 1439, 1442, 1447,
1452, 1456, 1459, 1472, 1476, 1478,
1480, 1482, 1484, 1486, 1488, 1534,
1599, 1599, 1600, 1661, 1680, 1706
\__draw_backend_miterlimit:n ...
.. 1325, 1343, 1457, 1475, 1722, 1748
\__draw_backend_moveto:nn .....
.. 1205, 1205, 1400, 1400, 1609, 1609
\__draw_backend_nonzero_rule: ...
.. 1240, 1242, 1429, 1431, 1650, 1652
\__draw_backend_path:n .....
..... 1654, 1656, 1688, 1695, 1697
\g__draw_backend_path_int 1669, 1686
```


<code>\g__draw_backend_path_tl</code>	<code>\exp_not:n</code>
.. 1609 , 1665 , 1681 , 1683 , 1710 , 1719	48, 96, 116 , 154 ,
<code>__draw_backend_rectangle:nnnn</code> ..	939 , 2329 , 2334 , 2398 , 2567 , 2568 ,
.. 1205 , 1221 , 1400 , 1420 , 1609 , 1619	2582 , 2583 , 2723 , 2728 , 2739 , 2758
<code>__draw_backend_scope_begin:</code> 1201 ,	<code>\ExplBackendFileDate</code> 1
1201 , 1395 , 1398 , 1398 , 1601 , 1601	
<code>__draw_backend_scope_end:</code> 1201 ,	
1203 , 1397 , 1398 , 1399 , 1601 , 1602	
<code>__draw_backend_shift:nn</code>	
.. 1357 , 1362 , 1489 , 1532 , 1762 , 1770	
<code>__draw_backend_stroke:</code> 1245 , 1247 ,	
1267 , 1434 , 1436 , 1654 , 1687 , 1692	
<code>__draw_backend_transform:nnnn</code> ..	
..... 1357 , 1357 , 1378 , 1379 ,	
1380 , 1489 , 1489 , 1762 , 1762 , 1781	
<code>__draw_backend_transform_-</code>	
aux:nnnn 1489 , 1503 , 1508	
<code>__draw_backend_transform_-</code>	
decompose:nnnnN . 1502 , 1537 , 1538	
<code>__draw_backend_transform_-</code>	
decompose_auxi:nnnnN	
..... 1537 , 1542 , 1550	
<code>__draw_backend_transform_-</code>	
decompose_auxii:nnnnN	
..... 1537 , 1554 , 1562	
<code>__draw_backend_transform_-</code>	
decompose_auxiii:nnnnN	
..... 1537 , 1566 , 1574	
<code>\g__draw_draw_clip_bool</code> .. 1245 , 1654	
<code>\g__draw_draw_eor_bool</code>	
... 1240 , 1257 , 1273 , 1280 , 1291 ,	
1302 , 1318 , 1429 , 1443 , 1448 , 1453	
<code>\g__draw_draw_path_int</code> 1654	
E	
<code>\errmessage</code> 38	
<code>\evensidemargin</code> 2981	
exp commands:	
<code>\exp_args:Ne</code> 615 ,	
669 , 850 , 1863 , 1921 , 1923 , 1947 ,	
1949 , 2338 , 2353 , 2404 , 2778 , 2977	
<code>\exp_args:Nf</code> 1330 , 1462 , 2899	
<code>\exp_args:Nne</code> 2745	
<code>\exp_args:NNf</code> 247 , 295 , 352	
<code>\exp_args:Nno</code> 3379	
<code>\exp_args:No</code> 3385	
<code>\exp_not:N</code> 579 ,	
585 , 586 , 587 , 615 , 617 , 618 , 621 ,	
622 , 627 , 2597 , 2599 , 2602 , 2608 ,	
2610 , 2613 , 2650 , 2651 , 2657 , 2658 ,	
2677 , 2682 , 3125 , 3127 , 3130 , 3160 ,	
