

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}{2026-07-20}{}
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2026-07-20}{}
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2026-07-20}{}
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2026-07-20}{}
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2026-07-20}{}
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2026-07-20}{}
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` `\_box_backend_rotate_aux:Nn` Rotating using **dvips** does not require that the box dimensions are altered and has a very convenient built-in operation. Zero rotation must be written as 0 not -0 so there is a quick test.

```

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 sine/cosine values not the angle itself. We store the rounded values to avoid rounding twice. There are also a couple of comparisons to ensure that `-0` is not written to the output, as this avoids any issues with problematic display programs. Note that numbers are compared to 0 after rounding.

`\__box_backend_rotate_aux:Nn`
`\l__box_backend_cos_fp`
`\l__box_backend_sin_fp`

```

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 LuaTeX/pdfTeX: 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 $\text{\TeX}$ 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 $\text{\TeX}$ box is inserted to get things back on track. The clip path needs to come between those two such that if lines up with the current point, as does the $\text{\TeX}$ box.

```
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_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_rgb:nN #1
511 { \_color_backend_select:nN { rgb ~ #1 } }
512 \cs_new_protected:Npn \_color_backend_select:nN #1#2
513 {
514   \tl_set:Nn #2 {#1}
515   \_kernel_backend_literal:n { color~push~ #1 }
516 <*dvips>

```

```

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

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

522 </dvips | dvisvgm>

```

3.2.2 LuaTeX and pdfTeX

```

523 <*luatex | pdftex>

```

```

\l__color_backend_fill_tl
\l__color_backend_stroke_tl

```

```

524 \tl_new:N \l__color_backend_fill_tl
525 \tl_new:N \l__color_backend_stroke_tl
526 \tl_set:Nn \l__color_backend_fill_tl { 0 ~ g }
527 \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.

```

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

```

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

```

543 </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.

```

544 <*dvipdfmx | xetex>

```

```

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

```

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

```

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

```

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

```

564 </dviptdpmx | xetex>

```

3.3 Separations

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

```

565 <*dviptdpmx | luatex | pdftex | xetex | dvips>

```

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

```

\g_color_backend_colorant_prop

```

```

566 \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

```

```

567 \cs_new:Npe \__color_backend_devicen_colorants:n #1
568 {
569   \exp_not:N \tl_if_blank:nF {#1}
570   {
571     \c_space_tl
572     << ~
573     /Colorants ~
574     << ~
575     \exp_not:N \__color_backend_devicen_colorants:w #1 ~
576     \exp_not:N \q_recursion_tail \c_space_tl
577     \exp_not:N \q_recursion_stop
578     >> ~
579     >>
580   }
581 }
582 \cs_new:Npn \__color_backend_devicen_colorants:w #1 ~

```

```

583 {
584   \quark_if_recursion_tail_stop:n {#1}
585   \prop_if_in:NnT \g__color_backend_colorant_prop {#1}
586   {
587     #1 ~
588     \prop_item:Nn \g__color_backend_colorant_prop {#1} ~
589   }
590   \__color_backend_devicen_colorants:w
591 }

```

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

```

592 </dvipdfmx | luatex | pdftex | xetex | dvips>

```

```

593 <*dvips>

```

```

\__color_backend_select_separation:nnN
\__color_backend_select_devicen:nnN

```

```

594 \cs_new_protected:Npn \__color_backend_select_separation:nnN #1#2#3
595 {
596   \tl_set:Nn #3 { ~ #1 ~ #2 }
597   \__color_backend_select:nN { separation ~ #1 ~ #2 } \l__color_tmp_tl
598 }
599 \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.

```

600 \cs_new_protected:Npn \__color_backend_select_iccbased:nnN #1#2#3 { }

```

(End of definition for __color_backend_select_iccbased:nnN.)

```

\__color_backend_separation_init:nnnnn
\__color_backend_separation_init:neenn
\__color_backend_separation_init_aux:nnnnnn
\__color_backend_separation_init_DeviceCMYK:nnn
\__color_backend_separation_init_DeviceGray:nnn
\__color_backend_separation_init_DeviceRGB:nnn
\__color_backend_separation_init_Device:Nn
\__color_backend_separation_init:nnn
\__color_backend_separation_init_count:n
\__color_backend_separation_init_count:w
\__color_backend_separation_init:nnnn
\__color_backend_separation_init:w
\__color_backend_separation_init:n
\__color_backend_separation_init:nw
\__color_backend_separation_init_CIELAB:nnn

```

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.

```

601 \cs_new_protected:Npe \__color_backend_separation_init:nnnnn #1#2#3#4#5
602 {
603   \bool_if:NT \g__kernel_backend_header_bool
604   {
605     \exp_not:N \exp_args:Ne \__kernel_backend_first_shipout:n
606     {
607       \exp_not:N \__color_backend_separation_init_aux:nnnnnn
608       { \exp_not:N \int_use:N \g__color_model_int }
609       {#1} {#2} {#3} {#4} {#5}
610     }
611     \prop_gput:Nee \exp_not:N \g__color_backend_colorant_prop
612     { / \exp_not:N \str_convert_pdfname:n {#1} }
613     {
614       << ~
615       /setcolorspace ~ {} ~
616       >> ~ begin ~
617       color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
618       end
619     }
620   }

```

```

621 }
622 \cs_generate_variant:Nn \_color_backend_separation_init:nnnnn { nee }
623 \cs_new_protected:Npn \_color_backend_separation_init_aux:nnnnnn #1#2#3#4#5#6
624 {
625   \_kernel_backend_literal:e
626   {
627     !
628     TeXDict ~ begin ~
629     /color #1
630     {
631       [ ~
632       /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
633       [ ~ #3 ~ ] ~
634       {
635         \cs_if_exist_use:cF { \_color_backend_separation_init_ #3 :nnn }
636         { \_color_backend_separation_init:nnn }
637         {#4} {#5} {#6}
638       }
639       ] ~ setcolorspace
640     } ~ def ~
641     end
642   }
643 }
644 \cs_new:cpn { \_color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
645 { \_color_backend_separation_init_Device:Nn 4 {#3} }
646 \cs_new:cpn { \_color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
647 { \_color_backend_separation_init_Device:Nn 1 {#3} }
648 \cs_new:cpn { \_color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
649 { \_color_backend_separation_init_Device:Nn 2 {#3} }
650 \cs_new:Npn \_color_backend_separation_init_Device:Nn #1#2
651 {
652   #2 ~
653   \prg_replicate:nn {#1}
654   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
655   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
656 }

```

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.

```

657 \cs_new:Npn \_color_backend_separation_init:nnn #1#2#3
658 {
659   \exp_args:Ne \_color_backend_separation_init:nnnn
660   { \_color_backend_separation_init_count:n {#2} }
661   {#1} {#2} {#3}
662 }
663 \cs_new:Npn \_color_backend_separation_init_count:n #1
664 { \int_eval:n { 0 \_color_backend_separation_init_count:w #1 ~ \s_color_stop } }
665 \cs_new:Npn \_color_backend_separation_init_count:w #1 ~ #2 \s_color_stop
666 {
667   +1
668   \tl_if_blank:nF {#2}
669   { \_color_backend_separation_init_count:w #2 \s_color_stop }

```

670 }

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.

```

671 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
672 {
673   \__color_backend_separation_init:w #3 ~ \s__color_stop #4 ~ \s__color_stop
674   \prg_replicate:nn {#1}
675   {
676     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
677     \int_eval:n { 3 * #1 } ~ index ~ mul ~
678     2 ~ index ~ add ~
679     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
680   }
681   \int_step_function:nnnN {#1} { -1 } { 1 }
682   \__color_backend_separation_init:n
683   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
684   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
685   \tl_if_blank:nF {#2}
686   { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
687 }
688 \cs_new:Npn \__color_backend_separation_init:w
689 #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
690 {
691   #1 ~ #3 ~ 0 ~
692   \tl_if_blank:nF {#2}
693   { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }
694 }
695 \cs_new:Npn \__color_backend_separation_init:n #1
696 { \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.

```

697 \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s__color_stop
698 {
699   #2 ~ #3 ~
700   2 ~ index ~ 2 ~ index ~ lt ~
701   { ~ pop ~ exch ~ pop ~ } ~
702   { ~
703     2 ~ index ~ 1 ~ index ~ gt ~
704     { ~ exch ~ pop ~ exch ~ pop ~ } ~
705     { ~ pop ~ pop ~ } ~
706     ifelse ~
707   }

```

```

708     ifelse ~
709     #1 ~ 1 ~ roll ~
710     \tl_if_blank:nF {#4}
711     { \__color_backend_separation_init:nw {#1} #4 \s__color_stop }
712 }

```

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.

```

713 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
714 {
715     \__color_backend_separation_init:neenn
716     {#2}
717     {
718         /CIEBasedABC ~
719         << ~
720         /RangeABC ~ [ ~ \c__color_model_range_CIELAB_tl \c_space_tl ] ~
721         /DecodeABC ~
722         [ ~
723         { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
724         { ~ 500 ~ div ~ } ~ bind ~
725         { ~ 200 ~ div ~ } ~ bind ~
726         ] ~
727         /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
728         /DecodeLMN ~
729         [ ~
730         { ~
731             dup ~ 6 ~ 29 ~ div ~ ge ~
732             { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
733             { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
734             ifelse ~
735             0.9505 ~ mul ~
736             } ~ bind ~
737             { ~
738                 dup ~ 6 ~ 29 ~ div ~ ge ~
739                 { ~ dup ~ dup ~ mul ~ mul ~ } ~
740                 { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
741                 ifelse ~
742                 } ~ bind ~
743                 { ~
744                     dup ~ 6 ~ 29 ~ div ~ ge ~
745                     { ~ dup ~ dup ~ mul ~ mul ~ } ~
746                     { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
747                     ifelse ~
748                     1.0890 ~ mul ~
749                     } ~ bind
750                 ] ~
751             /WhitePoint ~
752             [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
753             >>
754         }
755         { \c__color_model_range_CIELAB_tl }
756         { 100 ~ 0 ~ 0 }
757         {#3}
758     }

```

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

`\_color_backend_devicen_init:nnn` Trivial as almost all of the work occurs in the shared code.

```

759 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
760 {
761   \_kernel_backend_literal:e
762   {
763     !
764     TeXDict ~ begin ~
765     /color \int_use:N \g\_color_model_int
766     {
767       [ ~
768         /DeviceN ~
769         [ ~ #1 ] ~
770         #2 ~
771         { ~ #3 ~ } ~
772         \_color_backend_devicen_colorants:n {#1}
773       ] ~ setcolorspace
774     } ~ def ~
775   }
776 }
777 }
```

(End of definition for `\_color_backend_devicen_init:nnn`.)

`\_color_backend_iccbased_init:nnn` No support at present.

```

778 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3 { }
779
780 </dvips>
781
782 <*dvisvgm>
```

(End of definition for `\_color_backend_iccbased_init:nnn`.)

`\_color_backend_select_separation:nnN` No support at present.

```

781 \cs_new_protected:Npn \_color_backend_select_separation:nnN #1#2#3 { }
782 \cs_new_eq:NN \_color_backend_select_devicen:nnN \_color_backend_select_separation:nnN
783
784 (End of definition for \_color_backend_select_separation:nnN and \_color_backend_select_
devicen:nnN.)
```

`\_color_backend_separation_init:nnnnn` No support at present.

```

783 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }
784 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnnnnn #1#2#3 { }
785
786 (End of definition for \_color_backend_separation_init:nnnnn and \_color_backend_separation_
init_CIELAB:nnn.)
```

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

```

785 \cs_new_protected:Npn \_color_backend_select_iccbased:nn #1#2
786 {
787   \_kernel_backend_literal_svg:e
788   {
789     <style>
790       @color-profile ~
```

```

791         \str_if_eq:nnTF {#2} { cmyk }
792         { device-cmyk }
793         { --color \int_use:N \g__color_model_int }
794         \c_space_tl
795     {
796         src:{"#1"}
797     }
798     </style>
799 }
800 }

```

(End of definition for __color_backend_select_iccbased:nn.)

```
801 </dvisvgm>
```

```
802 <*dviptdpmx | luatex | pdftex | xetex>
```

```

\__color_backend_select_separation:nnN
\__color_backend_select_devicen:nnN
\__color_backend_select_iccbased:nnN

```

```

803 <*dviptdpmx | xetex>
804 \cs_new_protected:Npn \__color_backend_select_separation:nnN #1#2#3
805 {
806     \tl_set:Nn #3 { ~ #1 ~ #2 }
807     \__kernel_backend_literal:e { pdf : bc ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
808 }
809 </dviptdpmx | xetex>
810 <*luatex | pdftex>
811 \cs_new_protected:Npn \__color_backend_select_separation:nnN #1#2
812 { \__color_backend_select:nnN { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
813 </luatex | pdftex>
814 \cs_new_eq:NN \__color_backend_select_devicen:nnN \__color_backend_select_separation:nnN
815 \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 dviptdpmx/X_YTeX, we skip this as at present it's handled by the backend.

```

816 \cs_new_protected:Npn \__color_backend_init_resource:n #1
817 {
818 <*luatex | pdftex>
819     \bool_lazy_and:nnT
820     { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
821     { \pdfmanagement_if_active_p: }
822     {
823         \use:e
824         {
825             \pdfmanagement_add:nnn
826             { Page / Resources / ColorSpace }
827             { #1 }
828             { \pdf_object_ref_last: }
829         }
830     }
831 </luatex | pdftex>
832 }

```

(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.

```

833 \cs_new_protected:Npn \__color_backend_separation_init:nnnnn #1#2#3#4#5
834 {
835   \pdf_object_unnamed_write:ne { dict }
836   {
837     /FunctionType ~ 2
838     /Domain ~ [0 ~ 1]
839     \tl_if_blank:nF {#3} { /Range ~ [#3] }
840     /C0 ~ [#4] ~
841     /C1 ~ [#5] /N ~ 1
842   }
843   \exp_args:Ne \__color_backend_separation_init:nn
844   { \str_convert_pdfname:n {#1} } {#2}
845   \__color_backend_init_resource:n { color \int_use:N \g__color_model_int }
846 }
847 \cs_new_protected:Npn \__color_backend_separation_init:nn #1#2
848 {
849   \use:e
850   {
851     \pdf_object_new:n { color \int_use:N \g__color_model_int }
852     \pdf_object_write:nnn { color \int_use:N \g__color_model_int } { array }
853     { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
854   }
855   \prop_gput:Nne \g__color_backend_colorant_prop { /#1 }
856   { \pdf_object_ref_last: }
857 }

```

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

```

858 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
859 {
860   \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
861   {
862     \pdf_object_new:n { __color_illuminant_CIELAB_ #1 }
863     \pdf_object_write:nne { __color_illuminant_CIELAB_ #1 } { array }
864     {
865       /Lab ~
866       <<
867       /WhitePoint ~
868       [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
869       /Range ~ [ \c__color_model_range_CIELAB_tl ]
870       >>
871     }
872   }
873   \__color_backend_separation_init:nnnnn
874   {#2}
875   { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
876   { \c__color_model_range_CIELAB_tl }
877   { 100 ~ 0 ~ 0 }
878   {#3}
879 }

```

(End of definition for `__color_backend_separation_init:nnnn`, `__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.

```

880 \cs_new_protected:Npn __color_backend_devicen_init:nnn #1#2#3
881 {
882   \pdf_object_unnamed_write:ne { stream }
883   {
884     {
885       /FunctionType ~ 4 ~
886       /Domain ~
887       [ ~
888         \prg_replicate:nn
889         { 0 __color_backend_devicen_init:w #1 ~ \s__color_stop }
890         { 0 ~ 1 ~ }
891       ] ~
892       /Range ~
893       [ ~
894         \str_case:nn {#2}
895         {
896           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
897           { /DeviceGray } { 0 ~ 1 }
898           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
899         } ~
900       ]
901     }
902     { {#3} }
903   }
904   \use:e
905   {
906     \pdf_object_new:n { color \int_use:N \g__color_model_int }
907     \pdf_object_write:nnn { color \int_use:N \g__color_model_int } { array }
908     {
909       /DeviceN ~
910       [ ~ #1 ] ~
911       #2 ~
912       \pdf_object_ref_last:
913       __color_backend_devicen_colorants:n {#1}
914     }
915   }
916   __color_backend_init_resource:n { color \int_use:N \g__color_model_int }
917 }
918 \cs_new:Npn __color_backend_devicen_init:w #1 ~ #2 \s__color_stop
919 {
920   + 1
921   \tl_if_blank:nF {#2}
922   { __color_backend_devicen_init:w #2 \s__color_stop }
923 }

```

(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.

```

924 \cs_new_protected:Npn __color_backend_iccbased_init:nnn #1#2#3

```

```

925 {
926   \pdf_object_if_exist:nF { __color_icc_ #1 }
927   {
928     \pdf_object_new:n { __color_icc_ #1 }
929     \pdf_object_write:nne { __color_icc_ #1 } { fstream }
930     {
931       {
932         /N ~ \exp_not:n { #2 } ~
933         \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
934       }
935       {#1}
936     }
937   }
938   \pdf_object_unnamed_write:ne { array }
939   { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
940   \__color_backend_init_resource:n { color \int_use:N \g__color_model_int }
941 }

```

(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.

```

942 \cs_new_protected:Npn \__color_backend_iccbased_device:nnn #1#2#3
943 {
944   \pdf_object_if_exist:nF { __color_icc_ #1 }
945   {
946     \pdf_object_new:n { __color_icc_ #1 }
947     \pdf_object_write:nnn { __color_icc_ #1 } { fstream }
948     {
949       { /N ~ #3 }
950       {#1}
951     }
952   }
953   \pdf_object_unnamed_write:ne { array }
954   { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
955   \__color_backend_init_resource:n { Default #2 }
956 }

```

(End of definition for __color_backend_iccbased_device:nnn.)

```

957 </dvipdfmx | luatex | pdftex | xetex>

```

3.4 Fill and stroke color

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

```

958 <*dvipdfmx | xetex>

```

```

\__color_backend_fill:nN
  \__color_backend_fill_cmyk:nN
  \__color_backend_fill_gray:nN
\__color_backend_fill_rgb:nN
\__color_backend_stroke:nN
  \__color_backend_stroke_cmyk:nN
  \__color_backend_stroke_gray:nN
  \__color_backend_stroke_rgb:nN

```

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

```

959 \cs_new_protected:Npn \__color_backend_fill:nN #1#2

```

```

960 { \_kernel_backend_literal:n { pdf : bc ~ fill ~ [ #1 ] } }
961 \cs_new_eq:NN \_color_backend_fill_cmyk:nN \_color_backend_fill:nN
962 \cs_new_eq:NN \_color_backend_fill_gray:nN \_color_backend_fill:nN
963 \cs_new_eq:NN \_color_backend_fill_rgb:nN \_color_backend_fill:nN
964 \cs_new_protected:Npn \_color_backend_stroke:nN #1#2
965 { \_kernel_backend_literal:n { pdf : bc ~ stroke ~ [ #1 ] } }
966 \cs_new_eq:NN \_color_backend_stroke_cmyk:nN \_color_backend_stroke:nN
967 \cs_new_eq:NN \_color_backend_stroke_gray:nN \_color_backend_stroke:nN
968 \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
969 \cs_new_protected:Npn \_color_backend_fill_separation:nnN #1#2#3
970 {
971   \_kernel_backend_literal:e
972   { pdf : bc ~ fill ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
973 }
974 \cs_new_protected:Npn \_color_backend_stroke_separation:nnN #1#2#3
975 {
976   \_kernel_backend_literal:e
977   { pdf : bc ~ stroke ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
978 }
979 \cs_new_eq:NN \_color_backend_fill_devicen:nnN \_color_backend_fill_separation:nnN
980 \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:
981 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
982 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:

```

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

```

983 </dviPDFmx | xetex>
984 <*luatex | pdftex>

```

Drawing (fill/stroke) color is handled in dvipdfmx/X_YT_EX in the same way as LuaT_EX/pdfT_EX.

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.

```

\_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
  \_color_backend_stroke:nN
985 \cs_new_protected:Npn \_color_backend_fill_cmyk:nN #1
986 { \_color_backend_fill:nN { #1 ~ k } }
987 \cs_new_protected:Npn \_color_backend_fill_gray:nN #1
988 { \_color_backend_fill:nN { #1 ~ g } }
989 \cs_new_protected:Npn \_color_backend_fill_rgb:nN #1
990 { \_color_backend_fill:nN { #1 ~ rg } }
991 \cs_new_protected:Npn \_color_backend_fill:nN #1#2
992 {
993   \tl_set:Nn \l_color_backend_fill_tl {#1}
994   \tl_set:Nn \l_color_backend_stroke_tl {#1 ~ \l_color_backend_stroke_tl }
995   \_kernel_color_backend_stack_push:nn \l_color_backend_stack_int {#2}
996 }
997 \cs_new_protected:Npn \_color_backend_stroke_cmyk:nN #1
998 { \_color_backend_stroke:nN { #1 ~ K } }

```

```

999 \cs_new_protected:Npn \__color_backend_stroke_gray:nN #1
1000 { \__color_backend_stroke:nN { #1 ~ G } }
1001 \cs_new_protected:Npn \__color_backend_stroke_rgb:nN #1
1002 { \__color_backend_stroke:nN { #1 ~ RG } }
1003 \cs_new_protected:Npn \__color_backend_stroke:nN #1#2
1004 {
1005   \tl_set:Nn \l__color_backend_stroke_tl {#1}
1006   \tl_set:Ne #2 { \l__color_backend_fill_tl \c_space_tl #1 }
1007   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int {#2}
1008 }

```

(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
1009 \cs_new_protected:Npn \__color_backend_fill_separation:nnN #1#2#3
1010 { \__color_backend_fill:nN { /#1 ~ cs ~ #2 ~ scn } }
1011 \cs_new_protected:Npn \__color_backend_stroke_separation:nnN #1#2#3
1012 { \__color_backend_stroke:nN { /#1 ~ CS ~ #2 ~ SCN } }
1013 \cs_new_eq:NN \__color_backend_fill_devicen:nnN \__color_backend_fill_separation:nnN
1014 \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:
1015 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1016 \cs_new_eq:NN \__color_backend_stroke_reset: \__color_backend_reset:

```

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

```

1017 </luatex | pdftex>
1018 <*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
1019 \cs_new_protected:Npn \__color_backend_fill_cmyk:nN #1
1020 { \__color_backend_fill:nN { cmyk ~ #1 } }
1021 \cs_new_protected:Npn \__color_backend_fill_gray:nN #1
1022 { \__color_backend_fill:nN { gray ~ #1 } }
1023 \cs_new_protected:Npn \__color_backend_fill_rgb:nN #1
1024 { \__color_backend_fill:nN { rgb ~ #1 } }
1025 \cs_new_protected:Npn \__color_backend_fill:nN #1#2
1026 {
1027   \tl_set:Nn #2 {#1}
1028   \__kernel_backend_literal:n { color~push~ #1 }
1029 }
1030 \cs_new_protected:Npn \__color_backend_stroke_cmyk:nN #1#2
1031 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmycolor } def } }
1032 \cs_new_protected:Npn \__color_backend_stroke_gray:nN #1#2
1033 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1034 \cs_new_protected:Npn \__color_backend_stroke_rgb:nN #1#2
1035 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }

```

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.

