

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}{2025-10-09}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2025-10-09}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2025-10-09}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2025-10-09}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2025-10-09}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2025-10-09}{ }
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

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>

```

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.

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

```

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

```

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

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

`_box_backend_rotate_aux:Nn`

```

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

```

(End of definition for `_box_backend_rotate:Nn` and `_box_backend_rotate_aux:Nn`.)

`__box_backend_scale:Nnn` The **dvips** backend once again has a dedicated operation we can use here.

```

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

```

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

```

277 </dvips>

```

2.2 LuaTeX and pdfTeX backends

```

278 <!*luatex | pdftex>

```

`__box_backend_clip:N` The general method is to save the current location, define a clipping path equivalent to the bounding box, then insert the content at the current position and in a zero width box. The “real” width is then made up using a horizontal skip before tidying up. There are other approaches that can be taken (for example using XForm objects), but the logic here shares as much code as possible and uses the same conversions (and so same rounding errors) in all cases.

```

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

```

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

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

```

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

```

```

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

```

(End of definition for __box_backend_rotate:Nn and others.)

__box_backend_scale:Nnn The same idea as for rotation but without the complexity of signs and cosines.

```

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

```

(End of definition for __box_backend_scale:Nnn.)

334 </luatex | pdftex>

2.3 dvipdfmx/X_YTeX backend

335 <*dvipdfmx | xetex>

__box_backend_clip:N The code here is identical to that for Lua_YTeX/pdf_YTeX: 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 dvipdfmx/X_YTeX can be implemented using either PDF or backend-specific code. The former approach however is not “aware” of the content of boxes: this means that any embedded links would not be adjusted by the rotation. As such, the backend-native approach is preferred: the code therefore is similar (though not identical) to the dvips version (notice the rotation angle here is positive). As for dvips, zero rotation is written as 0 not -0.

```

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

```

(End of definition for __box_backend_rotate:Nn and __box_backend_rotate_aux:Nn.)

__box_backend_scale:Nnn

Much the same idea for scaling: use the higher-level backend operation to allow for box content.

```

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

```

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

378 `</dviptfm | xetex>`

2.4 dvisvgm backend

379 `<*dvisvgm>`

`__box_backend_clip:N`
`\g__kernel_clip_path_int`

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

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

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

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

```

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

```

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

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

```

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

```

(End of definition for __box_backend_rotate:Nn.)

__box_backend_scale:Nnn In contrast to rotation, we have to account for the current position in this case. That is done using a couple of translations in addition to the scaling (which is therefore done backward with a flip).

```

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

```

```

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

```

(End of definition for __box_backend_scale:Nnn.)

```

462 </dvisvgm>
463 </package>

```

3 I3backend-color implementation

```

464 <*package>
465 <@@=color>

```

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

3.1 The color stack

For PDF-based engines, we have a color stack available inside the specials. This is used for concepts beyond color itself: it is needed to manage the graphics state generally. Although dvipdfmx/X_YL_AT_EX have multiple color stacks in recent releases, the way these interact with the original single stack and with other graphic state operations means that currently it is not feasible to use the multiple stacks.

3.1.1 Common code

```

466 <*luatex | pdftex>

```

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

```

467 \int_new:N \l__color_backend_stack_int

```

(End of definition for \l__color_backend_stack_int.)

```

468 </luatex | pdftex>

```

3.1.2 Lua_YL_AT_EX and pdf_YL_AT_EX

```

469 <*luatex | pdftex>

```

__kernel_color_backend_stack_init:Nnn

```

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

```

```

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

```

(End of definition for _kernel_color_backend_stack_init:Nnn.)

```

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

```

(End of definition for _kernel_color_backend_stack_push:nn and _kernel_color_backend_stack_pop:n.)