3162 , 3165 , 3223 , 3224 , 3225 , 3230	
	<code>\file_compare_timestamp:nNnTF</code> . 1935
	<code>\file_parse_full_name:nNNN</code> 1917 , 1945
	<code>\fmtversion</code> 51
	fp commands:
	<code>\fp_compare:nNnTF</code>
	. 254 , 301 , 307 , 359 , 1513 , 1526 , 1576
	<code>\fp_eval:n</code>
	. 247 , 256 , 269 , 270 , 295 , 312 , 327 ,
	329 , 352 , 361 , 372 , 373 , 437 , 452 ,
	453 , 1075 , 1088 , 1089 , 1090 , 1515 ,
	1520 , 1521 , 1528 , 1543 , 1544 , 1545 ,
	1546 , 1555 , 1556 , 1557 , 1558 , 1567 ,
	1568 , 1569 , 1570 , 2395 , 2495 , 2772
	<code>\fp_new:N</code> 320 , 321
	<code>\fp_set:Nn</code> 300 , 303
	<code>\fp_use:N</code> 306 , 310 , 315
	<code>\fp_zero:N</code> 302
	<code>\c_zero_fp</code> 254 , 301 , 307 , 359 , 1513 , 1526
	G
	graphics commands:
	<code>\l_graphics_search_ext_seq</code>
 1801 , 1819 , 1967 , 2151
	graphics internal commands:
	<code>\l_graphics_attr_tl</code> 1821 ,
	1828 , 1846 , 1858 , 1865 , 1867 , 1905
	<code>__graphics_backend_dequote:w</code> ...
 1823 , 1864 , 1901
	<code>\l_graphics_backend_dir_str</code> . 1910
	<code>\l_graphics_backend_ext_str</code> . 1910
	<code>__graphics_backend_get_pagecount:n</code>
 1816 , 1816 , 1959 , 1959 ,
	2078 , 2079 , 2140 , 2140 , 2285 , 2285
	<code>__graphics_backend_getbb_auxi:n</code>
 1823 , 1837 , 1854 , 1856
	<code>__graphics_backend_getbb_-</code>
	auxi:nN 2083 , 2087 , 2096 , 2098
	<code>__graphics_backend_getbb_-</code>
	auxii:n 1823 , 1859 , 1861
	<code>__graphics_backend_getbb_-</code>
	auxii:nnN .. 2083 , 2101 , 2104 , 2106
	<code>__graphics_backend_getbb_-</code>
	auxiii:n 1823 , 1863 , 1869
	<code>__graphics_backend_getbb_-</code>
	auxiii:nNnn . 2083 , 2102 , 2105 , 2107
	<code>__graphics_backend_getbb_-</code>
	auxiv:nnNnn . 2083 , 2110 , 2114 , 2126

```

\__graphics_backend_getbb_-
  auxv:nNnn .. 2083, 2111, 2118, 2127
\__graphics_backend_getbb_-
  auxvi:nNnn ..... 2130, 2132
\__graphics_backend_getbb_bmp:n .
  ..... 1969, 1980, 2083, 2091
\__graphics_backend_getbb_eps:n .
  ..... 1802, 1802, 1910,
  1915, 1932, 1969, 1969, 2232, 2232
\__graphics_backend_getbb_eps:nm
  ..... 1910
\__graphics_backend_getbb_eps:nn
  ..... 1921, 1933
\__graphics_backend_getbb_jpeg:n
  ..... 1823, 1839,
  1969, 1978, 2083, 2089, 2234, 2240
\__graphics_backend_getbb_jpg:n .
  1823, 1823, 1839, 1840, 1969, 1972,
  1978, 1979, 1980, 2083, 2083, 2089,
  2090, 2091, 2234, 2234, 2240, 2241
\__graphics_backend_getbb_-
  pagebox:w ..... 2083, 2122, 2139
\__graphics_backend_getbb_pdf:n .