(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
1036 \cs_new_protected:Npn \__color_backend_fill_separation:nnN #1#2#3
1037 { \__color_backend_fill:n { separation ~ #1 ~ #2 } }
1038 \cs_new_protected:Npn \__color_backend_stroke_separation:nnN #1#2#3
1039 { \__kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1040 \cs_new_eq:NN \__color_backend_fill_devicen:nnN \__color_backend_fill_separation:nnN
1041 \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:
1042 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1043 \cs_new_protected:Npn \__color_backend_stroke_reset: { }

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

1044 </dvips>
1045 <*dvisvgm>

\__color_backend_fill_cmyk:nN
\__color_backend_fill_gray:nN
\__color_backend_fill_rgb:nN
\__color_backend_fill:nN
240 Fill color here is the same as general color.
1046 \cs_new_protected:Npn \__color_backend_fill_cmyk:nN #1
1047 { \__color_backend_fill:nN { cmyk ~ #1 } }
1048 \cs_new_protected:Npn \__color_backend_fill_gray:nN #1
1049 { \__color_backend_fill:nN { gray ~ #1 } }
1050 \cs_new_protected:Npn \__color_backend_fill_rgb:nN #1
1051 { \__color_backend_fill:nN { rgb ~ #1 } }
1052 \cs_new_protected:Npn \__color_backend_fill:nN #1#2
1053 {
1054   \tl_set:Nn #2 {#1}
1055   \__kernel_backend_literal:n { color~push~ #1 }
1056 }

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

\__color_backend_stroke_cmyk:nN
\__color_backend_stroke_gray:nN
\__color_backend_stroke_gray_aux:n
\__color_backend_stroke_rgb:nN
\__color_backend_stroke_rgb:w
\__color_backend:nnn
241 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.
1057 \cs_new_protected:Npn \__color_backend_stroke_cmyk:nN #1#2
1058 {
1059   \__color_backend_fill_cmyk:n {#1}
1060   \__kernel_backend_scope:n { stroke = "{?color}" }
1061   \__color_backend_reset:
1062 }
1063 \cs_new_protected:Npn \__color_backend_stroke_gray:nN #1#2
1064 {
1065   \use:e
1066   {
1067     \__color_backend_stroke_gray_aux:n
1068     { \fp_eval:n { 100 * (#1) } }
1069   }
1070 }
1071 \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1072 { \__color_backend:nnn {#1} {#1} {#1} }
1073 \cs_new_protected:Npn \__color_backend_stroke_rgb:nN #1#2

```

```

1074 { \_color_backend_rgb:w #1 \s\_color_stop }
1075 \cs_new_protected:Npn \_color_backend_stroke_rgb:w
1076 #1 ~ #2 ~ #3 \s\_color_stop
1077 {
1078   \use:e
1079   {
1080     \_color_backend:nnn
1081     { \fp_eval:n { 100 * (#1) } }
1082     { \fp_eval:n { 100 * (#2) } }
1083     { \fp_eval:n { 100 * (#3) } }
1084   }
1085 }
1086 \cs_new_protected:Npe \_color_backend:nnn #1#2#3
1087 {
1088   \_kernel_backend_scope:n
1089   {
1090     stroke =
1091     "
1092     rgb
1093     (
1094       #1 \c_percent_str ,
1095       #2 \c_percent_str ,
1096       #3 \c_percent_str
1097     )
1098     "
1099   }
1100 }

```

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

```

\_color_backend_fill_separation:nnN
\_color_backend_stroke_separation:nnN
\_color_backend_fill_devicen:nnN
\_color_backend_stroke_devicen:nnN

```

At present, these are no-ops.

```

1101 \cs_new_protected:Npn \_color_backend_fill_separation:nnN #1#2#3 { }
1102 \cs_new_protected:Npn \_color_backend_stroke_separation:nnN #1#2#3 { }
1103 \cs_new_eq:NN \_color_backend_fill_devicen:nnN \_color_backend_fill_separation:nnN
1104 \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:

```

```

1105 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
1106 \cs_new_protected:Npn \_color_backend_stroke_reset: { }

```

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

```

\_color_backend_devicen_init:nnn
\_color_backend_iccbased_init:nnn

```

No support at present.

```

1107 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3 { }
1108 \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.)

```

1109 </dvisvgm>
1110 </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.

```

1111  $\langle$ *lua $\rangle$ 
1112 local l = lpeg
1113 local spaces = l.P' '^0
1114 local digit16 = l.R('09', 'af', 'AF')
1115
1116 local octet = digit16 * digit16 / function(s)
1117   return string.format('%.3g ', tonumber(s, 16) / 255)
1118 end
1119
1120 if luaotfload and luaotfload.set_transparent_colorstack then
1121   local htmlcolor = l.Cs(octet * octet * octet * -1 * l.Cc'rg')
1122   local color_export = {
1123     token.create'tex_endlocalcontrol:D',
1124     token.create'tex_hpack:D',
1125     token.new(0, 1),
1126     token.create'color_export:nnN',
1127     token.new(0, 1),
1128     '',
1129     token.new(0, 2),
1130     token.new(0, 1),
1131     'backend',
1132     token.new(0, 2),
1133     token.create'l__color_tmp_tl',
1134     token.create'exp_after:wN',
1135     token.create'__color_select_aux:nnN',
1136     token.create'l__color_tmp_tl',
1137     token.create'l__color_tmp_tl',
1138     token.new(0, 2),
1139   }
1140   local group_end = token.create'group_end:'
1141   local value = (1 - l.P'}')^0
1142   luatexbase.add_to_callback('luaotfload.parse_color', function (value)
1143     % Also allow HTML colors to preserve compatibility
1144     local html = htmlcolor:match(value)
1145     if html then return html end
1146
1147     % If no l3color named color with this name is known, check for defined xcolor colors
1148     local l3color_prop = token.get_macro(string.format('l__color_named_%s_prop', value))
1149     if l3color_prop == nil or l3color_prop == '' then
1150       local legacy_color_macro = token.create(string.format('\\color@%s', value))
1151       if legacy_color_macro.cmdname ~= 'undefined_cs' then
1152         token.put_next(legacy_color_macro)
1153         return token.scan_argument()
1154       end
1155     end
1156
1157     tex.runtoks(function()
1158       token.get_next()
1159       color_export[6] = value
1160       tex.sprint(-2, color_export)

```

```

1161     end)
1162     local list = token.scan_list()
1163     if not list.head or list.head.next
1164         or list.head.subtype ~= node.subtype'pdf_colorstack' then
1165         error'Unexpected backend behavior'
1166     end
1167     local cmd = list.head.data
1168     node.free(list)
1169     return cmd
1170 end, 'l3color')
1171 end
1172 </lua>
1173 <*luatex>
1174 <*package>
1175 \lua_load_module:n {l3backend-luatex}
1176 </package>
1177 </luatex>

```

4 l3backend-draw implementation

```

1178 <*package>
1179 <@@=draw>

```

4.1 dvips backend

```

1180 <*dvips>

```

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

```

1181 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_postscript:n
1182 \cs_generate_variant:Nn \__draw_backend_literal:n { e }

```

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

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

```

1183 \cs_new_protected:Npn \__draw_backend_begin:
1184 {
1185     \__draw_backend_literal:n { [begin] }
1186     \__draw_backend_literal:n { /draw.x~currentpoint~/draw.y~exch~def~def }
1187     \__draw_backend_literal:n { @beginspecial }
1188 }
1189 \cs_new_protected:Npn \__draw_backend_end:
1190 {
1191     \__draw_backend_literal:n { @endspecial }
1192     \__draw_backend_literal:n { [end] }
1193 }

```

(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.

```

1194 \cs_new_protected:Npn \_draw_backend_scope_begin:
1195 { \_draw_backend_literal:n { save } }
1196 \cs_new_protected:Npn \_draw_backend_scope_end:
1197 { \_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:nmmn` any variable values are forced to literals before any possible caching. There is no native
`\_draw_backend_curveto:nnmmnn` rectangular path command (without also clipping, filling or stroking), so that task is
done using a small amount of PostScript.

```

1198 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1199 {
1200   \_draw_backend_literal:e
1201   {
1202     \dim_to_decimal_in_bp:n {#1} ~
1203     \dim_to_decimal_in_bp:n {#2} ~ moveto
1204   }
1205 }
1206 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1207 {
1208   \_draw_backend_literal:e
1209   {
1210     \dim_to_decimal_in_bp:n {#1} ~
1211     \dim_to_decimal_in_bp:n {#2} ~ lineto
1212   }
1213 }
1214 \cs_new_protected:Npn \_draw_backend_rectangle:nmmn #1#2#3#4
1215 {
1216   \_draw_backend_literal:e
1217   {
1218     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1219     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1220     moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1221   }
1222 }
1223 \cs_new_protected:Npn \_draw_backend_curveto:nnmmnn #1#2#3#4#5#6
1224 {
1225   \_draw_backend_literal:e
1226   {
1227     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1228     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1229     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1230     curveto
1231   }
1232 }

```

(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`

```

1233 \cs_new_protected:Npn \_draw_backend_evenodd_rule:

```

```

1234 { \bool_gset_true:N \g__draw_draw_eor_bool }
1235 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1236 { \bool_gset_false:N \g__draw_draw_eor_bool }
1237 \bool_new:N \g__draw_draw_eor_bool

```

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

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

```

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

```

```

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

```

(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_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1318 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1319 {
1320   \__draw_backend_literal:e
1321   {
1322     [
1323       \exp_args:Nf \use:n
1324       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1325     ] ~
1326     \dim_to_decimal_in_bp:n {#2} ~ setdash
1327   }

```

```

1328 }
1329 \cs_new:Npn \__draw_backend_dash:n #1
1330 { ~ \dim_to_decimal_in_bp:n {#1} }
1331 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1332 {
1333   \__draw_backend_literal:e
1334   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1335 }
1336 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1337 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1338 \cs_new_protected:Npn \__draw_backend_cap_but:
1339 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1340 \cs_new_protected:Npn \__draw_backend_cap_round:
1341 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1342 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1343 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1344 \cs_new_protected:Npn \__draw_backend_join_miter:
1345 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1346 \cs_new_protected:Npn \__draw_backend_join_round:
1347 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1348 \cs_new_protected:Npn \__draw_backend_join_bevel:
1349 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

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

__draw_backend_transform:nnnn
__draw_backend_shift:nn

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

```

1350 \cs_new_protected:Npn \__draw_backend_transform:nnnn #1#2#3#4
1351 {
1352   \__draw_backend_literal:n
1353   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1354 }
1355 \cs_new_protected:Npn \__draw_backend_shift:nn #1#2
1356 {
1357   \__draw_backend_literal:n
1358   { [ 1 ~ 0 ~ 0 ~ 1 ~ #1 ~ #2 ] ~ concat }
1359 }

```

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

__draw_backend_box_use:Nnnnn

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

```

1360 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5

```

```

1361 {
1362   \_draw_backend_literal:n { save }
1363   \_draw_backend_literal:n { 72~Resolution~div~72~VResolution~div~neg~scale }
1364   \_draw_backend_literal:n { magscale { 1~DVImag~div~dup~scale } if }
1365   \_draw_backend_literal:n { draw.x~neg~draw.y~neg~translate }
1366   \_draw_backend_literal:n { [end] }
1367   \_draw_backend_literal:n { [begin] }
1368   \_draw_backend_literal:n { save }
1369   \_draw_backend_literal:n { currentpoint }
1370   \_draw_backend_literal:n { currentpoint~translate }
1371   \_draw_backend_transform:nnnn { 1 } { 0 } { 0 } { -1 }
1372   \_draw_backend_transform:nnnn {#2} {#3} {#4} {#5}
1373   \_draw_backend_transform:nnnn { 1 } { 0 } { 0 } { -1 }
1374   \_draw_backend_literal:n { neg~exch~neg~exch~translate }
1375   \_draw_backend_literal:n { [end] }
1376   \hbox_overlap_right:n { \box_use:N #1 }
1377   \_draw_backend_literal:n { [begin] }
1378   \_draw_backend_literal:n { restore }
1379   \_draw_backend_literal:n { [end] }
1380   \_draw_backend_literal:n { [begin] }
1381   \_draw_backend_literal:n { restore }
1382 }

```

(End of definition for _draw_backend_box_use:Nnnnn.)

```

1383 </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.

```

1384 <{*dvipdfmx | luatex | pdftex | xetex}

```

4.2.1 Drawing

<pre> _draw_backend_literal:n _draw_backend_literal:e </pre>	Pass data through using a dedicated interface. <pre> 1385 \cs_new_eq:NN _draw_backend_literal:n _kernel_backend_literal_pdf:n 1386 \cs_new_eq:NN _draw_backend_literal:e _kernel_backend_literal_pdf:e </pre>
--	--

(End of definition for _draw_backend_literal:n.)

<pre> _draw_backend_begin: _draw_backend_end: </pre>	No special requirements here, so simply set up a drawing scope. <pre> 1387 \cs_new_protected:Npn _draw_backend_begin: 1388 { _draw_backend_scope_begin: } 1389 \cs_new_protected:Npn _draw_backend_end: 1390 { _draw_backend_scope_end: } </pre>
--	---

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

<pre> _draw_backend_scope_begin: _draw_backend_scope_end: </pre>	Use the backend-level scope mechanisms. <pre> 1391 \cs_new_eq:NN _draw_backend_scope_begin: _kernel_backend_scope_begin: 1392 \cs_new_eq:NN _draw_backend_scope_end: _kernel_backend_scope_end: </pre>
--	---

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

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

`\__draw_backend_lineto:nn`

```

\__draw_backend_moveto:nnnnnn 1393 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
\__draw_backend_rectangle:nnnn 1394 {
1395   \__draw_backend_literal:e
1396   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1397 }
1398 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1399 {
1400   \__draw_backend_literal:e
1401   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1402 }
1403 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1404 {
1405   \__draw_backend_literal:e
1406   {
1407     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1408     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1409     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1410     c
1411   }
1412 }
1413 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1414 {
1415   \__draw_backend_literal:e
1416   {
1417     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1418     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1419     re
1420   }
1421 }

```

(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`

```

1422 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1423 { \bool_gset_true:N \g__draw_draw_eor_bool }
1424 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1425 { \bool_gset_false:N \g__draw_draw_eor_bool }
1426 \bool_new:N \g__draw_draw_eor_bool

```

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

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

`\__draw_backend_stroke:`

`\__draw_backend_closestroke:`

`\__draw_backend_fill:`

`\__draw_backend_fillstroke:`

`\__draw_backend_clip:`

`\__draw_backend_discardpath:`

```

1427 \cs_new_protected:Npn \__draw_backend_closepath:
1428 { \__draw_backend_literal:n { h } }
1429 \cs_new_protected:Npn \__draw_backend_stroke:
1430 { \__draw_backend_literal:n { S } }
1431 \cs_new_protected:Npn \__draw_backend_closestroke:
1432 { \__draw_backend_literal:n { s } }
1433 \cs_new_protected:Npn \__draw_backend_fill:
1434 {
1435   \__draw_backend_literal:e

```

```

1436     { f \bool_if:NT \g__draw_draw_eor_bool * }
1437   }
1438   \cs_new_protected:Npn \__draw_backend_fillstroke:
1439   {
1440     \__draw_backend_literal:e
1441     { B \bool_if:NT \g__draw_draw_eor_bool * }
1442   }
1443   \cs_new_protected:Npn \__draw_backend_clip:
1444   {
1445     \__draw_backend_literal:e
1446     { W \bool_if:NT \g__draw_draw_eor_bool * }
1447   }
1448   \cs_new_protected:Npn \__draw_backend_discardpath:
1449   { \__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.

```

1450 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1451 {
1452   \__draw_backend_literal:e
1453   {
1454     [
1455       \exp_args:Nf \use:n
1456       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1457     ] ~
1458     \dim_to_decimal_in_bp:n {#2} ~ d
1459   }
1460 }
1461 \cs_new:Npn \__draw_backend_dash:n #1
1462 { ~ \dim_to_decimal_in_bp:n {#1} }
1463 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1464 {
1465   \__draw_backend_literal:e
1466   { \dim_to_decimal_in_bp:n {#1} ~ w }
1467 }
1468 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1469 { \__draw_backend_literal:e { #1 ~ M } }
1470 \cs_new_protected:Npn \__draw_backend_cap_but:
1471 { \__draw_backend_literal:n { 0 ~ J } }
1472 \cs_new_protected:Npn \__draw_backend_cap_round:
1473 { \__draw_backend_literal:n { 1 ~ J } }
1474 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1475 { \__draw_backend_literal:n { 2 ~ J } }
1476 \cs_new_protected:Npn \__draw_backend_join_miter:
1477 { \__draw_backend_literal:n { 0 ~ j } }
1478 \cs_new_protected:Npn \__draw_backend_join_round:
1479 { \__draw_backend_literal:n { 1 ~ j } }
1480 \cs_new_protected:Npn \__draw_backend_join_bevel:
1481 { \__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/XYTeX`, 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/XYTeX`, 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.

```

1482 \cs_new_protected:Npn \__draw_backend_transform:nnnn #1#2#3#4
1483 {
1484   <*luatex | pdftex>
1485     \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1486   </luatex | pdftex>
1487   <*dvipdfmx | xetex>
1488     \str_if_eq:nnTF { #2 ~ #3 } { 0 ~ 0 }
1489     {
1490       \__kernel_backend_literal:n { x:rotate~0 }
1491       \__kernel_backend_literal:n { x:scale~#1~#4 }
1492       \__kernel_backend_literal:n { x:rotate~0 }
1493     }
1494     {
1495       \__draw_backend_transform_decompose:nnnnN {#1} {#2} {#3} {#4}
1496       \__draw_backend_transform_aux:nnnn
1497     }
1498   </dvipdfmx | xetex>
1499 }
1500 <*dvipdfmx | xetex>
1501 \cs_new_protected:Npn \__draw_backend_transform_aux:nnnn #1#2#3#4
1502 {
1503   \__kernel_backend_literal:e
1504   {
1505     x:rotate~
1506     \fp_compare:nNnTF {#1} = \c_zero_fp
1507       { 0 }
1508       { \fp_eval:n { round ( -#1 , 5 ) } }
1509   }
1510   \__kernel_backend_literal:e
1511   {
1512     x:scale~
1513     \fp_eval:n { round ( #2 , 5 ) } ~
1514     \fp_eval:n { round ( #3 , 5 ) }
1515   }
1516   \__kernel_backend_literal:e
1517   {
1518     x:rotate~
1519     \fp_compare:nNnTF {#4} = \c_zero_fp
1520       { 0 }
1521       { \fp_eval:n { round ( -#4 , 5 ) } }
1522   }
1523 }
1524 </dvipdfmx | xetex>

```

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

```

1525 \cs_new_protected:Npn \__draw_backend_shift:nn #1#2

```

```

1526 {
1527   \_draw_backend_literal:n
1528   { 1 ~ 0 ~ 0 ~ 1 ~ #1 ~ #2 ~ cm }
1529 }

```

(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.