```

504 </luatex | pdfTeX>

```

3.2 General color

3.2.1 dvips-style

```

505 <*dvips | dvisvgm>

```

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

```

506 \cs_new_protected:Npn \_color_backend_select_cmyk:n #1
507 { \_color_backend_select:n { cmyk ~ #1 } }
508 \cs_new_protected:Npn \_color_backend_select_gray:n #1
509 { \_color_backend_select:n { gray ~ #1 } }
510 \cs_new_protected:Npn \_color_backend_select_named:n #1
511 { \_color_backend_select:n { ~ #1 } }
512 \cs_new_protected:Npn \_color_backend_select_rgb:n #1
513 { \_color_backend_select:n { rgb ~ #1 } }
514 \cs_new_protected:Npn \_color_backend_select:n #1
515 {
516   \_kernel_backend_literal:n { color~push~ #1 }

```

```

517 <dvips>
518   \_kernel_backend_postscript:n { /color.sc ~ { } ~ def }
519 </dvips>
520 }
521 \cs_new_protected:Npn \_color_backend_reset:
522 { \_kernel_backend_literal:n { color~pop } }

```

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

```

523 </dvips | dvisvgm>

```

3.2.2 LuaTeX and pdfTeX

```

524 <*luatex | pdftex>

```

```

\_l_color_backend_fill_tl
\_l_color_backend_stroke_tl

```

```

525 \tl_new:N \l_color_backend_fill_tl
526 \tl_new:N \l_color_backend_stroke_tl
527 \tl_set:Nn \l_color_backend_fill_tl { 0 ~ g }
528 \tl_set:Nn \l_color_backend_stroke_tl { 0 ~ G }

```

(End of definition for \l_color_backend_fill_tl and \l_color_backend_stroke_tl.)

```

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

```

Store the values then pass to the stack.

```

529 \cs_new_protected:Npn \_color_backend_select_cmyk:n #1
530 { \_color_backend_select:nn { #1 ~ k } { #1 ~ K } }
531 \cs_new_protected:Npn \_color_backend_select_gray:n #1
532 { \_color_backend_select:nn { #1 ~ g } { #1 ~ G } }
533 \cs_new_protected:Npn \_color_backend_select_rgb:n #1
534 { \_color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
535 \cs_new_protected:Npn \_color_backend_select:nn #1#2
536 {
537   \tl_set:Nn \l_color_backend_fill_tl {#1}
538   \tl_set:Nn \l_color_backend_stroke_tl {#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:n 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:n
  \__color_backend_select_gray:n
  \__color_backend_select_rgb:n
\__color_backend_reset:
545 \cs_new_protected:Npn \__color_backend_select:n #1
546 { \__kernel_backend_literal:n { pdf : bc ~ [ #1 ] } }
547 \cs_new_eq:NN \__color_backend_select_cmyk:n \__color_backend_select:n
548 \cs_new_eq:NN \__color_backend_select_gray:n \__color_backend_select:n
549 \cs_new_eq:NN \__color_backend_select_rgb:n \__color_backend_select:n
550 \cs_new_protected:Npn \__color_backend_reset:
551 { \__kernel_backend_literal:n { pdf : ec } }

```

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

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

```

\__color_backend_select_named:n
552 \cs_new_protected:Npn \__color_backend_select_named:n #1
553 {
554   \str_if_eq:nnTF {#1} { Black }
555   { \__color_backend_select_gray:n { 0 } }
556   { \msg_error:nnn { color } { unknown-named-color } {#1} }
557 }
558 \msg_new:nnn { color } { unknown-named-color }
559 { Named-color~'#1'~is~not~known. }

```

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

(End of definition for __color_backend_select_named:n.)

560 </dviptdmx | xetex>

3.3 Separations

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

561 <*dviptdmx | luatex | pdftex | xetex | dvips>

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

```

\g__color_backend_colorant_prop
562 \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
563 \cs_new:Npe \__color_backend_devicen_colorants:n #1
564 {
565   \exp_not:N \tl_if_blank:nF {#1}
566   {
567     \c_space_tl
568     << ~
569     /Colorants ~
570     << ~
571     \exp_not:N \__color_backend_devicen_colorants:w #1 ~
572     \exp_not:N \q_recursion_tail \c_space_tl
573     \exp_not:N \q_recursion_stop
574     >> ~
575     >>
576   }
577 }
578 \cs_new:Npn \__color_backend_devicen_colorants:w #1 ~

```