  ..... 1823, 1841, 1941,
  1969, 1981, 2083, 2092, 2242, 2242
\__graphics_backend_getbb_png:n .
  ..... 1823, 1840,
  1969, 1979, 2083, 2090, 2234, 2241
\__graphics_backend_getbb_ps:n ..
  ..... 1802, 1803,
  1910, 1932, 1969, 1970, 2232, 2233
\__graphics_backend_getbb_svg:n .
  ..... 2153, 2153
\__graphics_backend_getbb_svg_-
  auxi:nNn ... 2153, 2169, 2174, 2187
\__graphics_backend_getbb_svg_-
  auxii:w .... 2153, 2191, 2213, 2218
\__graphics_backend_getbb_svg_-
  auxiii:Nw ..... 2153, 2201, 2219
\__graphics_backend_getbb_svg_-
  auxiv:Nw ..... 2153, 2204, 2221
\__graphics_backend_getbb_svg_-
  auxv:Nw ..... 2153, 2205, 2223
\__graphics_backend_getbb_svg_-
  auxvi:Nn 2153, 2220, 2222, 2224, 2225
\__graphics_backend_getbb_svg_-
  auxvii:w ..... 2153, 2227, 2231
\__graphics_backend_include:nn ..
  ..... 2248, 2249, 2252, 2253
\__graphics_backend_include_-
  auxi:n ..... 1990, 2005, 2013, 2015
\__graphics_backend_include_-
  auxii:nn ... 1990, 2017, 2030, 2039
\__graphics_backend_include_-
  auxiii:nn ..... 1990, 2037, 2040
\__graphics_backend_include_-
  bmp:n ..... 1990, 2008
\__graphics_backend_include_-
  dequote:w ..... 2264, 2275, 2283
\__graphics_backend_include_-
  eps:n ..... 1804,
  1804, 1815, 1910, 1943, 1957,
  1990, 1990, 2001, 2248, 2248, 2250
\__graphics_backend_include_-
  jpeg:n . 1902, 1907, 2007, 2264, 2281
\__graphics_backend_include_-
  jpg:n ..... 1902,
  1902, 1907, 1908, 1909, 1990,
  2002, 2007, 2008, 2009, 2264, 2282
\__graphics_backend_include_-
  jpseg:n ..... 1990
\__graphics_backend_include_-
  pdf:n ..... 1902,
  1908, 1947, 1990, 2010, 2248, 2251
\__graphics_backend_include_-
  png:n .....
  .. 1902, 1909, 1990, 2009, 2264, 2280
\__graphics_backend_include_ps:n
  ..... 1804, 1815,
  1910, 1957, 1990, 2001, 2248, 2250
\__graphics_backend_include_-
  svg:n .. 2264, 2264, 2280, 2281, 2282
\l__graphics_backend_name_str . 1910
\__graphics_bb_restore:nTF .....
  ..... 1858, 2129, 2155
\__graphics_bb_save:n 1867, 2137, 2182
\l__graphics_decodearray_str ...
  ..... 1830, 1831,
  1843, 1876, 1882, 1883, 1983, 2023,
  2024, 2064, 2068, 2069, 2094, 2244
\__graphics_extract_bb:n .....
  ..... 1976, 1985, 2238, 2246
\l__graphics_final_name_str .. 1940
\__graphics_get_pagecount:n ....
  ..... 1816, 2079, 2285
\l__graphics_interpolate_bool ...
  ..... 1832, 1845, 1874, 1886,
  1984, 2025, 2062, 2072, 2095, 2245
\l__graphics_llx_dim .....
  ..... 1809, 1995, 2054, 2161, 2258
\l__graphics_lly_dim .....