```

1530 <*dvipdfmx | xetex>
1531 \cs_new_protected:Npn \_draw_backend_transform_decompose:nnnnN #1#2#3#4#5
1532 {
1533   \use:e
1534   {
1535     \_draw_backend_transform_decompose_auxi:nnnnN
1536     { \fp_eval:n { (#1 + #4) / 2 } }
1537     { \fp_eval:n { (#1 - #4) / 2 } }
1538     { \fp_eval:n { (#3 + #2) / 2 } }
1539     { \fp_eval:n { (#3 - #2) / 2 } }
1540   }
1541   #5
1542 }
1543 \cs_new_protected:Npn \_draw_backend_transform_decompose_auxi:nnnnN #1#2#3#4#5
1544 {
1545   \use:e
1546   {
1547     \_draw_backend_transform_decompose_auxii:nnnnN

```

```

1548         { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1549         { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1550         { \fp_eval:n { atand ( #3 , #2 ) } }
1551         { \fp_eval:n { atand ( #4 , #1 ) } }
1552     }
1553     #5
1554 }
1555 \cs_new_protected:Npn \__draw_backend_transform_decompose_auxiii:nnnnN #1#2#3#4#5
1556 {
1557     \use:e
1558     {
1559         \__draw_backend_transform_decompose_auxiii:nnnnN
1560         { \fp_eval:n { ( #4 - #3 ) / 2 } }
1561         { \fp_eval:n { ( #1 + #2 ) / 2 } }
1562         { \fp_eval:n { ( #1 - #2 ) / 2 } }
1563         { \fp_eval:n { ( #4 + #3 ) / 2 } }
1564     }
1565     #5
1566 }
1567 \cs_new_protected:Npn \__draw_backend_transform_decompose_auxiii:nnnnN #1#2#3#4#5
1568 {
1569     \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1570     { #5 {#1} {#2} {#3} {#4} }
1571     { #5 {#1} {#3} {#2} {#4} }
1572 }
1573 </dviPDFmx | 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.

```

1574 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1575 {
1576     \__kernel_backend_scope_begin:
1577     < *luatex | pdftex >
1578     \__kernel_backend_matrix:n { #2 ~ #3 ~ #4 ~ #5 }
1579     < /luatex | pdftex >
1580     < *dviPDFmx | xetex >
1581     \__kernel_backend_literal:n
1582     { pdf:btrans-matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1583     < /dviPDFmx | xetex >
1584     \hbox_overlap_right:n { \box_use:N #1 }
1585     < *dviPDFmx | xetex >
1586     \__kernel_backend_literal:n { pdf:etrans }
1587     < /dviPDFmx | xetex >
1588     \__kernel_backend_scope_end:
1589 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1590 < /dviPDFmx | luatex | pdftex | xetex >

```

4.3 dvisvgm backend

1591 $\langle *dvisvgm \rangle$

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

`\_draw_backend_literal:e` 1592 `\cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_svg:n`
 1593 `\cs_generate_variant:Nn \_draw_backend_literal:n { e }`

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

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

`\_draw_backend_scope_end:` 1594 `\cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:`
 1595 `\cs_new_eq:NN \_draw_backend_scope_end: \_kernel_backend_scope_end:`

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

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

1596 `\cs_new_protected:Npn \_draw_backend_begin:`
 1597 `{`
 1598 `\_kernel_backend_scope_begin:`
 1599 `\_kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }`
 1600 `}`
 1601 `\cs_new_eq:NN \_draw_backend_end: \_kernel_backend_scope_end:`

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

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

`\g__draw_backend_path_tl` 1602 `\cs_new_protected:Npn \_draw_backend_moveto:nn #1#2`
 1603 `{`
 1604 `\_draw_backend_add_to_path:n`
 1605 `{ M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }`
 1606 `}`
 1607 `\cs_new_protected:Npn \_draw_backend_lineto:nn #1#2`
 1608 `{`
 1609 `\_draw_backend_add_to_path:n`
 1610 `{ L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }`
 1611 `}`
 1612 `\cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4`
 1613 `{`
 1614 `\_draw_backend_add_to_path:n`
 1615 `{`
 1616 `M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}`
 1617 `h ~ \dim_to_decimal:n {#3} ~`
 1618 `v ~ \dim_to_decimal:n {#4} ~`
 1619 `h ~ \dim_to_decimal:n { -#3 } ~`
 1620 `Z`
 1621 `}`
 1622 `}`
 1623 `\cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6`
 1624 `{`

```

1625 \__draw_backend_add_to_path:n
1626 {
1627   C ~
1628   \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1629   \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1630   \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1631 }
1632 }
1633 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1634 {
1635   \tl_gset:Nx \g__draw_backend_path_tl
1636   {
1637     \g__draw_backend_path_tl
1638     \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1639     #1
1640   }
1641 }
1642 \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:
1643 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1644 { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1645 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1646 { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

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

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

```

\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int
1647 \cs_new_protected:Npn \__draw_backend_closepath:
1648 { \__draw_backend_add_to_path:n { Z } }
1649 \cs_new_protected:Npn \__draw_backend_path:n #1
1650 {
1651   \bool_if:NTF \g__draw_draw_clip_bool
1652   {
1653     \int_gincr:N \g__kernel_clip_path_int
1654     \__draw_backend_literal:e
1655     {
1656       < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1657       { ?nl }
1658       <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1659       < /clipPath > { ? nl }
1660       <
1661       use~xlink:href =
1662       "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1663       #1
1664     } />
1665   }
1666   \__kernel_backend_scope:e

```

```

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

```

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

```

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

```

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

```

1715 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2

```

```

1716 {
1717   \use:e
1718   {
1719     \__draw_backend_dash_aux:nn
1720     { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1721     { \dim_to_decimal:n {#2} }
1722   }
1723 }
1724 \cs_new:Npn \__draw_backend_dash:n #1
1725 { , \dim_to_decimal_in_bp:n {#1} }
1726 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1727 {
1728   \__kernel_backend_scope:e
1729   {
1730     stroke-dasharray =
1731     "
1732     \tl_if_empty:nTF {#1}
1733     { none }
1734     { \use_none:n #1 }
1735     " ~
1736     stroke-offset=" #2 "
1737   }
1738 }
1739 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1740 { \__kernel_backend_scope:e { stroke-width=" \dim_to_decimal:n {#1} " } }
1741 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1742 { \__kernel_backend_scope:e { stroke-miterlimit=" #1 " } }
1743 \cs_new_protected:Npn \__draw_backend_cap_but:
1744 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1745 \cs_new_protected:Npn \__draw_backend_cap_round:
1746 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1747 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1748 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1749 \cs_new_protected:Npn \__draw_backend_join_miter:
1750 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1751 \cs_new_protected:Npn \__draw_backend_join_round:
1752 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1753 \cs_new_protected:Npn \__draw_backend_join_bevel:
1754 { \__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.

```

1755 \cs_new_protected:Npn \__draw_backend_transform:nnnn #1#2#3#4
1756 {
1757   \__kernel_backend_scope:n
1758   {
1759     transform =
1760     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1761   }
1762 }
1763 \cs_new_protected:Npn \__draw_backend_shift:nn #1#2
1764 {

```

```

1765 \__kernel_backend_scope:n
1766 {
1767     transform =
1768     " matrix ( 1 , 0 , 0 , 1 , #1pt , #2pt ) "
1769 }
1770 }

```

(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.

```

1771 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1772 {
1773     \__kernel_backend_scope_begin:
1774     \__draw_backend_transform:nnnn {#2} {#3} {#4} {#5}
1775     \__kernel_backend_literal_svg:n
1776     {
1777         < g~
1778         stroke="none"~
1779         transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1780         >
1781     }
1782     \box_set_wd:Nn #1 { Opt }
1783     \box_set_ht:Nn #1 { Opt }
1784     \box_set_dp:Nn #1 { Opt }
1785     \box_use:N #1
1786     \__kernel_backend_literal_svg:n { </g> }
1787     \__kernel_backend_scope_end:
1788 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1789 </divisvgn>
1790 </package>

```

5 l3backend-graphics implementation

```

1791 <*package>
1792 <@@=graphics>

```

5.1 dvips backend

```

1793 <*dvips>

```

\l_graphics_search_ext_seq

```

1794 \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.

```

1795 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1796 \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.

```

1797 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1798 {
1799   \__kernel_backend_literal:e
1800   {
1801     PSfile = #1 \c_space_tl
1802     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1803     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1804     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1805     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1806   }
1807 }
1808 \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`

```

1809 \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n

```

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

```

1810 </dvips>

```

5.2 LuaTeX and pdfTeX backends

```

1811 <*luatex | pdftex>

```

`\l__graphics_search_ext_seq`

```

1812 \seq_set_from_clist:Nn \l__graphics_search_ext_seq
1813 { .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.

```

1814 \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.

```

1815 \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.

```

1816 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1817 {

```

```

1818 \int_zero:N \l__graphics_page_int
1819 \tl_clear:N \l__graphics_pagebox_tl
1820 \bool_set_false:N \l__graphics_transgroup_bool
1821 \tl_set:Ne \l__graphics_attr_tl
1822 {
1823   \tl_if_empty:NF \l__graphics_decodearray_str
1824   { :D \l__graphics_decodearray_str }
1825   \bool_if:NT \l__graphics_interpolate_bool
1826   { :I }
1827   \str_if_empty:NF \l__graphics_pdf_str
1828   { :X \l__graphics_pdf_str }
1829 }
1830 \__graphics_backend_getbb_auxi:n {#1}
1831 }
1832 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1833 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1834 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1835 {
1836   \tl_clear:N \l__graphics_decodearray_str
1837   \bool_set_true:N \l__graphics_transgroup_bool
1838   \bool_set_false:N \l__graphics_interpolate_bool
1839   \tl_set:Ne \l__graphics_attr_tl
1840   {
1841     : \l__graphics_pagebox_tl
1842     \int_compare:nNnT \l__graphics_page_int > 1
1843     { :P \int_use:N \l__graphics_page_int }
1844     \str_if_empty:NF \l__graphics_pdf_str
1845     { :X \l__graphics_pdf_str }
1846   }
1847   \__graphics_backend_getbb_auxi:n {#1}
1848 }
1849 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1850 {
1851   \__graphics_bb_restore:eF { #1 \l__graphics_attr_tl }
1852   { \__graphics_backend_getbb_auxii:n {#1} }
1853 }

```

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.

```

1854 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1855 {
1856   \exp_args:Ne \__graphics_backend_getbb_auxiii:n
1857   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1858   \int_const:cn { c__graphics_ #1 \l__graphics_attr_tl _int }
1859   { \tex_the:D \tex_pdflastximage:D }
1860   \__graphics_bb_save:e { #1 \l__graphics_attr_tl }
1861 }
1862 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1863 {
1864   \tex_immediate:D \tex_pdfximage:D
1865   \bool_lazy_any:nT

```

```

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

```

1895   \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1896   {
1897     \tex_pdfrefximage:D
1898     \int_use:c { c__graphics_ #1 \l__graphics_attr_tl _int }
1899   }
1900   \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1901   \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1902   \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

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

```

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

```

EPS graphics may be included in LuaTeX/pdfTeX by conversion to PDF: this requires restricted shell escape. Modeled on the `epstopdf` L^AT_EX 2_ε package, but simplified, conversion takes place here if we have shell access.

```

1903   \sys_if_shell:T
1904   {
1905     \str_new:N \l__graphics_backend_dir_str
1906     \str_new:N \l__graphics_backend_name_str
1907     \str_new:N \l__graphics_backend_ext_str

```

```

1908 \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1909 {
1910   \file_parse_full_name:nNNN {#1}
1911   \l__graphics_backend_dir_str
1912   \l__graphics_backend_name_str
1913   \l__graphics_backend_ext_str
1914   \exp_args:Ne \__graphics_backend_getbb_eps:nn
1915   {
1916     \exp_args:Ne \__kernel_file_name_quote:n
1917     {
1918       \l__graphics_backend_name_str
1919       - \str_tail:N \l__graphics_backend_ext_str
1920       -converted-to.pdf
1921     }
1922   }
1923   {#1}
1924 }
1925 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_backend_getbb_eps:n
1926 \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1927 {
1928   \file_compare_timestamp:nNnT {#2} > {#1}
1929   {
1930     \sys_shell_now:n
1931     { repstopdf ~ #2 ~ #1 }
1932   }
1933   \tl_set:Nn \l__graphics_final_name_str {#1}
1934   \__graphics_backend_getbb_pdf:n {#1}
1935 }
1936 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1937 {
1938   \file_parse_full_name:nNNN {#1}
1939   \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1940   \exp_args:Ne \__graphics_backend_include_pdf:n
1941   {
1942     \exp_args:Ne \__kernel_file_name_quote:n
1943     {
1944       \l__graphics_backend_name_str
1945       - \str_tail:N \l__graphics_backend_ext_str
1946       -converted-to.pdf
1947     }
1948   }
1949 }
1950 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1951 }

```

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

__graphics_backend_get_pagecount:n Simply load and store.

```

1952 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1953 {
1954   \tex_pdfximage:D {#1}
1955   \int_const:cn { c__graphics_ #1 _pages_int }
1956   { \int_use:N \tex_pdflastximagepages:D }
1957 }

```

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

1958 `</luatex | pdftex>`

5.3 dvipdfmx backend

1959 `<*dvipdfmx | xetex>`

`\l_graphics_search_ext_seq`

1960 `\seq_set_from_clist:Nn \l_graphics_search_ext_seq`
1961 `{ .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }`

(End of definition for `\l_graphics_search_ext_seq`.)

`\__graphics_backend_getbb_eps:n`
`\__graphics_backend_getbb_ps:n`
`\__graphics_backend_getbb_jpg:n`
`\__graphics_backend_getbb_jpeg:n`
`\__graphics_backend_getbb_pdf:n`
`\__graphics_backend_getbb_png:n`
`\__graphics_backend_getbb_bmp:n`

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

1962 `\cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n`
1963 `\cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n`
1964 `<*dvipdfmx>`
1965 `\cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1`
1966 `{`
1967 `\int_zero:N \l__graphics_page_int`
1968 `\tl_clear:N \l__graphics_pagebox_tl`
1969 `\__graphics_extract_bb:n {#1}`
1970 `}`
1971 `\cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n`
1972 `\cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n`
1973 `\cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n`
1974 `\cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1`
1975 `{`
1976 `\tl_clear:N \l__graphics_decodearray_str`
1977 `\bool_set_false:N \l__graphics_interpolate_bool`
1978 `\__graphics_extract_bb:n {#1}`
1979 `}`
1980 `</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.

1981 `\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.

1982 `\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.

1983 `\cs_new_protected:Npn \__graphics_backend_include_eps:n #1`
1984 `{`
1985 `\__kernel_backend_literal:e`
1986 `}`

```

1987     PSfile = #1 \c_space_tl
1988     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1989     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1990     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1991     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1992   }
1993 }
1994 \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.

```

1995 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1996 {
1997   \bool_set_false:N \l__graphics_transgroup_bool
1998   \__graphics_backend_include_auxi:n {#1}
1999 }
2000 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
2001 \cs_new_eq:NN \__graphics_backend_include_bmp:n \__graphics_backend_include_jpg:n
2002 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
2003 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2004 {
2005   \bool_set_true:N \l__graphics_transgroup_bool
2006   \__graphics_backend_include_auxi:n {#1}
2007 }
2008 \cs_new_protected:Npn \__graphics_backend_include_auxi:n #1
2009 {
2010   \__graphics_backend_include_auxii:en
2011   {
2012     \tl_if_empty:NF \l__graphics_pagebox_tl
2013     { : \l__graphics_pagebox_tl }
2014     \int_compare:nNnT \l__graphics_page_int > 1
2015     { :P \int_use:N \l__graphics_page_int }
2016     \tl_if_empty:NF \l__graphics_decodearray_str
2017     { :D \l__graphics_decodearray_str }
2018     \bool_if:NT \l__graphics_interpolate_bool
2019     { :I }
2020   }
2021   {#1}
2022 }
2023 \cs_new_protected:Npn \__graphics_backend_include_auxii:nn #1#2
2024 {
2025   \int_if_exist:cTF { c__graphics_ #2#1 _int }
2026   {
2027     \__kernel_backend_literal:e
2028     { pdf:usexobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
2029   }
2030   { \__graphics_backend_include_auxiii:nn {#2} {#1} }
2031 }
2032 \cs_generate_variant:Nn \__graphics_backend_include_auxii:nn { e }

```

Inclusion using the specials is relatively straight-forward, but there is one wrinkle. To get the `pagebox` correct for PDF graphics in all cases, it is necessary to provide both that

information and the bbox argument: odd things happen otherwise! We use the dvipdfmx special in all cases as it allows attributes to be added to the XObject.

```

2033 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nn #1#2
2034 {
2035   \int_gincr:N \g__graphics_track_int
2036   \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
2037   \__kernel_backend_literal:e
2038   {
2039     pdf:image ~
2040     @graphic \int_use:c { c__graphics_ #1#2 _int } ~
2041     \int_compare:nNnT \l__graphics_page_int > 1
2042     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2043     \tl_if_empty:NF \l__graphics_pagebox_tl
2044     {
2045       pagebox ~ \l__graphics_pagebox_tl \c_space_tl
2046       bbox ~
2047         \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2048         \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2049         \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2050         \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
2051     }
2052     (#1)
2053     \bool_lazy_any:nT
2054     {
2055       { \l__graphics_interpolate_bool }
2056       { \l__graphics_transgroup_bool }
2057       { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
2058     }
2059     {
2060       <<
2061         \tl_if_empty:NF \l__graphics_decodearray_str
2062         { /Decode~[ \l__graphics_decodearray_str ] }
2063         \bool_if:NT \l__graphics_transgroup_bool
2064         { /Group << /S /Transparency /K ~ false /I ~ false >> }
2065         \bool_if:NT \l__graphics_interpolate_bool
2066         { /Interpolate~true }
2067       >>
2068     }
2069   }
2070 }

```

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

__graphics_backend_get_pagecount:n

```

2071 <dvipdfmx>
2072 \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n
2073 </dvipdfmx>

```

(End of definition for __graphics_backend_get_pagecount:n.)

```

2074 </dvipdfmx | xetex>

```

5.4 X_YTeX backend

2075 $\langle *xetex \rangle$

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

```

2076 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2077 {
2078   \int_zero:N \l__graphics_page_int
2079   \tl_clear:N \l__graphics_pagebox_tl
2080   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2081 }
2082 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2083 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
2084 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2085 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2086 {
2087   \tl_clear:N \l__graphics_decodearray_str
2088   \bool_set_false:N \l__graphics_interpolate_bool
2089   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2090 }
2091 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2092 {
2093   \int_compare:nNnTF \l__graphics_page_int > 1
2094     { \__graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2095     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2096 }
2097 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2098 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2099 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2100 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2101 {
2102   \tl_if_empty:NTF \l__graphics_pagebox_tl
2103     { \__graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2104     { \__graphics_backend_getbb_auxv:nNnn {#1} #2 {#3} {#4} }
2105 }
2106 }
2107 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2108 {
2109   \use:e
2110   {
2111     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2112     {
2113       #5
2114       \tl_if_blank:nF {#1}
2115         { \c_space_tl \__graphics_backend_getbb_pagebox:w #1 }
2116     }
2117   }
2118 }
2119 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2120 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2121 {
2122   \__graphics_bb_restore:nF {#1#3}

```

```

2123     { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2124   }
2125   \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2126   {
2127     \hbox_set:Nn \l__graphics_tmp_box { #2 #1 ~ #4 }
2128     \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_tmp_box }
2129     \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_tmp_box }
2130     \__graphics_bb_save:n {#1#3}
2131   }
2132   \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.

```

2133   \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
2134   {
2135     \int_const:cn { c__graphics_ #1 _pages_int }
2136     {
2137       \int_max:nn
2138       { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2139       { 1 }
2140     }
2141   }

```

(End of definition for __graphics_backend_get_pagecount:n.)

2142 </xetex>

5.5 dvisvgm backend

2143 <*dvisvgm>

\l_graphics_search_ext_seq

```

2144   \seq_set_from_clist:Nn \l_graphics_search_ext_seq
2145   { .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.

```

2146   \cs_new_protected:Npn \__graphics_backend_getbb_svg:n #1
2147   {
2148     \__graphics_bb_restore:nF {#1}
2149     {
2150       \ior_open:Nn \l__graphics_tmp_ior {#1}
2151       \ior_if_eof:NTF \l__graphics_tmp_ior
2152       { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2153       {
2154         \dim_zero:N \l__graphics_llx_dim
2155         \dim_zero:N \l__graphics_lly_dim
2156         \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2157         \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2158         \ior_str_map_inline:Nn \l__graphics_tmp_ior

```

```

2159         {
2160             \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2161             {
2162                 \__graphics_backend_getbb_svg_auxi:nNn
2163                 { width } \l__graphics_urx_dim {##1}
2164             }
2165             \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2166             {
2167                 \__graphics_backend_getbb_svg_auxi:nNn
2168                 { height } \l__graphics_ury_dim {##1}
2169             }
2170             \bool_lazy_and:nnF
2171             { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2172             { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2173             { \ior_map_break: }
2174         }
2175         \__graphics_bb_save:n {#1}
2176     }
2177     \ior_close:N \l__graphics_tmp_ior
2178 }
2179 }
2180 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2181 {
2182     \use:e
2183     {
2184         \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxii:w
2185         ##1 \tl_to_str:n {#1} = ##2 \tl_to_str:n {#1} = ##3
2186         \s__graphics_stop
2187     }
2188     {
2189         \tl_if_blank:nF {##2}
2190         {
2191             \peek_remove_spaces:n
2192             {
2193                 \peek_meaning:NTF ' % '
2194                 { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2195                 {
2196                     \peek_meaning:NTF " % "
2197                     { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2198                     { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2199                 }
2200             }
2201             ##2 \s__graphics_stop
2202         }
2203     }
2204     \use:e
2205     {
2206         \__graphics_backend_getbb_svg_auxii:w #3
2207         \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2208         \s__graphics_stop
2209     }
2210 }
2211 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2212 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop

```

```

2213 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2214 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2215 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2216 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2217 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2218 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2219 {
2220   \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2221   \l__graphics_tmp_dim #2 bp \scan_stop:
2222   \dim_set_eq:NN #1 \l__graphics_tmp_dim
2223 }
2224 \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.

```

2225 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2226 \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.

```

2227 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2228 {
2229   \int_zero:N \l__graphics_page_int
2230   \tl_clear:N \l__graphics_pagebox_tl
2231   \__graphics_extract_bb:n {#1}
2232 }
2233 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2234 \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

```

2235 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2236 {
2237   \tl_clear:N \l__graphics_decodearray_str
2238   \bool_set_false:N \l__graphics_interpolate_bool
2239   \__graphics_extract_bb:n {#1}
2240 }

```

(End of definition for __graphics_backend_getbb_pdf:n.)