```

579 {
580   \quark_if_recursion_tail_stop:n {#1}
581   \prop_if_in:NnT \g__color_backend_colorant_prop {#1}
582   {
583     #1 ~
584     \prop_item:Nn \g__color_backend_colorant_prop {#1} ~
585   }
586   \__color_backend_devicen_colorants:w
587 }

```

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

```

588 </dviptfm | luatex | pdftex | xetex | dvips>

```

```

589 <*dvips>

```

```

\__color_backend_select_separation:nn
\__color_backend_select_devicen:nn

```

```

590 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
591 { \__color_backend_select:n { separation ~ #1 ~ #2 } }
592 \cs_new_eq:NN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn

```

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

```

\__color_backend_select_iccbased:nn

```

No support.

```

593 \cs_new_protected:Npn \__color_backend_select_iccbased:nn #1#2 { }

```

(End of definition for __color_backend_select_iccbased:nn.)

```

\__color_backend_separation_init:nnnnn
\__color_backend_separation_init:neemn
\__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.

```

594 \cs_new_protected:Npe \__color_backend_separation_init:nnnnn #1#2#3#4#5
595 {
596   \bool_if:NT \g__kernel_backend_header_bool
597   {
598     \exp_not:N \exp_args:Ne \__kernel_backend_first_shipout:n
599     {
600       \exp_not:N \__color_backend_separation_init_aux:nnnnnn
601       { \exp_not:N \int_use:N \g__color_model_int }
602       {#1} {#2} {#3} {#4} {#5}
603     }
604     \prop_gput:Nee \exp_not:N \g__color_backend_colorant_prop
605     { / \exp_not:N \str_convert_pdftname:n {#1} }
606     {
607       << ~
608       /setcolorspace ~ {} ~
609       >> ~ begin ~
610       color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
611       end
612     }
613   }
614 }
615 \cs_generate_variant:Nn \__color_backend_separation_init:nnnnn { nee }
616 \cs_new_protected:Npn \__color_backend_separation_init_aux:nnnnnn #1#2#3#4#5#6
617 {

```

```

618   \__kernel_backend_literal:e
619   {
620     !
621     TeXDict ~ begin ~
622     /color #1
623     {
624       [ ~
625       /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
626       [ ~ #3 ~ ] ~
627       {
628         \cs_if_exist_use:cF { __color_backend_separation_init_ #3 :nnn }
629         { \__color_backend_separation_init:nnn }
630         {#4} {#5} {#6}
631       }
632       ] ~ setcolorspace
633     } ~ def ~
634   end
635 }
636 }
637 \cs_new:cpn { __color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
638 { \__color_backend_separation_init_Device:Nn 4 {#3} }
639 \cs_new:cpn { __color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
640 { \__color_backend_separation_init_Device:Nn 1 {#3} }
641 \cs_new:cpn { __color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
642 { \__color_backend_separation_init_Device:Nn 2 {#3} }
643 \cs_new:Npn \__color_backend_separation_init_Device:Nn #1#2
644 {
645   #2 ~
646   \prg_replicate:nn {#1}
647   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
648   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
649 }

```

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.

```

650 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3
651 {
652   \exp_args:Ne \__color_backend_separation_init:nnnn
653   { \__color_backend_separation_init_count:n {#2} }
654   {#1} {#2} {#3}
655 }
656 \cs_new:Npn \__color_backend_separation_init_count:n #1
657 { \int_eval:n { 0 \__color_backend_separation_init_count:w #1 ~ \s_color_stop } }
658 \cs_new:Npn \__color_backend_separation_init_count:w #1 ~ #2 \s_color_stop
659 {
660   +1
661   \tl_if_blank:nF {#2}
662   { \__color_backend_separation_init_count:w #2 \s_color_stop }
663 }