  ..... 1810, 1996, 2055, 2162, 2259
\l__graphics_page_int 1825, 1849,
  1850, 1891, 1892, 1974, 2021, 2022,
  2048, 2049, 2085, 2100, 2101, 2236
\l__graphics_pagebox_tl ... 1826,
  1848, 1893, 1894, 1975, 2019, 2020,

```


<code>__kernel_backend_literal_svg:n</code> . . 179 , 179 , 181 , 186 , 197 , 205 , 215 , 383 , 385 , 402 , 797 , 1599 , 1782 , 1793	<code>\mode_if_math:TF</code> 2956
<code>__kernel_backend_matrix:n</code> 146 , 146 , 156 , 304 , 325 , 1492 , 1585	msg commands: <code>\msg_error:nnn</code> 570 , 2159 <code>\msg_new:nnn</code> 572
<code>__kernel_backend_postscript:n</code> 63 , 63 , 65 , 519 , 1038 , 1040 , 1042 , 1046 , 2292 , 2343 , 2358 , 2378 , 2412 , 2419 , 2905 , 2911 , 2916 , 2952 , 2984 , 2991 , 2995 , 3009 , 3037 , 3080 , 3087 , 3094 , 3101 , 3297	O <code>\oddsidemargin</code> 2980
<code>__kernel_backend_scope:n</code> 184 , 213 , 218 , 412 , 417 , 1067 , 1095 , 1606 , 1651 , 1653 , 1673 , 1713 , 1735 , 1747 , 1749 , 1751 , 1753 , 1755 , 1757 , 1759 , 1761 , 1764 , 1772 , 3430	opacity internal commands: <code>__opacity_backend:nn</code> 3423 , 3424 , 3426 , 3428 , 3429 <code>__opacity_backend:nnn</code> 3276 , 3278 , 3279 , 3283 , 3290 , 3295 , 3321 , 3328 <code>__opacity_backend_fill:n</code> 3276 , 3281 , 3377 , 3377 , 3423 , 3425 <code>__opacity_backend_fill_stroke:nn</code> . 3377 , 3379 , 3385 , 3389 , 3411 , 3416 <code>\l__opacity_backend_fill_tl</code> 3346 , 3352 , 3386 , 3394 <code>__opacity_backend_reset:</code> 3276 , 3314 , 3350 , 3365 , 3375 , 3376 , 3411 , 3417 , 3418 , 3419 , 3423 , 3431 , 3432 , 3433 <code>__opacity_backend_reset_fill:</code> 3276 , 3316 , 3319 , 3350 , 3375 , 3411 , 3418 , 3423 , 3432 <code>__opacity_backend_reset_stroke:</code> 3276 , 3317 , 3326 , 3350 , 3376 , 3411 , 3419 , 3423 , 3433 <code>__opacity_backend_select:n</code> 3276 , 3276 , 3350 , 3350 , 3392 , 3411 , 3415 , 3423 , 3423 <code>\c__opacity_backend_stack_int</code> 3335 , 3361 , 3372 , 3406 <code>__opacity_backend_stroke:n</code> 3276 , 3288 , 3377 , 3383 , 3423 , 3427 <code>\l__opacity_backend_stroke_tl</code> 3346 , 3353 , 3381 , 3395
<code>__kernel_backend_scope_begin:</code> .. . 82 , 82 , 128 , 128 , 173 , 173 , 184 , 184 , 226 , 250 , 265 , 281 , 298 , 324 , 338 , 355 , 368 , 1398 , 1583 , 1601 , 1605 , 1780	P pdf commands: <code>\pdf_object_if_exist:nTF</code> 867 , 933 , 951 <code>\pdf_object_new:n</code> 858 , 869 , 913 , 935 , 953 <code>\pdf_object_ref:n</code> 815 , 882 , 946 , 961 , 979 , 984 <code>\pdf_object_ref_last:</code> 835 , 860 , 863 , 919 <code>\pdf_object_unnamed_write:nn</code> 842 , 889 , 945 , 960 <code>\pdf_object_write:nnn</code> 859 , 870 , 914 , 936 , 954