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

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

```

2241 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2242 { \__graphics_backend_include:nn { PSfile } {#1} }
2243 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
2244 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2245 { \__graphics_backend_include:nn { pdffile } {#1} }
2246 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
2247 {
2248   \__kernel_backend_literal:e
2249   {

```

```

2250         #1 = #2 \c_space_tl
2251         llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2252         lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2253         urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2254         ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2255     }
2256 }

```

(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.

```

2257 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2258 {
2259     \box_move_up:nn { \l__graphics_ury_dim }
2260     {
2261         \hbox:n
2262         {
2263             \__kernel_backend_literal:e
2264             {
2265                 dvisvgm:img~
2266                 \dim_to_decimal:n { \l__graphics_urx_dim } ~
2267                 \dim_to_decimal:n { \l__graphics_ury_dim } ~
2268                 \__graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2269             }
2270         }
2271     }
2272 }
2273 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_svg:n
2274 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_svg:n
2275 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_svg:n
2276 \cs_new:Npn \__graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2277 {#2}

```

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

```

\__graphics_backend_get_pagecount:n

```

```

2278 \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n

```

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

```

2279 </dvisvgm>
2280 </package>

```

6 l3backend-pdf implementation

```

2281 <*package>
2282 <@@=pdf>

```

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

6.1 dvips backend

2283 `<*dvips>`

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

Used often enough it should be a separate function.

2284 `\cs_new_protected:Npn \__pdf_backend_pdfmark:n #1`
2285 `{ \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }`
2286 `\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`

2287 `\cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2`
2288 `{ \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }`
2289 `\cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2`
2290 `{ \__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`

2291 `\cs_new_protected:Npn \__pdf_backend_object_new:`
2292 `{ \int_gincr:N \g__pdf_backend_object_int }`
2293 `\cs_new:Npn \__pdf_backend_object_ref:n #1 { { pdf.obj #1 } }`
2294 `\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.

2295 `\cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3`
2296 `{`
2297 `\__pdf_backend_object_write_aux:nnn`
2298 `{ \__pdf_backend_object_ref:n {#1} }`
2299 `{#2} {#3}`
2300 `}`
2301 `\cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }`
2302 `\cs_new_protected:Npn \__pdf_backend_object_write_aux:nnn #1#2#3`
2303 `{`
2304 `\__pdf_backend_pdfmark:e`
2305 `{`
2306 `/_objdef ~ #1`
2307 `/type`
2308 `\str_case:nn {#2}`
2309 `{`
2310 `{ array } { /array }`
2311 `{ dict } { /dict }`

```

2312         { fstream } { /stream }
2313         { stream } { /stream }
2314     }
2315     /OBJ
2316 }
2317 \use:c { __pdf_backend_object_write_ #2 :nn } {#1} {#3}
2318 }
2319 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2320 {
2321     \__pdf_backend_pdfmark:e
2322     { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2323 }
2324 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2325 {
2326     \__pdf_backend_pdfmark:e
2327     { #1 << \exp_not:n {#2} >> /PUT }
2328 }
2329 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2330 {
2331     \exp_args:Ne
2332     \__pdf_backend_object_write_fstream:nnn {#1} #2
2333 }
2334 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2335 {
2336     \__kernel_backend_postscript:n
2337     {
2338         SDict ~ begin ~
2339         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2340         mark ~ #1 ~ ( #3 ) ~ ( r ) ~ file ~ /PUT ~ pdfmark ~
2341         end
2342     }
2343 }
2344 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2345 {
2346     \exp_args:Ne
2347     \__pdf_backend_object_write_stream:nnn {#1} #2
2348 }
2349 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2350 {
2351     \__kernel_backend_postscript:n
2352     {
2353         mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2354         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2355     }
2356 }

```

(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
2357 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2358 {
2359     \int_gincr:N \g__pdf_backend_object_int
2360     \__pdf_backend_object_write_aux:nnn
2361     { { pdf.obj \int_use:N \g__pdf_backend_object_int } }

```

```

2362     {#1} {#2}
2363   }
2364   \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.

```

2365   \cs_new:Npn \__pdf_backend_object_last:
2366     { { 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.

```

2367   \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2368     { { Page #1 } }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.1.3 Destinations

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

```

2369   \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2370     {
2371       \__kernel_backend_postscript:n { pdf.dest.anchor }
2372       \__pdf_backend_pdfmark:e
2373       {
2374         /View
2375         [
2376           \str_case:nnF {#2}
2377             {
2378               { xyz } { /XYZ ~ pdf.dest.point ~ null }
2379               { fit } { /Fit }
2380               { fitb } { /FitB }
2381               { fitbh } { /FitBH ~ pdf.dest.y }
2382               { fitbv } { /FitBV ~ pdf.dest.x }
2383               { fith } { /FitH ~ pdf.dest.y }
2384               { fitv } { /FitV ~ pdf.dest.x }
2385               { fitr } { /Fit }
2386             }
2387             {
2388               /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2389             }
2390           ]
2391           /Dest ( \exp_not:n {#1} ) cvn
2392           /DEST
2393         }
2394       }
2395   \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2396     {
2397       \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2398       { \dim_eval:n {#2} } {#1} {#3} {#4}

```

```

2399 }
2400 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2401 {
2402   \vbox_to_zero:n
2403   {
2404     \dim_vertical:n {#4}
2405     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2406     \tex_vss:D
2407   }
2408   \dim_horizontal:n {#1}
2409   \vbox_to_zero:n
2410   {
2411     \dim_vertical:n { -#3 }
2412     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2413     \tex_vss:D
2414   }
2415   \dim_horizontal:n { -#1 }
2416   \__pdf_backend_pdfmark:n
2417   {
2418     /View
2419     [
2420       /FitR ~
2421       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2422       pdf.urx ~ pdf.ury ~ pdf.dest2device
2423     ]
2424     /Dest ( #2 ) cvn
2425     /DEST
2426   }
2427 }

```

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

6.1.4 Structure

Doable for the usual ps2pdf method.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2428 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2429 {
2430   \int_compare:nNnT {#1} = 0
2431   {
2432     \__kernel_backend_literal_postscript:n
2433     {
2434       /setdistillerparams ~ where
2435       { pop << /CompressPages ~ false >> setdistillerparams }
2436       if
2437     }
2438   }
2439 }
2440 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2441 {
2442   \bool_if:nF {#1}
2443   {
2444     \__kernel_backend_literal_postscript:n
2445     {

```

```

2446         /setdistillerparams ~ where
2447         { pop << /CompressStreams ~ false >> setdistillerparams }
2448         if
2449     }
2450 }
2451 }

```

(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
2452 \cs_new_protected:Npn __pdf_backend_version_major_gset:n #1
2453 {
2454     \cs_gset:Npe __pdf_backend_version_major: { \int_eval:n {#1} }
2455 }
2456 \cs_new_protected:Npn __pdf_backend_version_minor_gset:n #1
2457 {
2458     \cs_gset:Npe __pdf_backend_version_minor: { \int_eval:n {#1} }
2459 }

```

(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:
2460 \cs_new:Npn __pdf_backend_version_major: { -1 }
2461 \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:
2462 \cs_new_protected:Npn __pdf_backend_bdc:nn #1#2
2463 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2464 \cs_new_protected:Npn __pdf_backend_emc:
2465 { \__pdf_backend_pdfmark:n { /EMC } }

```

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

2466 $\langle$ /dvips $\rangle$

6.2 LuaTeX and pdfTeX backend

2467 $\langle$ *luatex | pdftex $\rangle$

6.2.1 Destinations

```

__pdf_backend_destination:nn A simple task: pass the data to the primitive. The \scan_stop: deals with the danger
__pdf_backend_destination:nnnn 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.

```

```

2468 \cs_new_protected:Npn __pdf_backend_destination:nn #1#2
2469 {
2470      $\langle$ *luatex $\rangle$ 
2471     \tex_pdfextension:D dest ~
2472      $\langle$ /luatex $\rangle$ 
2473      $\langle$ *pdftex $\rangle$ 
2474     \tex_pdfdest:D

```

```

2475 </pdfTeX>
2476     name {#1}
2477     \str_case:nnF {#2}
2478     {
2479         { xyz } { xyz }
2480         { fit } { fit }
2481         { fitb } { fitb }
2482         { fitbh } { fitbh }
2483         { fitbv } { fitbv }
2484         { fith } { fith }
2485         { fitv } { fitv }
2486         { fitr } { fitr }
2487     }
2488     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2489     \scan_stop:
2490 }
2491 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2492 {
2493 <*luatex>
2494     \tex_pdfextension:D dest ~
2495 </luatex>
2496 <*pdfTeX>
2497     \tex_pdfdest:D
2498 </pdfTeX>
2499     name {#1}
2500     fitr ~
2501     width \dim_eval:n {#2} ~
2502     height \dim_eval:n {#3} ~
2503     depth \dim_eval:n {#4} \scan_stop:
2504 }

```

(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
2505 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2506 {
2507 <*luatex>
2508     \tex_pdfextension:D catalog
2509 </luatex>
2510 <*pdfTeX>
2511     \tex_pdfcatalog:D
2512 </pdfTeX>
2513     { / #1 ~ #2 }
2514 }
2515 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2516 {
2517 <*luatex>
2518     \tex_pdfextension:D info
2519 </luatex>
2520 <*pdfTeX>
2521     \tex_pdfinfo:D
2522 </pdfTeX>

```

```

2523     { / #1 ~ #2 }
2524 }

```

(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.

```

2525 \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 2526 \cs_new_protected:Npn \_pdf_backend_object_new:
\_pdf_backend_object_id:n 2527 {
2528   \*luatex
2529   \tex_pdfextension:D obj ~
2530   \*pdfextension:D
2531   \tex_pdfobj:D
2532   \*pdfobj:D
2533   \tex_pdfobj:D
2534   reserveobjnum ~
2535   \int_gset:Nn \g_pdf_backend_object_int
2536   \*luatex
2537   { \tex_pdffeedback:D lastobj }
2538   \*pdfextension:D
2539   \tex_pdflastobj:D
2540   \*pdfextension:D
2541   { \tex_pdflastobj:D }
2542   }
2543 \cs_new:Npn \_pdf_backend_object_ref:n #1 { #1 ~ 0 ~ R }
2544 \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 2545 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
\_pdf_backend_object_write:nn 2546 {
\_pdf_exp_not_i:nn 2547   \*luatex
\_pdf_exp_not_ii:nn 2548   \tex_immediate:D \tex_pdfextension:D obj ~
2549   \*pdfextension:D
2550   \tex_immediate:D \tex_pdfobj:D
2551   \*pdfobj:D
2552   \tex_immediate:D \tex_pdfobj:D
2553   useobjnum ~ #1
2554   \_pdf_backend_object_write:nn {#2} {#3}
2555   }
2556 \cs_new:Npn \_pdf_backend_object_write:nn #1#2
2557 {
2558   \str_case:nn {#1}
2559   {
2560     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2561     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2562     { fstream }

```

```

2563     {
2564         stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2565         file ~ { \_pdf_exp_not_ii:nn #2 }
2566     }
2567 { stream }
2568 {
2569     stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2570     { \_pdf_exp_not_ii:nn #2 }
2571 }
2572 }
2573 }
2574 \cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nne }
2575 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2576 \cs_new:Npn \_pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

```

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

_pdf_backend_object_now:nn
_pdf_backend_object_now:ne

Much like writing, but direct creation.

```

2577 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2578 {
2579 <*luatex>
2580     \tex_immediate:D \tex_pdfextension:D obj ~
2581 </luatex>
2582 <*pdfTeX>
2583     \tex_immediate:D \tex_pdfobj:D
2584 </pdfTeX>
2585     \_pdf_backend_object_write:nn {#1} {#2}
2586 }
2587 \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.

```

2588 \cs_new:Npe \_pdf_backend_object_last:
2589 {
2590     \exp_not:N \int_value:w
2591 <*luatex>
2592     \exp_not:N \tex_pdffeedback:D lastobj ~
2593 </luatex>
2594 <*pdfTeX>
2595     \exp_not:N \tex_pdflastobj:D
2596 </pdfTeX>
2597     \c_space_tl 0 ~ R
2598 }

```

(End of definition for _pdf_backend_object_last:.)

_pdf_backend_pageobject_ref:n

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

```

2599 \cs_new:Npe \_pdf_backend_pageobject_ref:n #1
2600 {
2601     \exp_not:N \int_value:w
2602 <*luatex>
2603     \exp_not:N \tex_pdffeedback:D pageref
2604 </luatex>
2605 <*pdfTeX>

```

```

2606 \exp_not:N \tex_pdfpageref:D
2607 </pdfTeX>
2608 \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2609 }

```

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

6.2.4 Structure

```

\__pdf_backend_compresslevel:n Simply pass data to the engine.
\__pdf_backend_compress_objects:n 2610 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
\__pdf_backend_objcompresslevel:n 2611 {
2612   \tex_global:D
2613   <*luatex>
2614   \tex_pdfvariable:D compresslevel
2615   </luatex>
2616   <*pdfTeX>
2617   \tex_pdfcompresslevel:D
2618   </pdfTeX>
2619   \int_value:w \int_eval:n {#1} \scan_stop:
2620 }
2621 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2622 {
2623   \bool_if:nTF {#1}
2624   { \__pdf_backend_objcompresslevel:n { 2 } }
2625   { \__pdf_backend_objcompresslevel:n { 0 } }
2626 }
2627 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2628 {
2629   \tex_global:D
2630   <*luatex>
2631   \tex_pdfvariable:D objcompresslevel
2632   </luatex>
2633   <*pdfTeX>
2634   \tex_pdfobjcompresslevel:D
2635   </pdfTeX>
2636   #1 \scan_stop:
2637 }

```

(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`

```

2638 \cs_new_protected:Npe \__pdf_backend_version_major_gset:n #1
2639 {
2640   <*luatex>
2641   \int_compare:nNnT \tex_luatexversion:D > { 106 }
2642   {
2643     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2644     \exp_not:N \int_eval:n {#1} \scan_stop:
2645   }
2646   </luatex>
2647   <*pdfTeX>
2648   \cs_if_exist:NT \tex_pdfmajorversion:D
2649   {

```

```

2650         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2651         \exp_not:N \int_eval:n {#1} \scan_stop:
2652     }
2653 </pdfTeX>
2654 }
2655 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2656 {
2657     \tex_global:D
2658     <*luatex>
2659     \tex_pdfvariable:D minorversion
2660 </luatex>
2661 <*pdfTeX>
2662     \tex_pdfminorversion:D
2663 </pdfTeX>
2664     \int_eval:n {#1} \scan_stop:
2665 }

```

(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: 2666 \cs_new:Npe \__pdf_backend_version_major:
2667 {
2668     <*luatex>
2669     \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2670     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2671     { 1 }
2672 </luatex>
2673 <*pdfTeX>
2674     \cs_if_exist:NTF \tex_pdfmajorversion:D
2675     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2676     { 1 }
2677 </pdfTeX>
2678 }
2679 \cs_new:Npn \__pdf_backend_version_minor:
2680 {
2681     \tex_the:D
2682     <*luatex>
2683     \tex_pdfvariable:D minorversion
2684 </luatex>
2685 <*pdfTeX>
2686     \tex_pdfminorversion:D
2687 </pdfTeX>
2688 }

```

(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>.

```

2689 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2690 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2691 \cs_new_protected:Npn \__pdf_backend_emc:
2692 { \__kernel_backend_literal_page:n { EMC } }

```

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

2693 `</luatex | pdftex>`

6.3 dvipdfmx backend

2694 `<*dvipdfmx | xetex>`

`\_pdf_backend:n`
`\_pdf_backend:e`

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

2695 `\cs_new_protected:Npe \_pdf_backend:n #1`
2696 `{ \_kernel_backend_literal:n { pdf: #1 } }`
2697 `\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`

2698 `\cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2`
2699 `{ \_pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }`
2700 `\cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2`
2701 `{ \_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.

2702 `\prop_new:N \g_pdf_backend_object_prop`

(End of definition for `\g_pdf_backend_object_prop`.)

`\_pdf_backend_object_new:`
`\_pdf_backend_object_ref:n`
`\_pdf_backend_object_id:n`

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

2703 `\cs_new_protected:Npn \_pdf_backend_object_new:`
2704 `{ \int_gincr:N \g_pdf_backend_object_int }`
2705 `\cs_new:Npn \_pdf_backend_object_ref:n #1 { @pdf.obj #1 }`
2706 `\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`

This is where we choose the actual type.

2707 `\cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3`
2708 `{`
2709 `\use:c { \_pdf_backend_object_write_ #2 :nn }`
2710 `{ \_pdf_backend_object_ref:n {#1} } {#3}`
2711 `}`
2712 `\cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nne }`
2713 `\cs_new_protected:Npn \_pdf_backend_object_write_array:nn #1#2`
2714 `{`
2715 `\_pdf_backend:e`
2716 `{ obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }`
2717 `}`
2718 `\cs_new_protected:Npn \_pdf_backend_object_write_dict:nn #1#2`
2719 `{`

```

2720 \__pdf_backend:e
2721 { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2722 }
2723 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2724 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2725 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2726 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2727 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2728 {
2729 \__pdf_backend:e
2730 {
2731 #1 stream ~ #2 ~
2732 ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2733 }
2734 }

```

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

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

```

\__pdf_backend_object_now:ne
2735 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2736 {
2737 \int_gincr:N \g__pdf_backend_object_int
2738 \exp_args:Nne \use:c { __pdf_backend_object_write_ #1 :nn }
2739 { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2740 {#2}
2741 }
2742 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2743 \cs_new:Npn \__pdf_backend_object_last:
2744 { @pdf.obj \int_use:N \g__pdf_backend_object_int }

```

(End of definition for __pdf_backend_object_last:.)

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

```

2745 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2746 { @page #1 }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.3.3 Destinations

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

```

2747 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2748 {
2749 \__pdf_backend:e
2750 {
2751 dest ~ ( \exp_not:n {#1} )
2752 [

```

```

2753      @thispage
2754      \str_case:nnF {#2}
2755      {
2756        { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2757        { fit } { /Fit }
2758        { fitb } { /FitB }
2759        { fitbh } { /FitBH }
2760        { fitbv } { /FitBV ~ @xpos }
2761        { fith } { /FitH ~ @ypos }
2762        { fitv } { /FitV ~ @xpos }
2763        { fitr } { /Fit }
2764      }
2765      { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2766    ]
2767  }
2768 }
2769 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2770 {
2771   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2772   { \dim_eval:n {#2} } {#1} {#3} {#4}
2773 }
2774 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2775 {
2776   \vbox_to_zero:n
2777   {
2778     \dim_vertical:n {#4}
2779     \hbox:n
2780     {
2781       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2782       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2783     }
2784     \tex_vss:D
2785   }
2786   \dim_horizontal:n {#1}
2787   \vbox_to_zero:n
2788   {
2789     \dim_vertical:n { -#3 }
2790     \hbox:n
2791     {
2792       \__pdf_backend:n
2793       {
2794         dest ~ (#2)
2795         [
2796           @thispage
2797           /FitR ~
2798           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2799           @xpos ~ @ypos
2800         ]
2801       }
2802     }
2803     \tex_vss:D
2804   }
2805   \dim_horizontal:n { -#1 }
2806 }

```

(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.

```
2807 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2808 { \__kernel_backend_literal:e { dvipdfmx:config~z~ \int_eval:n {#1} } }
2809 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2810 {
2811   \bool_if:nF {#1}
2812   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2813 }
```

(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.

```
2814 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2815 {
2816   \cs_gset:Npe \_pdf_backend_version_major: { \int_eval:n {#1} }
2817   \__kernel_backend_literal:e { pdf:majorversion~ \_pdf_backend_version_major: }
2818 }
2819 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2820 {
2821   \cs_gset:Npe \_pdf_backend_version_minor: { \int_eval:n {#1} }
2822   \__kernel_backend_literal:e { pdf:minorversion~ \_pdf_backend_version_minor: }
2823 }
```

(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.

```
2824 \cs_new:Npn \_pdf_backend_version_major: { 1 }
2825 \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`
`\_pdf_backend_emc:`

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

```
2826 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2827 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2828 \cs_new_protected:Npn \_pdf_backend_emc:
2829 { \__kernel_backend_literal_page:n { EMC } }
```

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

```
2830 </dvipdfmx | xetex>
```

6.4 dvisvgm backend

2831 $\langle *dvisvgm \rangle$

6.4.1 Destinations

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

2832 `\cs_new_protected:Npn \_pdf_backend_destination:nn #1#2 { }`
2833 `\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`

No-op.

2834 `\cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2 { }`
2835 `\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:nne`
`\_pdf_backend_object_now:nn`
`\_pdf_backend_object_now:ne`
`\_pdf_backend_object_last:`
`\_pdf_backend_pageobject_ref:n`

All no-ops here.

2836 `\cs_new_protected:Npn \_pdf_backend_object_new: { }`
2837 `\cs_new:Npn \_pdf_backend_object_ref:n #1 { }`
2838 `\cs_new:Npn \_pdf_backend_object_id:n #1 { }`
2839 `\cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3 { }`
2840 `\cs_new_protected:Npn \_pdf_backend_object_write:nne #1#2#3 { }`
2841 `\cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }`
2842 `\cs_new_protected:Npn \_pdf_backend_object_now:ne #1#2 { }`
2843 `\cs_new:Npn \_pdf_backend_object_last: { }`
2844 `\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`

These are all no-ops.

2845 `\cs_new_protected:Npn \_pdf_backend_compresslevel:n #1 { }`
2846 `\cs_new_protected:Npn \_pdf_backend_compress_objects:n #1 { }`

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

`\_pdf_backend_version_major_gset:n`
`\_pdf_backend_version_minor_gset:n`

Data not available!

2847 `\cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }`
2848 `\cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1 { }`

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

`\_pdf_backend_version_major:`
`\_pdf_backend_version_minor:`

Data not available!

2849 `\cs_new:Npn \_pdf_backend_version_major: { -1 }`
2850 `\cs_new:Npn \_pdf_backend_version_minor: { -1 }`

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

```

\__pdf_backend_bdc:nn More no-ops.
\__pdf_backend_emc: 2851 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2852 \cs_new_protected:Npn \__pdf_backend_emc: { }
(End of definition for \__pdf_backend_bdc:nn and \__pdf_backend_emc:.)
2853 \</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.