```

Now we implement the algorithm. In the terms in the PostScript manual, we have $\mathbf{N} = 1$ and $\mathbf{Domain} = [0\ 1]$, with \mathbf{Range} as #2, $\mathbf{C0}$ as #3 and $\mathbf{C1}$ as #4, with the number of output components in #1. So all we have to do is implement $y_i = \mathbf{C0}_i + x(\mathbf{C1}_i - \mathbf{C0}_i)$

with lots of stack manipulation, then check the ranges. That's done by adding everything to the stack first, then using the fact we know all of the offsets. As manipulating the stack is tricky, we start by re-formatting the **C0** and **C1** arrays to be interleaved, and add a 0 to each pair: this is used to keep the stack of constant length while we are doing the first pass of mathematics. We then working through that list, calculating from the last to the first value before tidying up by removing all of the input values. We do that by first copying all of the final *y* values to the end of the stack, then rolling everything so we can pop the now-unneeded material.

```

664 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
665 {
666   \__color_backend_separation_init:w #3 ~ \s__color_stop #4 ~ \s__color_stop
667   \prg_replicate:nn {#1}
668   {
669     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
670     \int_eval:n { 3 * #1 } ~ index ~ mul ~
671     2 ~ index ~ add ~
672     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
673   }
674   \int_step_function:nnnN {#1} { -1 } { 1 }
675   \__color_backend_separation_init:n
676   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
677   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
678   \tl_if_blank:nF {#2}
679   { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
680 }
681 \cs_new:Npn \__color_backend_separation_init:w
682 #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
683 {
684   #1 ~ #3 ~ 0 ~
685   \tl_if_blank:nF {#2}
686   { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }
687 }
688 \cs_new:Npn \__color_backend_separation_init:n #1
689 { \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.

```

690 \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s__color_stop
691 {
692   #2 ~ #3 ~
693   2 ~ index ~ 2 ~ index ~ lt ~
694   { ~ pop ~ exch ~ pop ~ } ~
695   { ~
696     2 ~ index ~ 1 ~ index ~ gt ~
697     { ~ exch ~ pop ~ exch ~ pop ~ } ~
698     { ~ pop ~ pop ~ } ~
699     ifelse ~
700   }
701   ifelse ~
702   #1 ~ 1 ~ roll ~
703   \tl_if_blank:nF {#4}
704   { \__color_backend_separation_init:nw {#1} #4 \s__color_stop }

```

705 }

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.

```

706 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
707 {
708   \__color_backend_separation_init:neenn
709   {#2}
710   {
711     /CIEBasedABC ~
712     << ~
713     /RangeABC ~ [ ~ \c__color_model_range_CIELAB_tl \c_space_tl ] ~
714     /DecodeABC ~
715     [ ~
716     { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
717     { ~ 500 ~ div ~ } ~ bind ~
718     { ~ 200 ~ div ~ } ~ bind ~
719     ] ~
720     /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
721     /DecodeLMN ~
722     [ ~
723     { ~
724       dup ~ 6 ~ 29 ~ div ~ ge ~
725       { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
726       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
727       ifelse ~
728       0.9505 ~ mul ~
729     } ~ bind ~
730     { ~
731       dup ~ 6 ~ 29 ~ div ~ ge ~
732       { ~ dup ~ dup ~ mul ~ mul ~ } ~
733       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
734       ifelse ~
735     } ~ bind ~
736     { ~
737       dup ~ 6 ~ 29 ~ div ~ ge ~
738       { ~ dup ~ dup ~ mul ~ mul ~ } ~
739       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
740       ifelse ~
741       1.0890 ~ mul ~
742     } ~ bind
743     ] ~
744     /WhitePoint ~
745     [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
746     >>
747   }
748   { \c__color_model_range_CIELAB_tl }
749   { 100 ~ 0 ~ 0 }
750   {#3}
751 }

```

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

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

```

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

```

```

753 {
754   \__kernel_backend_literal:e
755   {
756     !
757     TeXDict ~ begin ~
758     /color \int_use:N \g__color_model_int
759     {
760       [ ~
761         /DeviceN ~
762         [ ~ #1 ~ ] ~
763         #2 ~
764         { ~ #3 ~ } ~
765         \__color_backend_devicen_colorants:n {#1}
766       ] ~ setcolorspace
767     } ~ def ~
768   end
769 }
770 }

```

(End of definition for __color_backend_devicen_init:nnn.)

__color_backend_iccbased_init:nnn No support at present.

```

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

```

(End of definition for __color_backend_iccbased_init:nnn.)