<code>__kernel_backend_scope_end:</code> 82 , 84 , 128 , 137 , 173 , 175 , 184 , 193 , 243 , 261 , 275 , 291 , 318 , 332 , 348 , 364 , 376 , 427 , 441 , 460 , 1399 , 1595 , 1602 , 1608 , 1794	pdf internal commands: <code>__pdf_backend:n</code> 2702 , 2702 , 2704 , 2706 , 2708 , 2722 , 2727 , 2736 , 2756 , 2788 , 2789 , 2799
<code>\g__kernel_backend_scope_int</code> 182 , 189 , 191 , 196 , 200 , 208 , 210 , 216	
<code>\l__kernel_backend_scope_int</code> 182 , 188 , 201 , 207	
<code>\g__kernel_clip_path_int</code> 380 , 1660 , 1663 , 1676 , 1705 , 1708 , 1716	
<code>__kernel_color_backend_stack_- init:Nnn</code> 470 , 470 , 3340	
<code>__kernel_color_backend_stack_- pop:n</code> 484 , 494 , 544 , 3372	
<code>__kernel_color_backend_stack_- push:nn</code> 484 , 484 , 541 , 1002 , 1014 , 3361 , 3406	
<code>__kernel_dependency_version_- check:Nn</code> 1	
<code>__kernel_dependency_version_- check:nn</code> 27 , 29	
<code>__kernel_file_name_quote:n</code> 1923 , 1949	
L	
lua commands: <code>\lua_load_module:n</code> 1182	
M	
<code>\MessageBreak</code> 40	
mode commands: <code>\mode_if_horizontal:TF</code> ... 3058 , 3065	

_pdf_backend_annotation:nnnn	3260	_pdf_backend_object_write_-	
_pdf_backend_annotation_last:	3261	array:nn	... 2302, 2326, 2714, 2720
_pdf_backend_bdc:nn	2469, 2469,	_pdf_backend_object_write_-	
2696, 2696, 2833, 2833, 2858, 2858		aux:nnn	... 2302, 2304, 2309, 2367
_pdf_backend_catalog_gput:nn	..	_pdf_backend_object_write_-	
..... 2294, 2294,		dict:nn	... 2302, 2331, 2714, 2725
2512, 2512, 2705, 2705, 2841, 2841		_pdf_backend_object_write_-	
_pdf_backend_compress_objects:n		fstream:nn	.. 2302, 2336, 2714, 2730
..... 2435, 2447,		_pdf_backend_object_write_-	
2617, 2628, 2814, 2816, 2852, 2853		fstream:nnn 2339, 2341
_pdf_backend_compresslevel:n	..	_pdf_backend_object_write_-	
..... 2435, 2435,		stream:nn	.. 2302, 2351, 2714, 2732
2617, 2617, 2814, 2814, 2852, 2852		_pdf_backend_object_write_-	
_pdf_backend_destination:nn	...	stream:nnn 2302, 2354, 2356
..... 2376, 2376,		_pdf_backend_object_write_-	
2475, 2475, 2754, 2754, 2839, 2839		stream:nnnn	.. 2714, 2731, 2733, 2734
_pdf_backend_destination:nnnn	..	_pdf_backend_pageobject_ref:n	..
..... 2376, 2402,	 2374, 2374,	
2475, 2498, 2754, 2776, 2839, 2840		2606, 2606, 2752, 2752, 2843, 2851	
_pdf_backend_destination_-		_pdf_backend_pagesize_gset:nn	..
aux:nnnn 2862, 2862, 2881, 2881, 2888, 2888	
.. 2376, 2404, 2407, 2754, 2778, 2781		_pdf_backend_pdfmark:n
_pdf_backend_emc: .. 2469, 2471,		2291, 2291, 2293, 2295, 2297, 2311,	
2696, 2698, 2833, 2835, 2858, 2859		2328, 2333, 2379, 2423, 2470, 2472	
_pdf_backend_info_gput:nn	_pdf_backend_version_major: ..	