```

2854 \<*dvipdfmx | dvips>

\__pdf_backend_pagesize_gset:nn This is done as a backend literal, so we deal with it using the shipout hook.
2855 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2
2856 {
2857   \__kernel_backend_first_shipout:n
2858   {
2859     \__kernel_backend_literal:e
2860     {
2861       \<*dvipdfmx>
2862         pdf:pagesize ~
2863         width ~ \dim_eval:n {#1} ~
2864         height ~ \dim_eval:n {#2}
2865       \</dvipdfmx>
2866       \<*dvips>
2867         papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
2868       \</dvips>
2869     }
2870   }
2871 }
(End of definition for \__pdf_backend_pagesize_gset:nn.)
2872 \</dvipdfmx | dvips>
2873 \<*luatex | pdftex | xetex>

```

```

\__pdf_backend_pagesize_gset:nn Pass to the primitives.
2874 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2
2875 {
2876   \dim_gset:Nn \tex_pagewidth:D {#1}
2877   \dim_gset:Nn \tex_pageheight:D {#2}
2878 }
(End of definition for \__pdf_backend_pagesize_gset:nn.)
2879 \</luatex | pdftex | xetex>
2880 \<*dvisvgm>

```

```

\__pdf_backend_pagesize_gset:nn A no-op.
2881 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2 { }
(End of definition for \__pdf_backend_pagesize_gset:nn.)
2882 \</dvisvgm>
2883 \</package>

```

7 l3backend-pdfannot implementation

```
2884 <*package>
2885 <@@=pdfannot>
```

7.1 dvips backend

```
2886 <*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.

```
2887 \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.

```
2888 \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.

```
2889 \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.

```
2890 \cs_new_protected:Npn \__pdfannot_backend_generic:nnnn #1#2#3#4
2891 {
2892   \exp_args:Nf \__pdfannot_backend_generic_aux:nnnn
2893     { \dim_eval:n {#1} } {#2} {#3} {#4}
2894 }
2895 \cs_new_protected:Npn \__pdfannot_backend_generic_aux:nnnn #1#2#3#4
2896 {
2897   \box_move_down:nn {#3}
2898   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2899   \box_move_up:nn {#2}
2900   {
2901     \hbox:n
2902     {
2903       \dim_horizontal:n {#1}
2904       \__kernel_backend_postscript:n { pdf.save.ur }
2905       \dim_horizontal:n { -#1 }
2906     }
2907   }
2908   \int_gincr:N \g__pdfannot_backend_int
2909   \__kernel_backend_postscript:e
2910   {
2911     mark
2912     /_objdef { pdf.annot \int_use:N \g__pdfannot_backend_int }
```

```

2913         pdf.rect
2914         #4 ~
2915         /ANN ~
2916         pdfmark
2917     }
2918 }

```

(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.

```

2919 \cs_new:Npn \_pdfannot_backend_last:
2920 { { 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.

```

2921 \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.

```

2922 \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.

```

2923 \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.

```

2924 \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.

```

2925 \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.

```

2926 \tl_new:N \l__pdfannot_backend_breaklink_pdfmark_tl
2927 \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.

```

2928 \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.

```

2929 \cs_new_eq:NN \_pdfannot_backend_breaklink_usebox:N \box_use:N

```

(End of definition for _pdfannot_backend_breaklink_usebox:N.)

```

\_pdfannot_backend_link_begin_goto:nnw
\_pdfannot_backend_link_begin_user:nnw
\_pdfannot_backend_link:nw
  \_pdfannot_backend_link_aux:nw
  \_pdfannot_backend_link_end:
  \_pdfannot_backend_link_end_aux:
  \_pdfannot_backend_link_minima:
  \_pdfannot_backend_link_outerbox:n
  \_pdfannot_backend_link_sf_save:
  \_pdfannot_backend_link_sf_restore:

```

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

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

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

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

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

```

2930 \cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nnw #1#2
2931 {
2932   \_pdfannot_backend_link_begin:nw
2933   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2934 }
2935 \cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nnw #1#2
2936 { \_pdfannot_backend_link_begin:nw {#1#2} }
2937 \cs_new_protected:Npn \_pdfannot_backend_link_begin:nw #1
2938 {
2939   \bool_if:NF \g__pdfannot_backend_link_bool
2940   { \_pdfannot_backend_link_begin_aux:nw {#1} }
2941 }

```

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.

```

2942 \cs_new_protected:Npn \_pdfannot_backend_link_begin_aux:nw #1
2943 {
2944   \bool_gset_true:N \g__pdfannot_backend_link_bool
2945   \_kernel_backend_postscript:n
2946   { /pdf.link.dict ( #1 ) def }
2947   \tl_gset:Nn \g__pdfannot_backend_link_dict_tl {#1}
2948   \_pdfannot_backend_link_sf_save:
2949   \mode_if_math:TF
2950   { \bool_gset_true:N \g__pdfannot_backend_link_math_bool }
2951   { \bool_gset_false:N \g__pdfannot_backend_link_math_bool }
2952   \hbox_set:Nw \l__pdfannot_backend_content_box
2953   \_pdfannot_backend_link_sf_restore:
2954   \bool_if:NT \g__pdfannot_backend_link_math_bool
2955   { \c_math_toggle_token }
2956 }
2957 \cs_new_protected:Npn \_pdfannot_backend_link_end:
2958 {
2959   \bool_if:NT \g__pdfannot_backend_link_bool
2960   { \_pdfannot_backend_link_end_aux: }
2961 }
2962 \cs_new_protected:Npn \_pdfannot_backend_link_end_aux:

```

```

2963 {
2964   \bool_if:NT \g__pdfannot_backend_link_math_bool
2965   { \c_math_toggle_token }
2966   \__pdfannot_backend_link_sf_save:
2967   \hbox_set_end:
2968   \__pdfannot_backend_link_minima:
2969   \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
2970   \exp_args:Ne \__pdfannot_backend_link_outerbox:n
2971   {
2972     \int_if_odd:nTF { \value { page } }
2973     { \oddsidemargin }
2974     { \evensidemargin }
2975   }
2976   \box_move_down:nn { \box_dp:N \l__pdfannot_backend_content_box }
2977   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2978   \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.begin }
2979   \__pdfannot_backend_breaklink_usebox:N \l__pdfannot_backend_content_box
2980   \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.end }
2981   \box_move_up:nn { \box_ht:N \l__pdfannot_backend_content_box }
2982   {
2983     \hbox:n
2984     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2985   }
2986   \int_gincr:N \g__pdfannot_backend_int
2987   \int_gset_eq:NN \g__pdfannot_backend_link_int \g__pdfannot_backend_int
2988   \__kernel_backend_postscript:e
2989   {
2990     mark
2991     /_objdef { pdf.annot \int_use:N \g__pdfannot_backend_link_int }
2992     \g__pdfannot_backend_link_dict_tl \c_space_tl
2993     pdf.rect
2994     /ANN ~ \l__pdfannot_backend_breaklink_pdfmark_tl
2995   }
2996   \__pdfannot_backend_link_sf_restore:
2997   \bool_gset_false:N \g__pdfannot_backend_link_bool
2998 }
2999 \cs_new_protected:Npn \__pdfannot_backend_link_minima:
3000 {
3001   \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
3002   \__kernel_backend_postscript:e
3003   {
3004     /pdf.linkdp.pad ~
3005     \dim_to_decimal:n
3006     {
3007       \dim_max:nn
3008       {
3009         \box_dp:N \l__pdfannot_backend_model_box
3010         - \box_dp:N \l__pdfannot_backend_content_box
3011       }
3012       { Opt }
3013     } ~
3014     pdf.pt.dvi ~ def
3015     /pdf.linkht.pad ~
3016     \dim_to_decimal:n

```

```

3017         {
3018             \dim_max:nn
3019             {
3020                 \box_ht:N \l__pdfannot_backend_model_box
3021                 - \box_ht:N \l__pdfannot_backend_content_box
3022             }
3023             { Opt }
3024         } ~
3025         pdf.pt.dvi ~ def
3026     }
3027 }
3028 \cs_new_protected:Npn \__pdfannot_backend_link_outerbox:n #1
3029 {
3030     \__kernel_backend_postscript:e
3031     {
3032         /pdf.outerbox
3033         [
3034             \dim_to_decimal:n {#1} ~
3035             \dim_to_decimal:n { -\box_dp:N \l__pdfannot_backend_model_box } ~
3036             \dim_to_decimal:n { #1 + \textwidth } ~
3037             \dim_to_decimal:n { \box_ht:N \l__pdfannot_backend_model_box }
3038         ]
3039         [ exch { pdf.pt.dvi } forall ] def
3040         /pdf.baselineskip ~
3041         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
3042         { pdf.pt.dvi ~ def }
3043         { pop ~ pop }
3044         ifelse
3045     }
3046 }
3047 \cs_new_protected:Npn \__pdfannot_backend_link_sf_save:
3048 {
3049     \int_gset:Nn \g__pdfannot_backend_link_sf_int
3050     {
3051         \mode_if_horizontal:TF
3052         { \tex_spacefactor:D }
3053         { 0 }
3054     }
3055 }
3056 \cs_new_protected:Npn \__pdfannot_backend_link_sf_restore:
3057 {
3058     \mode_if_horizontal:T
3059     {
3060         \int_compare:nNnT \g__pdfannot_backend_link_sf_int > { 0 }
3061         { \int_set:Nn \tex_spacefactor:D \g__pdfannot_backend_link_sf_int }
3062     }
3063 }

```

(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.

```

3064 \use_none:nnn
3065 \cs_if_exist:NT \hook_gput_code:nnn
3066 {

```

```

3067 \hook_gput_code:nnn { build/column/after } { backend }
3068 {
3069   \box_if_empty:NF \l_shipout_box
3070   {
3071     \vbox_set:Nn \l_shipout_box
3072     {
3073       \__kernel_backend_postscript:n
3074       {
3075         pdf.globaldict /pdf.brokenlink.rect ~ known
3076         { pdf.bordertracking.continue }
3077         if
3078       }
3079       \vbox_unpack_drop:N \l_shipout_box
3080       \__kernel_backend_postscript:n
3081       { pdf.bordertracking.endpage }
3082     }
3083   }
3084 }
3085 \tl_set:Nn \l__pdfannot_backend_breaklink_pdfmark_tl { pdf.pdfmark }
3086 \cs_set_eq:NN \__pdfannot_backend_breaklink_postscript:n
3087 \__kernel_backend_postscript:n
3088 \cs_set_eq:NN \__pdfannot_backend_breaklink_usebox:N \hbox_unpack:N
3089 }

```

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

```

3090 \cs_new:Npn \__pdfannot_backend_link_last:
3091 { { 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.

```

3092 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1
3093 {
3094   \__kernel_backend_postscript:e
3095   {
3096     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
3097   }
3098 }

```

(End of definition for __pdfannot_backend_link_margin:n.)

_pdfannot_backend_link_on:

```

\__pdfannot_backend_link_off:
3099 \cs_new_protected:Npn \__pdfannot_backend_link_on: { }
3100 \cs_new_protected:Npn \__pdfannot_backend_link_off: { }

```

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

```

3101 </dvips>

```

7.2 LuaTeX and pdfTeX backend

3102 `<*luatex | pdftex>`

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

```
3103 \cs_new_protected:Npn \_pdfannot_backend_generic:nnnn #1#2#3#4
3104 {
3105   <*luatex>
3106     \tex_pdfextension:D annot ~
3107   </luatex>
3108   <*pdftex>
3109     \tex_pdfannot:D
3110   </pdftex>
3111     width ~ \dim_eval:n {#1} ~
3112     height ~ \dim_eval:n {#2} ~
3113     depth ~ \dim_eval:n {#3} ~
3114     {#4}
3115 }
```

(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.

```
3116 \cs_new:Npe \_pdfannot_backend_last:
3117 {
3118   \exp_not:N \int_value:w
3119   <*luatex>
3120     \exp_not:N \tex_pdffeedback:D lastannot ~
3121   </luatex>
3122   <*pdftex>
3123     \exp_not:N \tex_pdflastannot:D
3124   </pdftex>
3125     \c_space_tl 0 ~ R
3126 }
```

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

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

```
\_pdfannot_backend_link_begin_user:nnw
\_pdfannot_backend_link_begin:nnnw
\_pdfannot_backend_link_end:
3127 \cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nnw #1#2
3128 { \_pdfannot_backend_link_begin:nnnw {#1} { goto~name } {#2} }
3129 \cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nnw #1#2
3130 { \_pdfannot_backend_link_begin:nnnw {#1} { user } {#2} }
3131 \cs_new_protected:Npn \_pdfannot_backend_link_begin:nnnw #1#2#3
3132 {
3133   <*luatex>
3134     \tex_pdfextension:D startlink ~
3135   </luatex>
3136   <*pdftex>
3137     \tex_pdfstartlink:D
3138   </pdftex>
3139     attr {#1}
3140     #2 {#3}
3141   }
3142 \cs_new_protected:Npn \_pdfannot_backend_link_end:
```

```

3143 {
3144 <*luatex>
3145 \tex_pdfextension:D endlink \scan_stop:
3146 </luatex>
3147 <*pdftex>
3148 \tex_pdfendlink:D
3149 </pdftex>
3150 }

```

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

_pdfannot_backend_link_last: Formatted for direct use.

```

3151 \cs_new:Npe \_pdfannot_backend_link_last:
3152 {
3153 \exp_not:N \int_value:w
3154 <*luatex>
3155 \exp_not:N \tex_pdffeedback:D lastlink ~
3156 </luatex>
3157 <*pdftex>
3158 \exp_not:N \tex_pdflastlink:D
3159 </pdftex>
3160 \c_space_tl 0 ~ R
3161 }

```

(End of definition for _pdfannot_backend_link_last:.)

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

```

3162 \cs_new_protected:Npn \_pdfannot_backend_link_margin:n #1
3163 {
3164 <*luatex>
3165 \tex_pdfvariable:D linkmargin
3166 </luatex>
3167 <*pdftex>
3168 \tex_pdflinkmargin:D
3169 </pdftex>
3170 \dim_eval:n {#1} \scan_stop:
3171 }

```

(End of definition for _pdfannot_backend_link_margin:n.)

_pdfannot_backend_link_on: Separate definitions for the two engines.

```

\_pdfannot_backend_link_off: 3172 \cs_new_protected:Npn \_pdfannot_backend_link_on:
3173 <*luatex>
3174 { \tex_pdfextension:D linkstate 0 ~ }
3175 </luatex>
3176 <*pdftex>
3177 { \tex_pdfrunninglinkon:D }
3178 </pdftex>
3179 \cs_new_protected:Npn \_pdfannot_backend_link_off:
3180 <*luatex>
3181 { \tex_pdfextension:D linkstate 1 ~ }
3182 </luatex>
3183 <*pdftex>
3184 { \tex_pdfrunninglinkoff:D }
3185 </pdftex>

```

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

3186 `</luatex | pdftex>`

7.3 dvipdfmx backend

3187 `<*dvipdfmx | xetex>`

`\_pdfannot_backend:n`
`\_pdfannot_backend:e`

A generic function for the backend PDF specials

3188 `\cs_new_protected:Npe \_pdfannot_backend:n #1`
3189 `{ \_kernel_backend_literal:n { pdf: #1 } }`
3190 `\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.

3191 `\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.

3192 `\cs_new_protected:Npn \_pdfannot_backend_generic:nnnn #1#2#3#4`
3193 `{`
3194 `\int_gincr:N \g_pdfannot_backend_int`
3195 `\_pdfannot_backend:e`
3196 `{`
3197 `ann ~ @pdfannot \int_use:N \g_pdfannot_backend_int \c_space_tl`
3198 `width ~ \dim_eval:n {#1} ~`
3199 `height ~ \dim_eval:n {#2} ~`
3200 `depth ~ \dim_eval:n {#3} ~`
3201 `<< /Type /Annot #4 >>`
3202 `}`
3203 `}`

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

`\_pdfannot_backend_last:`

3204 `\cs_new:Npn \_pdfannot_backend_last:`
3205 `{ @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.

3206 `\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`

`\_pdfannot_backend_link_begin:n`
`\_pdfannot_backend_link_end:`

3207 `\cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nnw #1#2`
3208 `{`
3209 `\_pdfannot_backend_link_begin:n`
3210 `{ #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> }`
3211 `}`
3212 `\cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nnw #1#2`
3213 `{ \_pdfannot_backend_link_begin:n {#1#2} }`
3214 `\cs_new_protected:Npe \_pdfannot_backend_link_begin:n #1`

```

3215 {
3216   \int_gincr:N \exp_not:N \g__pdfannot_backend_int
3217   \int_gset_eq:NN \exp_not:N \g__pdfannot_backend_link_int
3218   \exp_not:N \g__pdfannot_backend_int
3219   \__pdfannot_backend:e
3220   {
3221     bann ~
3222     @pdfannot
3223     \exp_not:N \int_use:N \exp_not:N \g__pdfannot_backend_link_int
3224     \c_space_tl
3225     <<
3226     /Type /Annot
3227     #1
3228     >>
3229   }
3230 }
3231 \cs_new_protected:Npn \__pdfannot_backend_link_end:
3232 { \__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.

```

3233 \cs_new:Npn \__pdfannot_backend_link_last:
3234 { @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.

```

3235 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1
3236 { \__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:

```

3237 \cs_new_protected:Npn \__pdfannot_backend_link_on: { \__pdfannot_backend:n { link } }
3238 \cs_new_protected:Npn \__pdfannot_backend_link_off: { \__pdfannot_backend:n { nolink } }

```

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

```

3239 </dvipdfmx | xetex>

```

7.4 dvisvgm backend

```

3240 <*dvisvgm>

```

_pdfannot_backend_generic:nnnn

```

3241 \cs_new_protected:Npn \__pdfannot_backend_generic:nnnn #1#2#3#4 { }

```

(End of definition for __pdfannot_backend_generic:nnnn.)

__pdfannot_backend_last:

```

3242 \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:
3243 \cs_new_protected:Npn \__pdfannot_backend_link_begin_goto:nnw #1#2 { }
3244 \cs_new_protected:Npn \__pdfannot_backend_link_begin_user:nnw #1#2 { }
3245 \cs_new_protected:Npn \__pdfannot_backend_link_begin:nnnw #1#2#3 { }
3246 \cs_new_protected:Npn \__pdfannot_backend_link_end: { }

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

\__pdfannot_backend_link_last:
3247 \cs_new:Npe \__pdfannot_backend_link_last: { }

(End of definition for \__pdfannot_backend_link_last:.)

\__pdfannot_backend_link_margin:n
3248 \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:
3249 \cs_new_protected:Npn \__pdfannot_backend_link_on: { }
3250 \cs_new_protected:Npn \__pdfannot_backend_link_off: { }

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

3251 </dvisvgm>

```

7.5 Transitional code

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

```

3252 <@@=)

3253 \cs_new_eq:NN \__pdf_backend_annotation:nnnn \__pdfannot_backend_generic:nnnn
3254 \cs_new_eq:NN \__pdf_backend_annotation_last: \__pdfannot_backend_last:
3255 \clist_map_inline:nn
3256 {
3257   begin_goto:nnw ,
3258   begin_user:nnw ,
3259   begin:nnnw ,
3260   end: ,
3261   last: ,
3262   margin:n
3263 }
3264 { \cs_new_eq:cc { __pdf_backend_link_ #1 } { __pdfannot_backend_link_ #1 } }

3265 </package>

```

8 l3backend-opacity implementation

```
3266 <*package>
3267 <@@=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.

```
3268 <*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.

```

__opacity_backend_select:n
  __opacity_backend_fill:n
__opacity_backend_stroke:n
  __opacity_backend:nnn
  __opacity_backend_reset:
    __opacity_backend_reset_fill:
    __opacity_backend_reset_stroke:
3269 \cs_new_protected:Npn __opacity_backend_select:n #1
3270 {
3271   __opacity_backend:nnn {#1} { fill } { ca }
3272   __opacity_backend:nnn {#1} { stroke } { CA }
3273 }
3274 \cs_new_protected:Npn __opacity_backend_fill:n #1
3275 {
3276   __opacity_backend:nnn
3277   { #1 }
3278   { fill }
3279   { ca }
3280 }
3281 \cs_new_protected:Npn __opacity_backend_stroke:n #1
3282 {
3283   __opacity_backend:nnn
3284   { #1 }
3285   { stroke }
3286   { CA }
3287 }
3288 \cs_new_protected:Npn __opacity_backend:nnn #1#2#3
3289 {
3290   __kernel_backend_postscript:n
3291   {
3292     product ~ (Ghostscript) ~ search
3293     {
3294       pop ~ pop ~ pop ~
3295       #1 ~ .set #2 constantalpha
3296     }
3297     {
3298       pop ~
3299       mark ~
3300       /#3 ~ #1
3301       /SetTransparency ~
3302       pdfmark
3303     }
3304     ifelse
3305   }

```

```

3306 }
3307 \cs_new_protected:Npn \__opacity_backend_reset:
3308 {
3309   \__opacity_backend_reset_fill:
3310   \__opacity_backend_reset_stroke:
3311 }
3312 \cs_new_protected:Npn \__opacity_backend_reset_fill:
3313 {
3314   \__opacity_backend:nnn
3315   { 1 }
3316   { fill }
3317   { ca }
3318 }
3319 \cs_new_protected:Npn \__opacity_backend_reset_stroke:
3320 {
3321   \__opacity_backend:nnn
3322   { 1 }
3323   { stroke }
3324   { CA }
3325 }

```

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

```

3326 </dvips>
3327 <*dvipdfmx | luatex | pdftex | xetex>

```

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

```

3328 \bool_lazy_and:nnT
3329 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3330 { \pdfmanagement_if_active_p: }
3331 {
3332   <*luatex | pdftex>
3333   \__kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3334   { page ~ direct } { /opacity 1 ~ gs }
3335   </luatex | pdftex>
3336   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3337   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3338 }

```

(End of definition for \c__opacity_backend_stack_int.)

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

```

3339 \tl_new:N \l__opacity_backend_fill_tl
3340 \tl_new:N \l__opacity_backend_stroke_tl
3341 \tl_set:Nn \l__opacity_backend_fill_tl { 1 }
3342 \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.

```

3343 \cs_new_protected:Npn \__opacity_backend_select:n #1
3344 {
3345   \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3346   \tl_set:Nn \l__opacity_backend_stroke_tl {#1}

```

```

3347 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3348 { opacity #1 }
3349 { << /ca ~ #1 /CA ~ #1 >> }
3350 <*dviptfm | xetex>
3351 \__kernel_backend_literal_pdf:n
3352 </dviptfm | xetex>
3353 <*luatex | pdftex>
3354 \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3355 </luatex | pdftex>
3356 { /opacity #1 ~ gs }
3357 }
3358 \cs_new_protected:Npn \__opacity_backend_reset:
3359 {
3360 <*dviptfm | xetex>
3361 \__kernel_backend_literal_pdf:n
3362 { /opacity1 ~ gs }
3363 </dviptfm | xetex>
3364 <*luatex | pdftex>
3365 \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int
3366 </luatex | pdftex>
3367 }
3368 \cs_new_eq:NN \__opacity_backend_reset_fill: \__opacity_backend_reset:
3369 \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
__opacity_backend_stroke:n stick to a single setting.