```

772 </dvips>

```

```

773 <*dvisvgm>

```

__color_backend_select_separation:nn No support at present.

__color_backend_select_devicen:nn

```

774 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2 { }

```

```

775 \cs_new_eq:NN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn

```

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

__color_backend_separation_init:nnnnn

No support at present.

__color_backend_separation_init_CIELAB:nnn

```

776 \cs_new_protected:Npn \__color_backend_separation_init:nnnnn #1#2#3#4#5 { }

```

```

777 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnnnnn #1#2#3 { }

```

(End of definition for __color_backend_separation_init:nnnnn and __color_backend_separation_init_CIELAB:nnn.)

__color_backend_select_iccbased:nn

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

```

778 \cs_new_protected:Npn \__color_backend_select_iccbased:nn #1#2
779 {
780   \__kernel_backend_literal_svg:e
781   {
782     <style>
783       @color-profile ~
784       \str_if_eq:nnTF {#2} { cmyk }
785       { device-cmyk }
786       { --color \int_use:N \g__color_model_int }
787       \c_space_tl
788     {

```

```

789         src:("#1")
790     }
791     </style>
792 }
793 }

```

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

```

794 </dvisvgm>
795 <*dvipdfmx | luatex | pdftex | xetex>

```

```

\_color_backend_select_separation:nn
\_color_backend_select_devicen:nn
\_color_backend_select_iccbased:nn

```

```

796 <*dvipdfmx | xetex>
797 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
798 { \_kernel_backend_literal:e { pdf : bc ~ \pdf_object_ref:n {#1} ~ [ #2 ] } }
799 </dvipdfmx | xetex>
800 <*luatex | pdftex>
801 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
802 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
803 </luatex | pdftex>
804 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
805 \cs_new_eq:NN \_color_backend_select_iccbased:nn \_color_backend_select_separation:nn

```

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

```
\_color_backend_init_resource:n
```

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

```

806 \cs_new_protected:Npn \_color_backend_init_resource:n #1
807 {
808   <*luatex | pdftex>
809   \bool_lazy_and:nnT
810     { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
811     { \pdfmanagement_if_active_p: }
812   {
813     \use:e
814     {
815       \pdfmanagement_add:nnn
816         { Page / Resources / ColorSpace }
817         { #1 }
818         { \pdf_object_ref_last: }
819     }
820   }
821 </luatex | pdftex>
822 }

```

(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.

```

823 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5
824 {
825   \pdf_object_unnamed_write:ne { dict }

```

```

826     {
827         /FunctionType ~ 2
828         /Domain ~ [0 ~ 1]
829         \t1_if_blank:nF {#3} { /Range ~ [#3] }
830         /C0 ~ [#4] ~
831         /C1 ~ [#5] /N ~ 1
832     }
833     \exp_args:Ne \_color_backend_separation_init:nn
834     { \str_convert_pdftname:n {#1} } {#2}
835     \_color_backend_init_resource:n { color \int_use:N \g\_color_model_int }
836 }
837 \cs_new_protected:Npn \_color_backend_separation_init:nn #1#2
838 {
839     \use:e
840     {
841         \pdf_object_new:n { color \int_use:N \g\_color_model_int }
842         \pdf_object_write:nnn { color \int_use:N \g\_color_model_int } { array }
843         { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
844     }
845     \prop_gput:Nne \g\_color_backend_colorant_prop { /#1 }
846     { \pdf_object_ref_last: }
847 }

```

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

```

848 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
849 {
850     \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
851     {
852         \pdf_object_new:n { __color_illuminant_CIELAB_ #1 }
853         \pdf_object_write:nne { __color_illuminant_CIELAB_ #1 } { array }
854         {
855             /Lab ~
856             <<
857             /WhitePoint ~
858             [ \t1_use:c { c\_color_model_whitepoint_CIELAB_ #1 _t1 } ]
859             /Range ~ [ \c\_color_model_range_CIELAB_t1 ]
860             >>
861         }
862     }
863     \_color_backend_separation_init:nnnnn
864     {#2}
865     { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
866     { \c\_color_model_range_CIELAB_t1 }
867     { 100 ~ 0 ~ 0 }
868     {#3}
869 }

```

(End of definition for _color_backend_separation_init:nnnnn, _color_backend_separation_init:nn, and _color_backend_separation_init_CIELAB:nnn.)

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