..... 2294, 2296,		... 2461, 2467, 2467, 2673, 2673,	
2512, 2522, 2705, 2707, 2841, 2842		2823, 2824, 2831, 2831, 2856, 2856	
_pdf_backend_objcompresslevel:n		_pdf_backend_version_major_-	
..... 2617, 2631, 2632, 2634		gset:n 2459, 2459,
_pdf_backend_object_id:n	2645, 2645, 2821, 2821, 2854, 2854	
..... 2298, 2301,		_pdf_backend_version_minor: ..	
2533, 2551, 2710, 2713, 2843, 2845		... 2465, 2467, 2468, 2673, 2686,	
_g_pdf_backend_object_int	2828, 2829, 2831, 2832, 2856, 2857	
..... 2299, 2366, 2368,		_pdf_backend_version_minor_-	
2373, 2542, 2711, 2744, 2746, 2751		gset:n 2459, 2463,
_pdf_backend_object_last: ..		2645, 2662, 2821, 2826, 2854, 2855	
..... 2372, 2372,		_pdf_exp_not_i:nn
2595, 2595, 2750, 2750, 2843, 2850	 2552, 2571, 2576, 2582	
_pdf_backend_object_new: ..		_pdf_exp_not_ii:nn
..... 2298, 2298,	 2552, 2572, 2577, 2583	
2533, 2533, 2710, 2710, 2843, 2843		pdf.baselineskip 3811
_pdf_backend_object_now:nn	...	pdf.bordertracking 3569
2364, 2364, 2371, 2584, 2584, 2594,		pdf.bordertracking.begin 3569
2742, 2742, 2749, 2843, 2848, 2849		pdf.bordertracking.continue 3569
_g_pdf_backend_object_prop	pdf.bordertracking.end 3569
..... 2532, 2709		pdf.bordertracking.endpage 3569
_pdf_backend_object_ref:n	pdf.breaklink 3707
2298, 2300, 2301, 2305, 2533, 2550,		pdf.breaklink.write 3707
2710, 2712, 2713, 2717, 2843, 2844		pdf.brokenlink.dict 3569
_pdf_backend_object_write:nn	..	pdf.brokenlink.rect 3569
..... 2552, 2561, 2563, 2592, 2843		pdf.brokenlink.skip 3569
_pdf_backend_object_write:nnn	..	pdf.count 3707
2302, 2302, 2308, 2552, 2552, 2581,		pdf.currentrect 3707
2714, 2714, 2719, 2843, 2846, 2847		pdf.cvs 3491

pdf.dest.anchor	3534	\g__pdfannot_backend_int	2896, 2915, 2919, 2927, 2993, 2994, 3198, 3201, 3204, 3212, 3223, 3225
pdf.dest.point	3534	__pdfannot_backend_last:	2926, 2926, 3123, 3123, 3211, 3211, 3249, 3249, 3261
pdf.dest.x	3534	__pdfannot_backend_link:nw	2937
pdf.dest.y	3534	__pdfannot_backend_link_aux:nw	2937
pdf.dest2device	3534	__pdfannot_backend_link_begin:n	3214, 3216, 3220, 3221
pdf.dev.x	3534	__pdfannot_backend_link_-begin:nnnw	3134, 3135, 3137, 3138, 3250, 3252
pdf.dev.y	3534	__pdfannot_backend_link_-begin:nw	2939, 2943, 2944
pdf.dvi.pt	3491	__pdfannot_backend_link_begin_-aux:nw	2947, 2949
pdf.globaldict	3488	__pdfannot_backend_link_begin_-goto:nnw	2937, 2937, 3134, 3134, 3214, 3214, 3250, 3250
pdf.leftboundary	3569	__pdfannot_backend_link_begin_-user:nnw	2937, 2942, 3134, 3136, 3214, 3219, 3250, 3251
pdf.linkdp.pad	3495	\g__pdfannot_backend_link_bool	2932, 2946, 2951, 2966, 3004