```

\__opacity_backend_fill_stroke:nn
3370 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3371 {
3372 \exp_args:Nno \__opacity_backend_fill_stroke:nn
3373 { #1 }
3374 { \l__opacity_backend_stroke_tl }
3375 }
3376 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3377 {
3378 \exp_args:No \__opacity_backend_fill_stroke:nn
3379 { \l__opacity_backend_fill_tl }
3380 { #1 }
3381 }
3382 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3383 {
3384 \str_if_eq:nnTF {#1} {#2}
3385 { \__opacity_backend_select:n {#1} }
3386 {
3387 \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3388 \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3389 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3390 { opacity.fill #1 }
3391 { << /ca ~ #1 >> }
3392 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3393 { opacity.stroke #2 }
3394 { << /CA ~ #2 >> }
3395 <*dviptfm | xetex>

```

```

3396         \__kernel_backend_literal_pdf:n
3397     </dvipdfmx | xetex>
3398     <*luatex | pdftex>
3399         \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3400     </luatex | pdftex>
3401         { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3402     }
3403 }

```

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

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

```

3404 \bool_lazy_and:nnF
3405 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3406 { \pdfmanagement_if_active_p: }
3407 {
3408     \cs_gset_protected:Npn \__opacity_backend_select:n #1 { }
3409     \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2 { }
3410     \cs_gset_protected:Npn \__opacity_backend_reset: { }
3411     \cs_gset_eq:NN \__opacity_backend_reset_fill: \__opacity_backend_reset:
3412     \cs_gset_eq:NN \__opacity_backend_reset_stroke: \__opacity_backend_reset:
3413 }

```

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

```

3414 </dvipdfmx | luatex | pdftex | xetex>
3415 <*dvisvgm>

```

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

```

3416 \cs_new_protected:Npn \__opacity_backend_select:n #1
3417     { \__opacity_backend:nn {#1} { } }
3418 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3419     { \__opacity_backend:nn {#1} { fill- } }
3420 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3421     { \__opacity_backend:nn {#1} { stroke- } }
3422 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3423     { \__kernel_backend_scope:e { #2 opacity = " #1 " } }
3424 \cs_new_protected:Npn \__opacity_backend_reset: { }
3425 \cs_new_eq:NN \__opacity_backend_reset_fill: \__opacity_backend_reset:
3426 \cs_new_eq:NN \__opacity_backend_reset_stroke: \__opacity_backend_reset:

```

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

```

3427 </dvisvgm>
3428 </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.

```

3429 <*lua>

```

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

```

3430 local pdfmanagement_active do
3431   local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3432   local cmd = pdfmanagement_if_active_p.cmdname
3433   if cmd == 'undefined_cs' then
3434     pdfmanagement_active = false
3435   else
3436     token.put_next(pdfmanagement_if_active_p)
3437     pdfmanagement_active = token.scan_int() ~= 0
3438   end
3439 end
3440
3441 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3442   luaotfload.set_transparent_colorstack(function() return token.create'c__opacity_backend_st
3443
3444   local transparent_register = {
3445     token.create'pdfmanagement_add:nnn',
3446     token.new(0, 1),
3447     'Page/Resources/ExtGState',
3448     token.new(0, 2),
3449     token.new(0, 1),
3450     '',
3451     token.new(0, 2),
3452     token.new(0, 1),
3453     '<</ca ',
3454     '',
3455     '/CA ',
3456     '',
3457     '>>',
3458     token.new(0, 2),
3459   }
3460   luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)
3461     value = (octet * -1):match(value)
3462     if not value then
3463       tex.error'Invalid transparency value'
3464       return
3465     end
3466     value = value:sub(1, -2)
3467     local result = 'opacity' .. value
3468     tex.runtoks(function()
3469       transparent_register[6], transparent_register[10], transparent_register[12] = result,
3470       tex.sprint(-2, transparent_register)
3471     end)
3472     return '/' .. result .. ' gs'
3473   end, 'l3opacity')
3474 end
3475 </lua>

```

9 l3backend-header implementation

```

3476 <*dvips & header>

```

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

```
3477 /color.sc { } def
```

(End of definition for color.sc.)

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

```
3478 TeXDict begin
3479 /TeXcolorseparation { setcolor } def
3480 end
```

(End of definition for TeXcolorseparation and separation.)

pdf.globaldict A small global dictionary for backend use.

```
3481 true setglobal
3482 /pdf.globaldict 4 dict def
3483 false setglobal
```

(End of definition for pdf.globaldict.)

pdf.cvs 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.

```
3484 /pdf.cvs { 65534 string cvs } def
3485 /pdf.dvi.pt { 72.27 mul Resolution div } def
3486 /pdf.pt.dvi { 72.27 div Resolution mul } def
3487 /pdf.rect.ht { dup 1 get neg exch 3 get add } def
```

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

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

```
3488 /pdf.linkmargin { 1 pdf.pt.dvi } def
3489 /pdf.linkdp.pad { 0 } def
3490 /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 separate links for generic annotations to support adding a margin and setting a minimal size.

```
3491 /pdf.rect
3492 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
3493 /pdf.save.ll
3494 {
3495   currentpoint
3496   /pdf.lly exch def
3497   /pdf.llx exch def
3498 }
3499 def
3500 /pdf.save.ur
3501 {
3502   currentpoint
3503   /pdf.ury exch def
3504   /pdf.urx exch def
3505 }
3506 def
```

```

3507 /pdf.save.linkll
3508 {
3509     currentpoint
3510     pdf.linkmargin add
3511     pdf.linkdp.pad add
3512     /pdf.lly exch def
3513     pdf.linkmargin sub
3514     /pdf.llx exch def
3515 }
3516 def
3517 /pdf.save.linkur
3518 {
3519     currentpoint
3520     pdf.linkmargin sub
3521     pdf.linkht.pad sub
3522     /pdf.ury exch def
3523     pdf.linkmargin add
3524     /pdf.urx exch def
3525 }
3526 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    3527 /pdf.dest.anchor
pdf.dev.y    3528 {
pdf.tmpa    3529     currentpoint exch
pdf.tmpb    3530     pdf.dvi.pt 72 add
pdf.tmpc    3531     /pdf.dest.x exch def
pdf.tmpd    3532     pdf.dvi.pt
3533     vsize 72 sub exch sub
3534     /pdf.dest.y exch def
3535 }
3536 def
3537 /pdf.dest.point
3538 { pdf.dest.x pdf.dest.y } def
3539 /pdf.dest2device
3540 {
3541     /pdf.dest.y exch def
3542     /pdf.dest.x exch def
3543     matrix currentmatrix
3544     matrix defaultmatrix
3545     matrix invertmatrix
3546     matrix concatmatrix
3547     cvx exec
3548     /pdf.dev.y exch def
3549     /pdf.dev.x exch def
3550     /pdf.tmpd exch def
3551     /pdf.tmpc exch def
3552     /pdf.tmpb exch def

```

```

3553     /pdf.tmpa exch def
3554 pdf.dest.x pdf.tmpa mul
3555     pdf.dest.y pdf.tmpc mul add
3556     pdf.dev.x add
3557 pdf.dest.x pdf.tmpb mul
3558     pdf.dest.y pdf.tmpd mul add
3559     pdf.dev.y add
3560 }
3561 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.

```

3562 /pdf.bordertracking false def
3563 /pdf.bordertracking.begin
3564 {
3565     SDict /pdf.bordertracking true put
3566     SDict /pdf.leftboundary undef
3567     SDict /pdf.rightboundary undef
3568     /a where
3569     {
3570         /a
3571         {
3572             currentpoint pop
3573             SDict /pdf.rightboundary known dup
3574             {
3575                 SDict /pdf.rightboundary get 2 index lt
3576                 { not }
3577                 if
3578             }
3579             if
3580             { pop }
3581             { SDict exch /pdf.rightboundary exch put }
3582             ifelse
3583             moveto
3584             currentpoint pop
3585             SDict /pdf.leftboundary known dup
3586             {
3587                 SDict /pdf.leftboundary get 2 index gt
3588                 { not }
3589                 if
3590             }
3591             if
3592             { pop }
3593             { SDict exch /pdf.leftboundary exch put }
3594             ifelse
3595         }
3596         put
3597     }
3598     if
3599 }

```

```

3600     def
3601 /pdf.bordertracking.end
3602 {
3603     /a where { /a { moveto } put } if
3604     /x where { /x { 0 exch rmoveto } put } if
3605     SDict /pdf.leftboundary known
3606     { pdf.outerbox 0 pdf.leftboundary put }
3607     if
3608     SDict /pdf.rightboundary known
3609     { pdf.outerbox 2 pdf.rightboundary put }
3610     if
3611     SDict /pdf.bordertracking false put
3612 }
3613     def
3614 /pdf.bordertracking.endpage
3615 {
3616     pdf.bordertracking
3617     {
3618         pdf.bordertracking.end
3619         true setglobal
3620         pdf.globaldict
3621         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3622         pdf.globaldict
3623         /pdf.brokenlink.skip pdf.baselineskip put
3624         pdf.globaldict
3625         /pdf.brokenlink.dict
3626         pdf.link.dict pdf.cvs put
3627         false setglobal
3628         mark pdf.link.dict cvx exec /Rect
3629         [
3630             pdf.llx
3631             pdf.lly
3632             pdf.outerbox 2 get pdf.linkmargin add
3633             currentpoint exch pop
3634             pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3635         ]
3636         /ANN pdf.pdfmark
3637     }
3638     if
3639 }
3640     def
3641 /pdf.bordertracking.continue
3642 {
3643     /pdf.link.dict pdf.globaldict
3644     /pdf.brokenlink.dict get def
3645     /pdf.outerbox pdf.globaldict
3646     /pdf.brokenlink.rect get def
3647     /pdf.baselineskip pdf.globaldict
3648     /pdf.brokenlink.skip get def
3649     pdf.globaldict dup dup
3650     /pdf.brokenlink.dict undef
3651     /pdf.brokenlink.skip undef
3652     /pdf.brokenlink.rect undef
3653     currentpoint

```

```

3654 /pdf.originy exch def
3655 /pdf.originx exch def
3656 /a where
3657 {
3658   /a
3659   {
3660     moveto
3661     SDict
3662     begin
3663       currentpoint pdf.originy ne exch
3664       pdf.originx ne or
3665       {
3666         pdf.save.linkll
3667         /pdf.lly
3668         pdf.lly pdf.outerbox 1 get sub def
3669         pdf.bordertracking.begin
3670       }
3671       if
3672       end
3673     }
3674     put
3675   }
3676   if
3677   /x where
3678   {
3679     /x
3680     {
3681       0 exch rmoveto
3682       SDict
3683       begin
3684         currentpoint
3685         pdf.originy ne exch pdf.originx ne or
3686         {
3687           pdf.save.linkll
3688           /pdf.lly
3689           pdf.lly pdf.outerbox 1 get sub def
3690           pdf.bordertracking.begin
3691         }
3692         if
3693         end
3694       }
3695       put
3696     }
3697     if
3698   }
3699   def

```

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

<pre>pdf.breaklink pdf.breaklink.write pdf.count pdf.currentrect</pre>	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.
--	---

```

3700 /pdf.breaklink
3701 {
3702     pop
3703     counttomark 2 mod 0 eq
3704     {
3705         counttomark /pdf.count exch def
3706         {
3707             pdf.count 0 eq { exit } if
3708             counttomark 2 roll
3709             1 index /Rect eq
3710             {
3711                 dup 4 array copy
3712                 dup dup
3713                 1 get
3714                 pdf.outerbox pdf.rect.ht
3715                 pdf.linkmargin 2 mul add sub
3716                 3 exch put
3717                 dup
3718                 pdf.outerbox 2 get
3719                 pdf.linkmargin add
3720                 2 exch put
3721                 dup dup
3722                 3 get
3723                 pdf.outerbox pdf.rect.ht
3724                 pdf.linkmargin 2 mul add add
3725                 1 exch put
3726                 /pdf.currentrect exch def
3727                 pdf.breaklink.write
3728                 {
3729                     pdf.currentrect
3730                     dup
3731                     pdf.outerbox 0 get
3732                     pdf.linkmargin sub
3733                     0 exch put
3734                     dup
3735                     pdf.outerbox 2 get
3736                     pdf.linkmargin add
3737                     2 exch put
3738                     dup dup
3739                     1 get
3740                     pdf.baselineskip add
3741                     1 exch put
3742                     dup dup
3743                     3 get
3744                     pdf.baselineskip add
3745                     3 exch put
3746                     /pdf.currentrect exch def
3747                     pdf.breaklink.write
3748                 }
3749                 1 index 3 get
3750                 pdf.linkmargin 2 mul add
3751                 pdf.outerbox pdf.rect.ht add
3752                 2 index 1 get sub
3753                 pdf.baselineskip div round cvi 1 sub

```

```

3754         exch
3755     repeat
3756     pdf.currentrect
3757     dup
3758         pdf.outerbox 0 get
3759         pdf.linkmargin sub
3760         0 exch put
3761     dup dup
3762         1 get
3763         pdf.baselineskip add
3764         1 exch put
3765     dup dup
3766         3 get
3767         pdf.baselineskip add
3768         3 exch put
3769     dup 2 index 2 get 2 exch put
3770     /pdf.currentrect exch def
3771     pdf.breaklink.write
3772     SDict /pdf.pdfmark.good false put
3773     exit
3774 }
3775 { pdf.count 2 sub /pdf.count exch def }
3776 ifelse
3777 }
3778 loop
3779 }
3780 if
3781 /ANN
3782 }
3783 def
3784 /pdf.breaklink.write
3785 {
3786     counttomark 1 sub
3787     index /_objdef eq
3788     {
3789         counttomark -2 roll
3790         dup wcheck
3791         {
3792             readonly
3793             counttomark 2 roll
3794         }
3795         { pop pop }
3796     } ifelse
3797 }
3798 if
3799 counttomark 1 add copy
3800 pop pdf.currentrect
3801 /ANN pdfmark
3802 }
3803 def

```

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

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

function. Only mark types which are known are altered. At present, this is purely ANN marks, which are measured relative to the size of the baseline skip. If they are more than one apparent line high, breaking is applied.

```

3804 /pdf.pdfmark
3805 {
3806   SDict /pdf.pdfmark.good true put
3807   dup /ANN eq
3808   {
3809     pdf.pdfmark.store
3810     pdf.pdfmark.dict
3811     begin
3812       Subtype /Link eq
3813       currentdict /Rect known and
3814       SDict /pdf.outerbox known and
3815       SDict /pdf.baselineskip known and
3816       {
3817         Rect 3 get
3818         pdf.linkmargin 2 mul add
3819         pdf.outerbox pdf.rect.ht add
3820         Rect 1 get sub
3821         pdf.baselineskip div round cvi 0 gt
3822         { pdf.breaklink }
3823         if
3824       }
3825       if
3826     end
3827     SDict /pdf.outerbox undef
3828     SDict /pdf.baselineskip undef
3829     currentdict /pdf.pdfmark.dict undef
3830   }
3831   if
3832   pdf.pdfmark.good
3833   { pdfmark }
3834   { cleartomark }
3835   ifelse
3836 }
3837 def
3838 /pdf.pdfmark.store
3839 {
3840   /pdf.pdfmark.dict 65534 dict def
3841   counttomark 1 add copy
3842   pop
3843   {
3844     dup mark eq
3845     {
3846       pop
3847       exit
3848     }
3849     {
3850       pdf.pdfmark.dict
3851       begin def end
3852     }
3853   ifelse
3854 }