```

870 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
871 {

```

```

872 \pdf_object_unnamed_write:ne { stream }
873 {
874   {
875     /FunctionType ~ 4 ~
876     /Domain ~
877     [ ~
878       \prg_replicate:nn
879       { 0 \_color_backend_devicen_init:w #1 ~ \s_color_stop }
880       { 0 ~ 1 ~ }
881     ] ~
882     /Range ~
883     [ ~
884       \str_case:nn {#2}
885       {
886         { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
887         { /DeviceGray } { 0 ~ 1 }
888         { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
889       } ~
890     ]
891   }
892   { {#3} }
893 }
894 \use:e
895 {
896   \pdf_object_new:n { color \int_use:N \g_color_model_int }
897   \pdf_object_write:nnn { color \int_use:N \g_color_model_int } { array }
898   {
899     /DeviceN ~
900     [ ~ #1 ~ ] ~
901     #2 ~
902     \pdf_object_ref_last:
903     \_color_backend_devicen_colorants:n {#1}
904   }
905 }
906 \_color_backend_init_resource:n { color \int_use:N \g_color_model_int }
907 }
908 \cs_new:Npn \_color_backend_devicen_init:w #1 ~ #2 \s_color_stop
909 {
910   + 1
911   \tl_if_blank:nF {#2}
912   { \_color_backend_devicen_init:w #2 \s_color_stop }
913 }

```

(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.

```

914 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3
915 {
916   \pdf_object_if_exist:nF { \_color_icc_ #1 }
917   {
918     \pdf_object_new:n { \_color_icc_ #1 }
919     \pdf_object_write:nne { \_color_icc_ #1 } { fstream }
920     {
921       {

```

```

922         /N ~ \exp_not:n { #2 } ~
923         \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
924     }
925     {#1}
926 }
927 }
928 \pdf_object_unnamed_write:ne { array }
929 { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
930 \__color_backend_init_resource:n { color \int_use:N \g__color_model_int }
931 }

```

(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.

```

932 \cs_new_protected:Npn \__color_backend_iccbased_device:nnn #1#2#3
933 {
934     \pdf_object_if_exist:nF { __color_icc_ #1 }
935     {
936         \pdf_object_new:n { __color_icc_ #1 }
937         \pdf_object_write:nnn { __color_icc_ #1 } { fstream }
938         {
939             { /N ~ #3 }
940             {#1}
941         }
942     }
943     \pdf_object_unnamed_write:ne { array }
944     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
945     \__color_backend_init_resource:n { Default #2 }
946 }

```

(End of definition for __color_backend_iccbased_device:nnn.)

```

947 </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.

```

948 <*dvipdfmx | xetex>

```

```

\__color_backend_fill:n
\__color_backend_fill_cmyk:n
\__color_backend_fill_gray:n
\__color_backend_fill_rgb:n
\__color_backend_stroke:n
\__color_backend_stroke_cmyk:n
\__color_backend_stroke_gray:n
\__color_backend_stroke_rgb:n
949 \cs_new_protected:Npn \__color_backend_fill:n #1
950 { \__kernel_backend_literal:n { pdf : bc ~ fill ~ [ #1 ] } }
951 \cs_new_eq:NN \__color_backend_fill_cmyk:n \__color_backend_fill:n
952 \cs_new_eq:NN \__color_backend_fill_gray:n \__color_backend_fill:n
953 \cs_new_eq:NN \__color_backend_fill_rgb:n \__color_backend_fill:n
954 \cs_new_protected:Npn \__color_backend_stroke:n #1
955 { \__kernel_backend_literal:n { pdf : bc ~ stroke ~ [ #1 ] } }
956 \cs_new_eq:NN \__color_backend_stroke_cmyk:n \__color_backend_stroke:n
957 \cs_new_eq:NN \__color_backend_stroke_gray:n \__color_backend_stroke:n
958 \cs_new_eq:NN \__color_backend_stroke_rgb:n \__color_backend_stroke:n