pdf.linkht.pad	3495	\g__pdfannot_backend_link_dict_-tl	2929, 2954, 2999
pdf.linkmargin	3495	__pdfannot_backend_link_end:	2937, 2964, 3134, 3149, 3214, 3238, 3250, 3253
pdf.llx	3498	__pdfannot_backend_link_end_-aux:	2937, 2967, 2969
pdf.lly	3498	\g__pdfannot_backend_link_int	2928, 2994, 2998, 3098, 3213, 3224, 3230, 3241
pdf.originx	3569	__pdfannot_backend_link_last:	3097, 3097, 3158, 3158, 3240, 3240, 3254, 3254
pdf.originy	3569	__pdfannot_backend_link_-margin:n	3099, 3099, 3169, 3169, 3242, 3242, 3255, 3255
pdf.outerbox	3811	\g__pdfannot_backend_link_math_-bool	2931, 2957, 2958, 2961, 2971
pdf.pdfmark	3811	__pdfannot_backend_link_minima:	2937, 2975, 3006
pdf.pdfmark.dict	3811	__pdfannot_backend_link_off:	3106, 3107, 3179, 3186, 3244, 3245, 3256, 3257
pdf.pdfmark.good	3811	__pdfannot_backend_link_on:	3106, 3106, 3179, 3179, 3244, 3244, 3256, 3256
pdf.pt.dvi	3491	__pdfannot_backend_link_-outerbox:n	2937, 2977, 3035
pdf.rect	3498		
pdf.rect.ht	3491		
pdf.rightboundary	3569		
pdf.save.linkll	3498		
pdf.save.linkur	3498		
pdf.save.ll	3498		
pdf.save.ur	3498		
pdf.tmpa	3534		
pdf.tmpb	3534		
pdf.tmpc	3534		
pdf.tmpd	3534		
pdf.urx	3498		
pdf.ury	3498		
pdfannot internal commands:			
__pdfannot_backend:n	3195, 3195, 3197, 3202, 3226, 3239, 3244, 3245		
\l__pdfannot_backend_breaklink_-pdfmark_tl	2933, 3001, 3092		
__pdfannot_backend_breaklink_-postscript:n	2935, 2935, 2985, 2987, 3093		
__pdfannot_backend_breaklink_-usebox:N	2936, 2936, 2986, 3095		
\l__pdfannot_backend_content_box	2894, 2959, 2983, 2986, 2988, 3017, 3028		
__pdfannot_backend_generic:nnnn	2897, 2897, 3110, 3110, 3199, 3199, 3248, 3248, 3260		
__pdfannot_backend_generic_-aux:nnnn	2897, 2899, 2902		

<code>\g_pdfannot_backend_link_sf_int</code> 2930, 3056, 3067, 3068 <code>_pdfannot_backend_link_sf_-</code> restore: ... 2937, 2960, 3003, 3063 <code>_pdfannot_backend_link_sf_-</code> save: 2937, 2955, 2973, 3054 <code>\l_pdfannot_backend_model_box</code> 2895, 2976, 3008, 3016, 3027, 3042, 3044 pdfmanagement commands: <code>\pdfmanagement_add:nnn</code> 832, 3343, 3354, 3396, 3399 <code>\pdfmanagement_if_active_p:</code> 827, 828, 3336, 3337, 3412, 3413 peek commands: <code>\peek_meaning:NTF</code> 2200, 2203 <code>\peek_remove_spaces:n</code> 2198 prg commands: <code>\prg_replicate:nn</code> 195, 663, 684, 694, 895 prop commands: <code>\prop_gput:Nnn</code> 621, 862 <code>\prop_if_in:NnTF</code> 595 <code>\prop_item:Nn</code> 598 <code>\prop_new:N</code> 576, 2532, 2709 <code>\ProvidesExplFile</code> 2	str commands: <code>\c_hash_str</code> 415, 1669, 1676, 1716 <code>\c_percent_str</code> 1101, 1102, 1103 <code>\str_case:nn</code> 901, 2315, 2565 <code>\str_case:nnTF</code> 2383, 2484, 2761 <code>\str_convert_pdfname:n</code> . 