```

```
3855     loop
3856 }
3857 def
(End of definition for pdf.pdfmark and others.)
3858 </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>	1150
A	
<code>\AtBeginDvi</code>	56
B	
bool commands:	
<code>\bool_gset_false:N</code>	1236, 1255, 1278, 1300, 1316, 1425, 1677, 1713, 2951, 2997
<code>\bool_gset_true:N</code>	1234, 1303, 1423, 1692, 2944, 2950
<code>\bool_if:NTF</code>	66, 603, 1246, 1250, 1266, 1269, 1273, 1284, 1291, 1295, 1307, 1311, 1436, 1441, 1446, 1651, 1696, 1825, 1877, 1879, 2018, 2063, 2065, 2939, 2954, 2959, 2964
<code>\bool_if:nTF</code>	2442, 2623, 2811
<code>\bool_lazy_and:nnTF</code>	819, 2170, 3328, 3404
<code>\bool_lazy_any:nTF</code>	1865, 2053
<code>\bool_new:N</code>	1237, 1304, 1426, 1693, 1815, 1981, 2924, 2925
<code>\bool_set_false:N</code>	1820, 1838, 1977, 1997, 2088, 2238
<code>\bool_set_true:N</code>	1837, 2005
box commands:	
<code>\box_dp:N</code>	235, 237, 285, 287, 342, 344, 391, 393, 395, 397, 2976, 3009, 3010, 3035
<code>\box_ht:N</code>	237, 287, 344, 395, 397, 1892, 2129, 2981, 3020, 3021, 3037
<code>\box_if_empty:N</code>	3069
<code>\box_move_down:nn</code>	2897, 2976
<code>\box_move_up:nn</code>	2259, 2899, 2981
<code>\box_new:N</code>	2887, 2888
<code>\box_set_dp:Nn</code>	1784
<code>\box_set_ht:Nn</code>	1783
<code>\box_set_wd:Nn</code>	299, 1782
<code>\box_use:N</code>	242, 260, 274, 290, 317, 331, 347, 363, 375, 426, 440, 459, 1376, 1584, 1785, 2929
<code>\box_wd:N</code>	236, 244, 286, 292, 343, 349, 392, 394, 1891, 2128
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>	1324, 1456, 1720
<code>\clist_map_inline:nn</code>	3255
color internal commands:	
<code>__color_backend:nnn</code>	1057, 1072, 1080, 1086
<code>\g__color_backend_colorant_prop</code>	566, 585, 588, 611, 855
<code>__color_backend_devicen_colorants:n</code>	567, 567, 772, 913
<code>__color_backend_devicen_colorants:w</code>	567, 575, 582, 590
<code>__color_backend_devicen_init:nnn</code>	759, 759, 880, 880, 1107, 1107
<code>__color_backend_devicen_init:w</code>	880, 889, 918, 922
<code>__color_backend_fill:n</code>	1037
<code>__color_backend_fill:nN</code>	959, 959, 961, 962, 963, 985, 986, 988, 990, 991, 1010, 1019, 1020, 1022, 1024, 1025, 1046, 1047, 1049, 1051, 1052
<code>__color_backend_fill_cmyk:n</code>	1059
<code>__color_backend_fill_cmyk:nN</code>	959, 961, 985, 985, 1019, 1019, 1046, 1046
<code>__color_backend_fill_devicen:nnN</code>	969, 979, 1009, 1013, 1036, 1040, 1101, 1103
<code>__color_backend_fill_gray:nN</code>	959, 962, 985, 987, 1019, 1021, 1046, 1048
<code>__color_backend_fill_reset:</code>	981, 981, 1015, 1015, 1042, 1042, 1105, 1105
<code>__color_backend_fill_rgb:nN</code>	959, 963, 985, 989, 1019, 1023, 1046, 1050

_color_backend_fill_separation:nnN	_color_backend_separation_-
.. 969, 969, 979, 1009, 1009, 1013,	init_/DeviceGray:nnn 601
1036, 1036, 1040, 1101, 1101, 1103	_color_backend_separation_-
\l_color_backend_fill_tl 524, 536, 993, 1006	init_/DeviceRGB:nnn 601
_color_backend_iccbased_-	_color_backend_separation_-
device:nnn 942, 942	init_aux:nnnnnn 601, 607, 623
_color_backend_iccbased_-	_color_backend_separation_-
init:nnn 778, 778, 924, 924, 1107, 1108	init_CIELAB:nnn 601, 713, 783, 833, 858
_color_backend_init_resource:n	_color_backend_separation_-
..... 816, 816, 845, 916, 940, 955	init_CIELAB:nnnnnn 784
_color_backend_reset: 506, 520, 528, 541, 545, 562,	_color_backend_separation_-
981, 982, 1015, 1016, 1042, 1061, 1105	init_count:n 601, 660, 663
_color_backend_rgb:w 1074	_color_backend_separation_-
_color_backend_select:n 545, 545, 550, 555, 560	init_count:w ... 601, 664, 665, 669
_color_backend_select:nN 506, 507, 509, 511, 512, 597	_color_backend_separation_-
_color_backend_select:nnN 528, 529, 531, 533, 534, 812	init_Device:Nn 601, 645, 647, 649, 650
_color_backend_select_cmyk:nN .	\l_color_backend_stack_int 467, 539, 542, 995, 1007
..... 506, 506, 528, 528, 545, 547	_color_backend_stroke:nN 959, 964, 966, 967,
_color_backend_select_devicen:nnN	968, 985, 998, 1000, 1002, 1003, 1012
..... 594, 599, 781, 782, 803, 814	_color_backend_stroke_cmyk:nN .
_color_backend_select_gray:nN .	 959,
..... 506, 508, 528, 530, 545, 552	966, 985, 997, 1019, 1030, 1057, 1057
_color_backend_select_iccbased:nn	_color_backend_stroke_devicen:nn
..... 785, 785	 1036
_color_backend_select_iccbased:nnN	_color_backend_stroke_devicen:nnN
..... 600, 600, 803, 815	969, 980, 1009, 1014, 1041, 1101, 1104
_color_backend_select_rgb:nN ..	_color_backend_stroke_gray:nN .
..... 506, 510, 528, 532, 545, 557	 959,
_color_backend_select_separation:nnN	967, 985, 999, 1019, 1032, 1057, 1063
..... 594, 594, 599,	_color_backend_stroke_gray_-
781, 781, 782, 803, 804, 811, 814, 815	aux:n 1057, 1067, 1071
_color_backend_separation_-	_color_backend_stroke_reset: ..
init:n 601, 682, 695	 981,
_color_backend_separation_-	982, 1015, 1016, 1042, 1043, 1105, 1106
init:nn 833, 843, 847	_color_backend_stroke_rgb:nN ..
_color_backend_separation_-	 959,
init:nnn 601, 636, 657	968, 985, 1001, 1019, 1034, 1057, 1073
_color_backend_separation_-	_color_backend_stroke_rgb:w ...
init:nnnn 601, 659, 671	 1057, 1075
_color_backend_separation_-	_color_backend_stroke_separation:nnN
init:nnnnn 601,	.. 969, 974, 980, 1009, 1011, 1014,
601, 622, 715, 783, 783, 833, 833, 873	1036, 1038, 1041, 1101, 1102, 1104
_color_backend_separation_-	\l_color_backend_stroke_tl 524, 537, 994, 1005
init:nw 601, 686, 697, 711	\g_color_model_int 608, 617, 765,
_color_backend_separation_-	793, 845, 851, 852, 906, 907, 916, 940
init:w 601, 673, 688, 693	\c_color_model_range_CIELAB_tl .
_color_backend_separation_-	 720, 755, 869, 876
init_/DeviceCMYK:nnn 601	\l_color_tmp_tl 597
	color.sc 3477

cs commands:

`\cs_generate_variant:Nn` . . . 62, 65,
 170, 181, 212, 218, 622, 1182, 1593,
 2032, 2099, 2119, 2286, 2301, 2364,
 2574, 2587, 2697, 2712, 2742, 3190
`\cs_gset:Npe` . . . 2454, 2458, 2816, 2821
`\cs_gset_eq:NN` 3411, 3412
`\cs_gset_protected:Npn`
 3408, 3409, 3410
`\cs_if_exist:NTF`
 27, 49, 2648, 2674, 3065
`\cs_if_exist_p:N` 820, 3329, 3405
`\cs_if_exist_use:NTF` 38, 635
`\cs_new:Npe`
 567, 2588, 2599, 2666, 3116, 3151, 3247
`\cs_new:Npn` 582, 644, 646,
 648, 650, 657, 663, 665, 671, 688,
 695, 697, 918, 1329, 1461, 1724,
 1894, 2132, 2276, 2293, 2365, 2367,
 2460, 2461, 2543, 2544, 2556, 2575,
 2576, 2679, 2705, 2743, 2745, 2824,
 2825, 2837, 2838, 2843, 2844, 2849,
 2850, 2919, 3090, 3204, 3233, 3242
`\cs_new_eq:NN` 46, 56, 58, 599, 782,
 814, 815, 961, 962, 963, 966, 967,
 968, 979, 980, 981, 982, 1013, 1014,
 1015, 1016, 1040, 1041, 1042, 1103,
 1104, 1105, 1181, 1385, 1386, 1391,
 1392, 1592, 1594, 1595, 1601, 1795,
 1796, 1808, 1809, 1832, 1833, 1900,
 1901, 1902, 1925, 1950, 1962, 1963,
 1971, 1972, 1973, 1994, 2000, 2001,
 2002, 2072, 2082, 2083, 2084, 2225,
 2226, 2233, 2234, 2243, 2273, 2274,
 2275, 2278, 2294, 2706, 2929, 3253,
 3254, 3264, 3368, 3369, 3425, 3426
`\cs_new_protected:Npe`
 . . . 601, 1086, 2638, 2695, 3188, 3214
`\cs_new_protected:Npn` 47,
 53, 60, 63, 71, 77, 82, 84, 88, 98,
 108, 118, 128, 137, 146, 156, 168,
 171, 173, 175, 179, 184, 193, 203,
 213, 224, 246, 248, 263, 279, 294,
 296, 322, 336, 351, 353, 366, 380,
 430, 443, 470, 484, 494, 506, 508,
 510, 512, 520, 528, 530, 532, 534,
 541, 545, 547, 552, 557, 562, 594,
 600, 623, 713, 759, 778, 781, 783,
 784, 785, 804, 811, 816, 833, 847,
 858, 880, 924, 942, 959, 964, 969,
 974, 985, 987, 989, 991, 997, 999,
 1001, 1003, 1009, 1011, 1019, 1021,
 1023, 1025, 1030, 1032, 1034, 1036,
 1038, 1043, 1046, 1048, 1050, 1052,

1057, 1063, 1071, 1073, 1075, 1101,
 1102, 1106, 1107, 1108, 1183, 1189,
 1194, 1196, 1198, 1206, 1214, 1223,
 1233, 1235, 1238, 1240, 1257, 1262,
 1280, 1302, 1305, 1318, 1331, 1336,
 1338, 1340, 1342, 1344, 1346, 1348,
 1350, 1355, 1360, 1387, 1389, 1393,
 1398, 1403, 1413, 1422, 1424, 1427,
 1429, 1431, 1433, 1438, 1443, 1448,
 1450, 1463, 1468, 1470, 1472, 1474,
 1476, 1478, 1480, 1482, 1501, 1525,
 1531, 1543, 1555, 1567, 1574, 1596,
 1602, 1607, 1612, 1623, 1633, 1643,
 1645, 1647, 1649, 1680, 1682, 1687,
 1689, 1691, 1694, 1715, 1726, 1739,
 1741, 1743, 1745, 1747, 1749, 1751,
 1753, 1755, 1763, 1771, 1797, 1816,
 1834, 1849, 1854, 1862, 1895, 1908,
 1926, 1936, 1952, 1965, 1974, 1983,
 1995, 2003, 2008, 2023, 2033, 2076,
 2085, 2091, 2097, 2100, 2107, 2120,
 2125, 2133, 2146, 2180, 2211, 2212,
 2214, 2216, 2218, 2224, 2227, 2235,
 2241, 2244, 2246, 2257, 2284, 2287,
 2289, 2291, 2295, 2302, 2319, 2324,
 2329, 2334, 2344, 2349, 2357, 2369,
 2395, 2400, 2428, 2440, 2452, 2456,
 2462, 2464, 2468, 2491, 2505, 2515,
 2526, 2545, 2577, 2610, 2621, 2627,
 2655, 2689, 2691, 2698, 2700, 2703,
 2707, 2713, 2718, 2723, 2725, 2727,
 2735, 2747, 2769, 2774, 2807, 2809,
 2814, 2819, 2826, 2828, 2832, 2833,
 2834, 2835, 2836, 2839, 2840, 2841,
 2842, 2845, 2846, 2847, 2848, 2851,
 2852, 2855, 2874, 2881, 2890, 2895,
 2928, 2930, 2935, 2937, 2942, 2957,
 2962, 2999, 3028, 3047, 3056, 3092,
 3099, 3100, 3103, 3127, 3129, 3131,
 3142, 3162, 3172, 3179, 3192, 3207,
 3212, 3231, 3235, 3237, 3238, 3241,
 3243, 3244, 3245, 3246, 3248, 3249,
 3250, 3269, 3274, 3281, 3288, 3307,
 3312, 3319, 3343, 3358, 3370, 3376,
 3382, 3416, 3418, 3420, 3422, 3424
`\cs_set_eq:NN` 3086, 3088
`\cs_set_protected:Npn` 2184

D

dim commands:

`\dim_compare:nNnTF` 2160, 2165
`\dim_compare_p:nNn` 2171, 2172
`\dim_eval:n`
 . . . 2398, 2501, 2502, 2503, 2772,

2863, 2864, 2867, 2893, 3111, 3112,
 3113, 3170, 3198, 3199, 3200, 3236
 \dim_gset:Nn 2876, 2877
 \dim_horizontal:n
 .. 2408, 2415, 2786, 2805, 2903, 2905
 \dim_max:nn 3007, 3018
 \dim_set:Nn
 .. 1891, 1892, 2128, 2129, 2156, 2157
 \dim_set_eq:NN 2222
 \dim_to_decimal:n .. 391, 392, 393,
 394, 395, 397, 1605, 1610, 1616,
 1617, 1618, 1619, 1628, 1629, 1630,
 1721, 1740, 2266, 2267, 3005, 3016,
 3034, 3035, 3036, 3037, 3041, 3096
 \dim_to_decimal_in_bp:n
 235, 236, 237, 285, 286, 287,
 342, 343, 344, 1202, 1203, 1210,
 1211, 1218, 1219, 1227, 1228, 1229,
 1326, 1330, 1334, 1396, 1401, 1407,
 1408, 1409, 1417, 1418, 1458, 1462,
 1466, 1725, 1802, 1803, 1804, 1805,
 1988, 1989, 1990, 1991, 2047, 2048,
 2049, 2050, 2251, 2252, 2253, 2254
 \dim_vertical:n 2404, 2411, 2778, 2789
 \dim_zero:N 2154, 2155
 \c_max_dim
 .. 2156, 2157, 2160, 2165, 2171, 2172
 draw internal commands:
 __draw_backend_add_to_path:n ...
 1602,
 1604, 1609, 1614, 1625, 1633, 1648
 __draw_backend_begin:
 .. 1183, 1183, 1387, 1387, 1596, 1596
 __draw_backend_box_use:Nnnnn ...
 .. 1360, 1360, 1574, 1574, 1771, 1771
 __draw_backend_cap_but:
 .. 1318, 1338, 1450, 1470, 1715, 1743
 __draw_backend_cap_rectangle: ..
 .. 1318, 1342, 1450, 1474, 1715, 1747
 __draw_backend_cap_round:
 .. 1318, 1340, 1450, 1472, 1715, 1745
 __draw_backend_clip:
 .. 1238, 1302, 1427, 1443, 1647, 1691
 __draw_backend_closepath:
 1238, 1238,
 1259, 1427, 1427, 1647, 1647, 1684
 __draw_backend_closestroke: ...
 .. 1238, 1257, 1427, 1431, 1647, 1682
 __draw_backend_curveto:nnnnnn ..
 .. 1198, 1223, 1393, 1403, 1602, 1623
 __draw_backend_dash:n
 1318, 1324, 1329,
 1450, 1456, 1461, 1715, 1720, 1724
 __draw_backend_dash_aux:nn
 1715, 1719, 1726
 __draw_backend_dash_pattern:nn ..
 .. 1318, 1318, 1450, 1450, 1715, 1715
 __draw_backend_discardpath: ...
 .. 1238, 1305, 1427, 1448, 1647, 1694
 __draw_backend_end:
 .. 1183, 1189, 1387, 1389, 1596, 1601
 __draw_backend_evenodd_rule: ...
 .. 1233, 1233, 1422, 1422, 1643, 1643
 __draw_backend_fill:
 .. 1238, 1262, 1427, 1433, 1647, 1687
 __draw_backend_fillstroke:
 .. 1238, 1280, 1427, 1438, 1647, 1689
 __draw_backend_join_bevel:
 .. 1318, 1348, 1450, 1480, 1715, 1753
 __draw_backend_join_miter:
 .. 1318, 1344, 1450, 1476, 1715, 1749
 __draw_backend_join_round:
 .. 1318, 1346, 1450, 1478, 1715, 1751
 __draw_backend_lineto:nn
 .. 1198, 1206, 1393, 1398, 1602, 1607
 __draw_backend_linewidth:n
 .. 1318, 1331, 1450, 1463, 1715, 1739
 __draw_backend_literal:n
 1181, 1181, 1182, 1185, 1186, 1187,
 1191, 1192, 1195, 1197, 1200, 1208,
 1216, 1225, 1239, 1242, 1243, 1244,
 1245, 1248, 1254, 1264, 1271, 1277,
 1282, 1287, 1288, 1289, 1290, 1293,
 1299, 1309, 1315, 1320, 1333, 1337,
 1339, 1341, 1343, 1345, 1347, 1349,
 1352, 1357, 1362, 1363, 1364, 1365,
 1366, 1367, 1368, 1369, 1370, 1374,
 1375, 1377, 1378, 1379, 1380, 1381,
 1385, 1385, 1386, 1395, 1400, 1405,
 1415, 1428, 1430, 1432, 1435, 1440,
 1445, 1449, 1452, 1465, 1469, 1471,
 1473, 1475, 1477, 1479, 1481, 1527,
 1592, 1592, 1593, 1654, 1673, 1699
 __draw_backend_miterlimit:n ...
 .. 1318, 1336, 1450, 1468, 1715, 1741
 __draw_backend_moveto:nn
 .. 1198, 1198, 1393, 1393, 1602, 1602
 __draw_backend_nonzero_rule: ...
 .. 1233, 1235, 1422, 1424, 1643, 1645
 __draw_backend_path:n
 1647, 1649, 1681, 1688, 1690
 \g__draw_backend_path_int 1662, 1679
 \g__draw_backend_path_tl
 .. 1602, 1658, 1674, 1676, 1703, 1712
 __draw_backend_rectangle:nnnn ..
 .. 1198, 1214, 1393, 1413, 1602, 1612

`\__draw_backend_scope_begin:` [1194](#),
[1194](#), [1388](#), [1391](#), [1391](#), [1594](#), [1594](#)
`\__draw_backend_scope_end:` [1194](#),
[1196](#), [1390](#), [1391](#), [1392](#), [1594](#), [1595](#)
`\__draw_backend_shift:nn`
... [1350](#), [1355](#), [1482](#), [1525](#), [1755](#), [1763](#)
`\__draw_backend_stroke:` [1238](#), [1240](#),
[1260](#), [1427](#), [1429](#), [1647](#), [1680](#), [1685](#)
`\__draw_backend_transform:nnnn` ..
..... [1350](#), [1350](#), [1371](#), [1372](#),
[1373](#), [1482](#), [1482](#), [1755](#), [1755](#), [1774](#)
`\__draw_backend_transform_-`
aux:nnnn [1482](#), [1496](#), [1501](#)
`\__draw_backend_transform_-`
decompose:nnnnN . [1495](#), [1530](#), [1531](#)
`\__draw_backend_transform_-`
decompose_auxi:nnnnN
..... [1530](#), [1535](#), [1543](#)
`\__draw_backend_transform_-`
decompose_auxii:nnnnN
..... [1530](#), [1547](#), [1555](#)
`\__draw_backend_transform_-`
decompose_auxiii:nnnnN
..... [1530](#), [1559](#), [1567](#)
`\g_draw_draw_clip_bool` .. [1238](#), [1647](#)
`\g_draw_draw_eor_bool`
... [1233](#), [1250](#), [1266](#), [1273](#), [1284](#),
[1295](#), [1311](#), [1422](#), [1436](#), [1441](#), [1446](#)
`\g_draw_draw_path_int` [1647](#)

E

`\errmessage` [38](#)
`\evensidemargin` [2974](#)
exp commands:
`\exp_args:Ne` [605](#),
[659](#), [843](#), [1856](#), [1914](#), [1916](#), [1940](#),
[1942](#), [2331](#), [2346](#), [2397](#), [2771](#), [2970](#)
`\exp_args:Nf` [1323](#), [1455](#), [2892](#)
`\exp_args:Nne` [2738](#)
`\exp_args:NNf` [247](#), [295](#), [352](#)
`\exp_args:Nno` [3372](#)
`\exp_args:No` [3378](#)
`\exp_not:N` [569](#),
[575](#), [576](#), [577](#), [605](#), [607](#), [608](#), [611](#),
[612](#), [617](#), [2590](#), [2592](#), [2595](#), [2601](#),
[2603](#), [2606](#), [2643](#), [2644](#), [2650](#), [2651](#),
[2670](#), [2675](#), [3118](#), [3120](#), [3123](#), [3153](#),
[3155](#), [3158](#), [3216](#), [3217](#), [3218](#), [3223](#)
`\exp_not:n` [48](#), [96](#), [116](#), [154](#),
[932](#), [2322](#), [2327](#), [2391](#), [2560](#), [2561](#),
[2575](#), [2576](#), [2716](#), [2721](#), [2732](#), [2751](#)
`\ExplBackendFileDate` [1](#)

F

file commands:
`\file_compare_timestamp:nNnTF` . [1928](#)
`\file_parse_full_name:nNNN` [1910](#), [1938](#)
`\fmtversion` [51](#)
fp commands:
`\fp_compare:nNnTF`
. [254](#), [301](#), [307](#), [359](#), [1506](#), [1519](#), [1569](#)
`\fp_eval:n`
. [247](#), [256](#), [269](#), [270](#), [295](#), [312](#), [327](#),
[329](#), [352](#), [361](#), [372](#), [373](#), [437](#), [452](#),
[453](#), [1068](#), [1081](#), [1082](#), [1083](#), [1508](#),
[1513](#), [1514](#), [1521](#), [1536](#), [1537](#), [1538](#),
[1539](#), [1548](#), [1549](#), [1550](#), [1551](#), [1560](#),
[1561](#), [1562](#), [1563](#), [2388](#), [2488](#), [2765](#)
`\fp_new:N` [320](#), [321](#)
`\fp_set:Nn` [300](#), [303](#)
`\fp_use:N` [306](#), [310](#), [315](#)
`\fp_zero:N` [302](#)
`\c_zero_fp` [254](#), [301](#), [307](#), [359](#), [1506](#), [1519](#)

G

graphics commands:
`\l_graphics_search_ext_seq`
..... [1794](#), [1812](#), [1960](#), [2144](#)
graphics internal commands:
`\l__graphics_attr_tl` [1814](#),
[1821](#), [1839](#), [1851](#), [1858](#), [1860](#), [1898](#)
`\__graphics_backend_dequote:w` ...
..... [1816](#), [1857](#), [1894](#)
`\l__graphics_backend_dir_str` . [1903](#)
`\l__graphics_backend_ext_str` . [1903](#)
`\__graphics_backend_get_pagecount:n`
..... [1809](#), [1809](#), [1952](#), [1952](#),
[2071](#), [2072](#), [2133](#), [2133](#), [2278](#), [2278](#)
`\__graphics_backend_getbb_auxi:n`
..... [1816](#), [1830](#), [1847](#), [1849](#)