```

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

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
959 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
960 {
961   \_kernel_backend_literal:e
962   { pdf : bc ~ fill ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
963 }
964 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
965 {
966   \_kernel_backend_literal:e
967   { pdf : bc ~ stroke ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
968 }
969 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
970 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

```

\_color_backend_fill_reset:
  \_color_backend_stroke_reset:
971 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
972 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:
(End of definition for \_color_backend_fill_reset: and \_color_backend_stroke_reset:.)
973 </dviPDFmx | xetex>
974 <*luatex | pdftex>

```

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

```

975 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
976 { \_color_backend_fill:n { #1 ~ k } }
977 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
978 { \_color_backend_fill:n { #1 ~ g } }
979 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
980 { \_color_backend_fill:n { #1 ~ rg } }
981 \cs_new_protected:Npn \_color_backend_fill:n #1
982 {
983   \tl_set:Nn \l__color_backend_fill_tl {#1}
984   \_kernel_color_backend_stack_push:nn \l__color_backend_stack_int
985   { #1 ~ \l__color_backend_stroke_tl }
986 }
987 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
988 { \_color_backend_stroke:n { #1 ~ K } }
989 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
990 { \_color_backend_stroke:n { #1 ~ G } }
991 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
992 { \_color_backend_stroke:n { #1 ~ RG } }
993 \cs_new_protected:Npn \_color_backend_stroke:n #1
994 {
995   \tl_set:Nn \l__color_backend_stroke_tl {#1}
996   \_kernel_color_backend_stack_push:nn \l__color_backend_stack_int
997   { \l__color_backend_fill_tl \c_space_tl #1 }
998 }

```

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

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
1000 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1001 { \_color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
1002 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1003 { \_color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
1004 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1005 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

```

\_color_backend_fill_reset:
  \_color_backend_stroke_reset:
1005 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
1006 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:

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

1007 </luatex | pdftex>
1008 <*dvips>

```

```

\_color_backend_fill_cmyk:n Fill color here is the same as general color except we skip the stroke part.
\_color_backend_fill_gray:n
\_color_backend_fill_rgb:n
  \_color_backend_fill:n
    \_color_backend_stroke_cmyk:n
    \_color_backend_stroke_gray:n
    \_color_backend_stroke_rgb:n
1009 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1010 { \_color_backend_fill:n { cmyk ~ #1 } }
1011 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1012 { \_color_backend_fill:n { gray ~ #1 } }
1013 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1014 { \_color_backend_fill:n { rgb ~ #1 } }
1015 \cs_new_protected:Npn \_color_backend_fill:n #1
1016 {
1017   \_kernel_backend_literal:n { color~push~ #1 }
1018 }
1019 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1020 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1021 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1022 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1023 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1024 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }

```

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

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
1025 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1026 { \_color_backend_fill:n { separation ~ #1 ~ #2 } }
1027 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1028 { \_kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1029 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1030 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

```

\_color_backend_fill_reset:
  \_color_backend_stroke_reset:
1031 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
1032 \cs_new_protected:Npn \_color_backend_stroke_reset: { }

```

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

1033 `</dvips>`

1034 `<*dvisvgm>`

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

```

1035 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1036 { \_color_backend_fill:n { cmyk ~ #1 } }
1037 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1038 { \_color_backend_fill:n { gray ~ #1 } }
1039 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1040 { \_color_backend_fill:n { rgb ~ #1 } }
1041 \cs_new_protected:Npn \_color_backend_fill:n #1
1042 {
1043   \_kernel_backend_literal:n { color~push~ #1 }
1044 }

```

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

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

```

1045 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1046 {
1047   \_color_backend_fill_cmyk:n {#1}
1048   \_kernel_backend_scope:n { stroke = "{?color}" }
1049   \_color_backend_reset:
1050 }
1051 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1052 {
1053   \use:e
1054   {
1055     \_color_backend_stroke_gray_aux:n
1056     { \fp_eval:n { 100 * (#1) } }
1057   }
1058 }
1059 \cs_new_protected:Npn \_color_backend_stroke_gray_aux:n #1
1060 { \_color_backend:nnn {#1} {#1} {#1} }
1061 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1062 { \_color_backend_rgb:w #1 \s_color_stop }
1063 \cs_new_protected:Npn \_color_backend_stroke_rgb:w
1064 #1 ~ #2 ~ #3 \s_color_stop
1065 {
1066   \use:e
1067   {
1068     \_color_backend:nnn
1069     { \fp_eval:n { 100 * (#1) } }
1070     { \fp_eval:n { 100 * (#2) } }
1071     { \fp_eval:n { 100 * (#3) } }
1072   }
1073 }
1074 \cs_new_protected:Npe \_color_backend:nnn #1#2#3
1075 {
1076   \_kernel_backend_scope:n