622, 642, 851 <code>\str_if_empty:NTF</code> 1834, 1851 <code>\str_if_empty_p:N</code> 1877 <code>\str_if_eq:nnTF</code> . 568, 801, 1495, 3391 <code>\str_new:N</code> 1912, 1913, 1914 <code>\str_tail:N</code> 1926, 1952 sys commands: <code>\sys_if_shell:TF</code> 1910 <code>\sys_shell_now:n</code> 1937
T	
TeX and L ^A T _E X 2 _ε commands:	
<code>\@ifl@t@r</code> 49, 51 <code>\current@color</code> 27 <code>\special</code> 2	
tex commands:	
<code>\tex_afterassignment:D</code> 2227 <code>\tex_baselineskip:D</code> 3048 <code>\tex_endinput:D</code> 44 <code>\tex_global:D</code> 2619, 2636, 2650, 2657, 2664 <code>\tex_immediate:D</code> 1871, 2555, 2558, 2587, 2590 <code>\tex_luatexversion:D</code> 2648, 2676 <code>\tex_pageheight:D</code> 2884 <code>\tex_pagewidth:D</code> 2883 <code>\tex_pdfannot:D</code> 3116 <code>\tex_pdfcatalog:D</code> 2518 <code>\tex_pdfcolorstack:D</code> 490, 500 <code>\tex_pdfcolorstackinit:D</code> 478 <code>\tex_pdfcompresslevel:D</code> 2624 <code>\tex_pdfdest:D</code> 2481, 2504 <code>\tex_pdfendlink:D</code> 3155 <code>\tex_pdfextension:D</code> . 91, 101, 111, 121, 131, 140, 149, 159, 487, 497, 2478, 2501, 2515, 2525, 2536, 2555, 2587, 3113, 3141, 3152, 3181, 3188 <code>\tex_pdffeedback:D</code> 475, 2544, 2599, 2610, 3127, 3162 <code>\tex_pdfinfo:D</code> 2528 <code>\tex_pdflastannot:D</code> 3130 <code>\tex_pdflastlink:D</code> 3165 <code>\tex_pdflastobj:D</code> 2547, 2602 <code>\tex_pdflastximage:D</code> 1866, 1897 <code>\tex_pdflastximagepages:D</code> 1963 <code>\tex_pdflinkmargin:D</code> 3175 <code>\tex_pdfliteral:D</code> ... 94, 104, 114, 124 <code>\tex_pdfmajorversion:D</code> 2655, 2657, 2681, 2682	
Q	
quark commands:	
<code>\quark_if_recursion_tail_stop:n</code> 594 <code>\q_recursion_stop</code> 587 <code>\q_recursion_tail</code> 586	
S	
scan commands:	
<code>\scan_stop:</code> 131, 140, 502, 2228, 2231, 2496, 2510, 2626, 2643, 2651, 2658, 2671, 3152, 3177	
scan internal commands:	
<code>\s_color_stop</code> 674, 675, 679, 683, 696, 699, 703, 707, 721, 896, 925, 929, 1081, 1083 <code>\s_graphics_stop</code> 1864, 1901, 2193, 2208, 2215, 2219, 2221, 2223, 2275, 2283	
separation 3485	
seq commands:	
<code>\seq_set_from_clist:Nn</code> 1801, 1819, 1967, 2151	
shipout commands:	
<code>\l_shipout_box</code> 3076, 3078, 3086	
skip commands:	
<code>\skip_horizontal:n</code> 244, 292, 349	