`\__graphics_backend_getbb_-`
auxi:nN [2076](#), [2080](#), [2089](#), [2091](#)
`\__graphics_backend_getbb_-`
auxii:n [1816](#), [1852](#), [1854](#)
`\__graphics_backend_getbb_-`
auxii:nnN .. [2076](#), [2094](#), [2097](#), [2099](#)
`\__graphics_backend_getbb_-`
auxiii:n [1816](#), [1856](#), [1862](#)
`\__graphics_backend_getbb_-`
auxiii:nNnn . [2076](#), [2095](#), [2098](#), [2100](#)
`\__graphics_backend_getbb_-`
auxiv:nnNnn . [2076](#), [2103](#), [2107](#), [2119](#)
`\__graphics_backend_getbb_-`
auxv:nNnn .. [2076](#), [2104](#), [2111](#), [2120](#)
`\__graphics_backend_getbb_-`
auxvi:nNnn [2123](#), [2125](#)

_graphics_backend_getbb_bmp:n .	_graphics_backend_include_-
..... 1962, 1973, 2076, 2084	dequote:w 2257, 2268, 2276
_graphics_backend_getbb_eps:n .	_graphics_backend_include_-
..... 1795, 1795, 1903,	eps:n 1797,
1908, 1925, 1962, 1962, 2225, 2225	1797, 1808, 1903, 1936, 1950,
_graphics_backend_getbb_eps:nm	1983, 1983, 1994, 2241, 2241, 2243
..... 1903	_graphics_backend_include_-
_graphics_backend_getbb_eps:nn	jpeg:n . 1895, 1900, 2000, 2257, 2274
..... 1914, 1926	_graphics_backend_include_-
_graphics_backend_getbb_jpeg:n	jpg:n 1895,
..... 1816, 1832,	1895, 1900, 1901, 1902, 1983,
1962, 1971, 2076, 2082, 2227, 2233	1995, 2000, 2001, 2002, 2257, 2275
_graphics_backend_getbb_jpg:n .	_graphics_backend_include_-
1816, 1816, 1832, 1833, 1962, 1965,	jpseg:n 1983
1971, 1972, 1973, 2076, 2076, 2082,	_graphics_backend_include_-
2083, 2084, 2227, 2227, 2233, 2234	pdf:n 1895,
_graphics_backend_getbb_-	1901, 1940, 1983, 2003, 2241, 2244
pagebox:w 2076, 2115, 2132	_graphics_backend_include_-
_graphics_backend_getbb_pdf:n .	png:n 1895,
..... 1816, 1834, 1934,	.. 1895, 1902, 1983, 2002, 2257, 2273
1962, 1974, 2076, 2085, 2235, 2235	_graphics_backend_include_ps:n
_graphics_backend_getbb_png:n .	 1797, 1808,
..... 1816, 1833,	1903, 1950, 1983, 1994, 2241, 2243
1962, 1972, 2076, 2083, 2227, 2234	_graphics_backend_include_-
_graphics_backend_getbb_ps:n .	svg:n . 2257, 2257, 2273, 2274, 2275
..... 1795, 1796,	\l_graphics_backend_name_str . 1903
1903, 1925, 1962, 1963, 2225, 2226	_graphics_bb_restore:nTF 1851, 2122, 2148
_graphics_backend_getbb_svg:n .	_graphics_bb_save:n 1860, 2130, 2175
..... 2146, 2146	\l_graphics_decodearray_str ...
_graphics_backend_getbb_svg_-	 1823, 1824,
auxi:nNn ... 2146, 2162, 2167, 2180	1836, 1869, 1875, 1876, 1976, 2016,
_graphics_backend_getbb_svg_-	2017, 2057, 2061, 2062, 2087, 2237
auxii:w ... 2146, 2184, 2206, 2211	_graphics_extract_bb:n 1969, 1978, 2231, 2239
_graphics_backend_getbb_svg_-	\l_graphics_final_name_str .. 1933
auxiii:Nw 2146, 2194, 2212	_graphics_get_pagecount:n ...
_graphics_backend_getbb_svg_-	 1809, 2072, 2278
auxiv:Nw 2146, 2197, 2214	\l_graphics_interpolate_bool ...
_graphics_backend_getbb_svg_-	 1825, 1838, 1867, 1879,
auxv:Nw 2146, 2198, 2216	1977, 2018, 2055, 2065, 2088, 2238
_graphics_backend_getbb_svg_-	\l_graphics_llx_dim 1802, 1988, 2047, 2154, 2251
auxvi:Nn 2146, 2213, 2215, 2217, 2218	\l_graphics_lly_dim 1803, 1989, 2048, 2155, 2252
_graphics_backend_getbb_svg_-	\l_graphics_page_int 1818, 1842,
auxvii:w 2146, 2220, 2224	1843, 1884, 1885, 1967, 2014, 2015,
_graphics_backend_include:nn ..	2041, 2042, 2078, 2093, 2094, 2229
..... 2241, 2242, 2245, 2246	\l_graphics_pagebox_tl ... 1819,
_graphics_backend_include_-	1841, 1886, 1887, 1968, 2012, 2013,
auxi:n 1983, 1998, 2006, 2008	2043, 2045, 2079, 2102, 2103, 2230
_graphics_backend_include_-	\l_graphics_pdf_str 1827, 1828, 1844, 1845, 1870, 1881
auxii:nn ... 1983, 2010, 2023, 2032	
_graphics_backend_include_-	
auxiii:nn 1983, 2030, 2033	
_graphics_backend_include_-	
bmp:n 1983, 2001	

<code>__kernel_backend_postscript:n</code>	63 , 63 , 65 , 517 , 1031 , 1033 , 1035 , 1039 , 2285 , 2336 , 2351 , 2371 , 2405 , 2412 , 2898 , 2904 , 2909 , 2945 , 2977 , 2984 , 2988 , 3002 , 3030 , 3073 , 3080 , 3087 , 3094 , 3290
<code>__kernel_backend_scope:n</code>	. . . 184 , 213 , 218 , 412 , 417 , 1060 , 1088 , 1599 , 1644 , 1646 , 1666 , 1706 , 1728 , 1740 , 1742 , 1744 , 1746 , 1748 , 1750 , 1752 , 1754 , 1757 , 1765 , 3423
<code>__kernel_backend_scope_begin:</code>	82 , 82 , 128 , 128 , 173 , 173 , 184 , 184 , 226 , 250 , 265 , 281 , 298 , 324 , 338 , 355 , 368 , 1391 , 1576 , 1594 , 1598 , 1773
<code>__kernel_backend_scope_begin:n</code>	 184 , 203 , 212 , 404 , 432 , 445
<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 , 1392 , 1588 , 1595 , 1601 , 1787
<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 , 1653 , 1656 , 1669 , 1698 , 1701 , 1709
<code>__kernel_color_backend_stack_-init:Nnn</code>	470 , 470 , 3333
<code>__kernel_color_backend_stack_-pop:n</code>	484 , 494 , 542 , 3365
<code>__kernel_color_backend_stack_-push:nn</code>	. 484 , 484 , 539 , 995 , 1007 , 3354 , 3399
<code>__kernel_dependency_version_-check:Nn</code>	1
<code>__kernel_dependency_version_-check:nn</code>	27 , 29
<code>__kernel_file_name_quote:n</code>	 1916 , 1942
L	
lua commands:	
<code>\lua_load_module:n</code>	1175
M	
<code>\MessageBreak</code>	40
mode commands:	
<code>\mode_if_horizontal:TF</code>	3051 , 3058
<code>\mode_if_math:TF</code>	2949
msg commands:	
<code>\msg_error:nnn</code>	2152
O	
<code>\oddsidemargin</code>	2973
opacity internal commands:	
<code>__opacity_backend:nn</code>	 3416 , 3417 , 3419 , 3421 , 3422
<code>__opacity_backend:nnn</code> 3269 , 3271 , 3272 , 3276 , 3283 , 3288 , 3314 , 3321	
<code>__opacity_backend_fill:n</code>	. . 3269 , 3274 , 3370 , 3370 , 3416 , 3418
<code>__opacity_backend_fill_stroke:nn</code>	. . 3370 , 3372 , 3378 , 3382 , 3404 , 3409
<code>\l__opacity_backend_fill_tl</code>	 3339 , 3345 , 3379 , 3387
<code>__opacity_backend_reset:</code>	 3269 , 3307 , 3343 , 3358 , 3368 , 3369 , 3404 , 3410 , 3411 , 3412 , 3416 , 3424 , 3425 , 3426
<code>__opacity_backend_reset_fill:</code>	 3269 , 3309 , 3312 , 3343 , 3368 , 3404 , 3411 , 3416 , 3425
<code>__opacity_backend_reset_stroke:</code>	 3269 , 3310 , 3319 , 3343 , 3369 , 3404 , 3412 , 3416 , 3426
<code>__opacity_backend_select:n</code>	 3269 , 3269 , 3343 , 3343 , 3385 , 3404 , 3408 , 3416 , 3416
<code>\c__opacity_backend_stack_int</code>	 3328 , 3354 , 3365 , 3399
<code>__opacity_backend_stroke:n</code>	. . 3269 , 3281 , 3370 , 3376 , 3416 , 3420
<code>\l__opacity_backend_stroke_tl</code>	 3339 , 3346 , 3374 , 3388
P	
pdf commands:	
<code>\pdf_object_if_exist:nTF</code> 860 , 926 , 944	
<code>\pdf_object_new:n</code>	 851 , 862 , 906 , 928 , 946
<code>\pdf_object_ref:n</code>	 807 , 875 , 939 , 954 , 972 , 977
<code>\pdf_object_ref_last:</code>	 828 , 853 , 856 , 912
<code>\pdf_object_unnamed_write:nn</code>	 835 , 882 , 938 , 953
<code>\pdf_object_write:nnn</code>	 852 , 863 , 907 , 929 , 947
pdf internal commands:	
<code>__pdf_backend:n</code>	 2695 , 2695 , 2697 , 2699 , 2701 , 2715 , 2720 , 2729 , 2749 , 2781 , 2782 , 2792
<code>__pdf_backend_annotation:nnnn</code> 3253	
<code>__pdf_backend_annotation_last:</code> 3254	
<code>__pdf_backend_bdc:nn</code> 2462 , 2462 , 2689 , 2689 , 2826 , 2826 , 2851 , 2851	

_pdf_backend_catalog_gput:nn . .	_pdf_backend_object_write_-
. 2287 , 2287 ,	dict:nn 2295 , 2324 , 2707 , 2718
2505 , 2505 , 2698 , 2698 , 2834 , 2834	_pdf_backend_object_write_-
_pdf_backend_compress_objects:n	fstream:nn . . 2295 , 2329 , 2707 , 2723
. 2428 , 2440 ,	_pdf_backend_object_write_-
2610 , 2621 , 2807 , 2809 , 2845 , 2846	fstream:nnn 2332 , 2334
_pdf_backend_compresslevel:n . .	_pdf_backend_object_write_-
. 2428 , 2428 ,	stream:nn . . 2295 , 2344 , 2707 , 2725
2610 , 2610 , 2807 , 2807 , 2845 , 2845	_pdf_backend_object_write_-
_pdf_backend_destination:nn . . .	stream:nnn 2295 , 2347 , 2349
. 2369 , 2369 ,	_pdf_backend_object_write_-
2468 , 2468 , 2747 , 2747 , 2832 , 2832	stream:nnnn . 2707 , 2724 , 2726 , 2727
_pdf_backend_destination:nnnn .	_pdf_backend_pageobject_ref:n .
. 2369 , 2395 ,	 2367 , 2367 ,
2468 , 2491 , 2747 , 2769 , 2832 , 2833	2599 , 2599 , 2745 , 2745 , 2836 , 2844
_pdf_backend_destination_-	_pdf_backend_pagesize_gset:nn .
aux:nnnn	. . 2855 , 2855 , 2874 , 2874 , 2881 , 2881
. . 2369 , 2397 , 2400 , 2747 , 2771 , 2774	_pdf_backend_pdfmark:n
_pdf_backend_emc: . . 2462 , 2464 ,	2284 , 2284 , 2286 , 2288 , 2290 , 2304 ,
2689 , 2691 , 2826 , 2828 , 2851 , 2852	2321 , 2326 , 2372 , 2416 , 2463 , 2465
_pdf_backend_info_gput:nn	_pdf_backend_version_major:
. 2287 , 2289 ,	. . . 2454 , 2460 , 2460 , 2666 , 2666 ,
2505 , 2515 , 2698 , 2700 , 2834 , 2835	2816 , 2817 , 2824 , 2824 , 2849 , 2849
_pdf_backend_objcompresslevel:n	_pdf_backend_version_major_-
. 2610 , 2624 , 2625 , 2627	gset:n 2452 , 2452 ,
_pdf_backend_object_id:n	2638 , 2638 , 2814 , 2814 , 2847 , 2847
. 2291 , 2294 ,	_pdf_backend_version_minor:
2526 , 2544 , 2703 , 2706 , 2836 , 2838	. . . 2458 , 2460 , 2461 , 2666 , 2679 ,
_pdf_backend_object_int	2821 , 2822 , 2824 , 2825 , 2849 , 2850
. 2292 , 2359 , 2361 ,	_pdf_backend_version_minor_-
2366 , 2535 , 2704 , 2737 , 2739 , 2744	gset:n 2452 , 2456 ,
_pdf_backend_object_last:	2638 , 2655 , 2814 , 2819 , 2847 , 2848
. 2365 , 2365 ,	_pdf_exp_not_i:nn
2588 , 2588 , 2743 , 2743 , 2836 , 2843	 2545 , 2564 , 2569 , 2575
_pdf_backend_object_new:	_pdf_exp_not_ii:nn
. 2291 , 2291 ,	 2545 , 2565 , 2570 , 2576
2526 , 2526 , 2703 , 2703 , 2836 , 2836	pdf.baselineskip 3804
_pdf_backend_object_now:nn	pdf.bordertracking 3562
. 2357 , 2357 , 2364 , 2577 , 2577 , 2587 ,	pdf.bordertracking.begin 3562
2735 , 2735 , 2742 , 2836 , 2841 , 2842	pdf.bordertracking.continue 3562
_pdf_backend_object_prop	pdf.bordertracking.end 3562
. 2525 , 2702	pdf.bordertracking.endpage 3562
_pdf_backend_object_ref:n	pdf.breaklink 3700
. 2291 , 2293 , 2294 , 2298 , 2526 , 2543 ,	pdf.breaklink.write 3700
2703 , 2705 , 2706 , 2710 , 2836 , 2837	pdf.brokenlink.dict 3562
_pdf_backend_object_write:nn . . .	pdf.brokenlink.rect 3562
. 2545 , 2554 , 2556 , 2585 , 2836	pdf.brokenlink.skip 3562
_pdf_backend_object_write:nnn . .	pdf.count 3700
. 2295 , 2295 , 2301 , 2545 , 2545 , 2574 ,	pdf.currentrect 3700
2707 , 2707 , 2712 , 2836 , 2839 , 2840	pdf.cvs 3484
_pdf_backend_object_write_-	pdf.dest.anchor 3527
array:nn . . . 2295 , 2319 , 2707 , 2713	pdf.dest.point 3527
_pdf_backend_object_write_-	pdf.dest.x 3527
aux:nnn 2295 , 2297 , 2302 , 2360	pdf.dest.y 3527

pdf.dest2device	3527	_pdfannot_backend_last:	2919, 2919, 3116,
pdf.dev.x	3527		3116, 3204, 3204, 3242, 3242, 3254
pdf.dev.y	3527	_pdfannot_backend_link:nw	2930
pdf.dvi.pt	3484	_pdfannot_backend_link_aux:nw	2930
pdf.globaldict	3481	_pdfannot_backend_link_begin:n	3207, 3209, 3213, 3214
pdf.leftboundary	3562	_pdfannot_backend_link_-	
pdf.linkdp.pad	3488	begin:nnnw	3127, 3128, 3130, 3131, 3243, 3245
pdf.linkht.pad	3488	_pdfannot_backend_link_-	
pdf.linkmargin	3488	begin:nw	2932, 2936, 2937
pdf.llx	3491	_pdfannot_backend_link_begin_-	
pdf.lly	3491	aux:nw	2940, 2942
pdf.originx	3562	_pdfannot_backend_link_begin_-	
pdf.originy	3562	goto:nnw	2930, 2930,
pdf.outerbox	3804	3127, 3127, 3207, 3207, 3243, 3243	
pdf.pdfmark	3804	_pdfannot_backend_link_begin_-	
pdf.pdfmark.dict	3804	user:nnw	2930, 2935,
pdf.pdfmark.good	3804	3127, 3129, 3207, 3212, 3243, 3244	
pdf.pt.dvi	3484	\g_pdfannot_backend_link_bool	2925, 2939, 2944, 2959, 2997
pdf.rect	3491	\g_pdfannot_backend_link_dict_-	
pdf.rect.ht	3484	t1	2922, 2947, 2992
pdf.rightboundary	3562	_pdfannot_backend_link_end:	2930, 2957,
pdf.save.linkll	3491	3127, 3142, 3207, 3231, 3243, 3246	
pdf.save.linkur	3491	_pdfannot_backend_link_end_-	
pdf.save.ll	3491	aux:	2930, 2960, 2962
pdf.save.ur	3491	\g_pdfannot_backend_link_int	2921, 2987,
pdf.tmpa	3527	2991, 3091, 3206, 3217, 3223, 3234	
pdf.tmpb	3527	_pdfannot_backend_link_last:	3090, 3090,
pdf.tmpc	3527	3151, 3151, 3233, 3233, 3247, 3247	
pdf.tmpd	3527	_pdfannot_backend_link_-	
pdf.urx	3491	margin:n	3092, 3092,
pdf.ury	3491	3162, 3162, 3235, 3235, 3248, 3248	
pdfannot internal commands:		\g_pdfannot_backend_link_math_-	
_pdfannot_backend:n	3188, 3188,	bool	2924, 2950, 2951, 2954, 2964
	3190, 3195, 3219, 3232, 3237, 3238	_pdfannot_backend_link_minima:	2930, 2968, 2999
\l_pdfannot_backend_breaklink_-		_pdfannot_backend_link_off:	
pdfmark_t1	2926, 2994, 3085	3099, 3100,	
_pdfannot_backend_breaklink_-		3172, 3179, 3237, 3238, 3249, 3250	
postscript:n	2928, 2928, 2978, 2980, 3086	_pdfannot_backend_link_on:	
_pdfannot_backend_breaklink_-		3099, 3099,	
usebox:N	2929, 2929, 2979, 3088	3172, 3172, 3237, 3237, 3249, 3249	
\l_pdfannot_backend_content_box		_pdfannot_backend_link_-	
	2887,	outerbox:n	2930, 2970, 3028
	2952, 2976, 2979, 2981, 3010, 3021	\g_pdfannot_backend_link_sf_int	2923, 3049, 3060, 3061
_pdfannot_backend_generic:nnnn		_pdfannot_backend_link_sf_-	
	2890, 2890, 3103,	restore:	2930, 2953, 2996, 3056
	3103, 3192, 3192, 3241, 3241, 3253		
_pdfannot_backend_generic_-			
aux:nnnn	2890, 2892, 2895		
\g_pdfannot_backend_int			
	2889, 2908, 2912, 2920, 2986, 2987,		
	3191, 3194, 3197, 3205, 3216, 3218		

<code>_pdfannot_backend_link_sf_-</code> <code> save:</code> 2930, 2948, 2966, 3047 <code>\l_pdfannot_backend_model_box . .</code> <code> </code> 2888, 2969, 3001, 3009, 3020, 3035, 3037 pdfmanagement commands: <code>\pdfmanagement_add:nnn</code> <code> </code> 825, 3336, 3347, 3389, 3392 <code>\pdfmanagement_if_active_p:</code> <code> </code> 820, 821, 3329, 3330, 3405, 3406 peek commands: <code>\peek_meaning:NTF</code> 2193, 2196 <code>\peek_remove_spaces:n</code> 2191 prg commands: <code>\prg_replicate:nn</code> <code> </code> 195, 653, 674, 684, 888 prop commands: <code>\prop_gput:Nnn</code> 611, 855 <code>\prop_if_in:NnTF</code> 585 <code>\prop_item:Nn</code> 588 <code>\prop_new:N</code> 566, 2525, 2702 <code>\ProvidesExplFile</code> 2	<code>\str_convert_pdfname:n</code> 612, 632, 844 <code>\str_if_empty:NTF</code> 1827, 1844 <code>\str_if_empty_p:N</code> 1870 <code>\str_if_eq:nnTF</code> 791, 1488, 3384 <code>\str_new:N</code> 1905, 1906, 1907 <code>\str_tail:N</code> 1919, 1945 sys commands: <code>\sys_if_shell:TF</code> 1903 <code>\sys_shell_now:n</code> 1930
Q quark commands: <code>\quark_if_recursion_tail_stop:n</code> 584 <code>\q_recursion_stop</code> 577 <code>\q_recursion_tail</code> 576	T TeX and L^AT_EX 2_ε commands: <code>\@ifl@t@r</code> 49, 51 <code>\current@color</code> 26 <code>\special</code> 2 tex commands: <code>\tex_afterassignment:D</code> 2220 <code>\tex_baselineskip:D</code> 3041 <code>\tex_endinput:D</code> 44 <code>\tex_global:D</code> <code> </code> 2612, 2629, 2643, 2650, 2657 <code>\tex_immediate:D</code> <code> </code> 1864, 2548, 2551, 2580, 2583 <code>\tex_luatexversion:D</code> 2641, 2669 <code>\tex_pageheight:D</code> 2877 <code>\tex_pagewidth:D</code> 2876 <code>\tex_pdfannot:D</code> 3109 <code>\tex_pdfcatalog:D</code> 2511 <code>\tex_pdfcolorstack:D</code> 490, 500 <code>\tex_pdfcolorstackinit:D</code> 478 <code>\tex_pdfcompresslevel:D</code> 2617 <code>\tex_pdfdest:D</code> 2474, 2497 <code>\tex_pdfendlink:D</code> 3148 <code>\tex_pdfextension:D</code> 91, 101, 111, 121, 131, 140, 149, 159, 487, 497, 2471, 2494, 2508, 2518, 2529, 2548, 2580, 3106, 3134, 3145, 3174, 3181 <code>\tex_pdffeedback:D</code> <code> </code> 475, 2537, 2592, 2603, 3120, 3155 <code>\tex_pdfinfo:D</code> 2521 <code>\tex_pdflastannot:D</code> 3123 <code>\tex_pdflastlink:D</code> 3158 <code>\tex_pdflastobj:D</code> 2540, 2595 <code>\tex_pdflastximage:D</code> 1859, 1890 <code>\tex_pdflastximagepages:D</code> 1956 <code>\tex_pdflinkmargin:D</code> 3168 <code>\tex_pdfliteral:D</code> 94, 104, 114, 124 <code>\tex_pdfmajorversion:D</code> <code> </code> 2648, 2650, 2674, 2675 <code>\tex_pdfminorversion:D</code> 2662, 2686 <code>\tex_pdfobj:D</code> 2532, 2551, 2583 <code>\tex_pdfobjcompresslevel:D</code> 2634 <code>\tex_pdfpageref:D</code> 2606 <code>\tex_pdfrefximage:D</code> 1890, 1897
S scan commands: <code>\scan_stop:</code> 131, 140, 502, 2221, 2224, 2489, 2503, 2619, 2636, 2644, 2651, 2664, 3145, 3170 scan internal commands: <code>\s__color_stop</code> <code> </code> 664, 665, 669, 673, 686, 689, 693, 697, 711, 889, 918, 922, 1074, 1076 <code>\s__graphics_stop</code> <code> </code> 1857, 1894, 2186, 2201, 2208, 2212, 2214, 2216, 2268, 2276 separation 3478 seq commands: <code>\seq_set_from_clist:Nn</code> <code> </code> 1794, 1812, 1960, 2144 shipout commands: <code>\l_shipout_box</code> 3069, 3071, 3079 skip commands: <code>\skip_horizontal:n</code> 244, 292, 349 str commands: <code>\c_hash_str</code> 415, 1662, 1669, 1709 <code>\c_percent_str</code> 1094, 1095, 1096 <code>\str_case:nn</code> 894, 2308, 2558 <code>\str_case:nnTF</code> 2376, 2477, 2754	