```

```

1077     {
1078         stroke =
1079         "
1080             rgb
1081             (
1082                 #1 \c_percent_str ,
1083                 #2 \c_percent_str ,
1084                 #3 \c_percent_str
1085             )
1086         "
1087     }
1088 }

```

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

At present, these are no-ops.

```

1089 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1090 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2 { }
1091 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1092 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

```

\_color_backend_fill_reset:
\_color_backend_stroke_reset:
1093 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
1094 \cs_new_protected:Npn \_color_backend_stroke_reset: { }

```

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

No support at present.

```

1095 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3 { }
1096 \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.`)

```

1097 </divsvgm>
1098 </package>

```

3.5 Font handling integration

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

```

1099 <lua>
1100 local l = lpeg
1101 local spaces = l.P' '^0
1102 local digit16 = l.R('09', 'af', 'AF')
1103
1104 local octet = digit16 * digit16 / function(s)
1105     return string.format('%.3g ', tonumber(s, 16) / 255)
1106 end
1107
1108 if luaotfload and luaotfload.set_transparent_colorstack then
1109     local htmlcolor = l.Cs(octet * octet * octet * -1 * l.Cc'rg')
1110     local color_export = {

```

```

1111 token.create'tex_endlocalcontrol:D',
1112 token.create'tex_hpack:D',
1113 token.new(0, 1),
1114 token.create'color_export:nnN',
1115 token.new(0, 1),
1116 '',
1117 token.new(0, 2),
1118 token.new(0, 1),
1119 'backend',
1120 token.new(0, 2),
1121 token.create'l_tmpa_tl',
1122 token.create'exp_after:wN',
1123 token.create'__color_select:nn',
1124 token.create'l_tmpa_tl',
1125 token.new(0, 2),
1126 }
1127 local group_end = token.create'group_end:'
1128 local value = (1 - l.P'')^0
1129 luatexbase.add_to_callback('luaotfload.parse_color', function (value)
1130 % Also allow HTML colors to preserve compatibility
1131 local html = htmlcolor:match(value)
1132 if html then return html end
1133
1134 % If no l3color named color with this name is known, check for defined xcolor colors
1135 local l3color_prop = token.get_macro(string.format('l__color_named_%s_prop', value))
1136 if l3color_prop == nil or l3color_prop == '' then
1137 local legacy_color_macro = token.create(string.format('\\color@%s', value))
1138 if legacy_color_macro.cmdname ~= 'undefined_cs' then
1139 token.put_next(legacy_color_macro)
1140 return token.scan_argument()
1141 end
1142 end
1143
1144 tex.runtoks(function()
1145 token.get_next()
1146 color_export[6] = value
1147 tex.sprint(-2, color_export)
1148 end)
1149 local list = token.scan_list()
1150 if not list.head or list.head.next
1151 or list.head.subtype ~= node.subtype'pdf_colorstack' then
1152 error'Unexpected backend behavior'
1153 end
1154 local cmd = list.head.data
1155 node.free(list)
1156 return cmd
1157 end, 'l3color')
1158 end
1159 </lua>
1160 <*luatex>
1161 <*package>
1162 \lua_load_module:n {l3backend-luatex}
1163 </package>

```

1164 \langle /luatex \rangle

4 l3backend-draw implementation

1165 \langle *package \rangle

1166 \langle @@=draw \rangle

4.1 dvips backend

1167 \langle *dvips \rangle

$\backslash_draw_backend_literal:n$ The same as literal PostScript: same arguments about positioning apply here.
 $\backslash_draw_backend_literal:e$ 1168 $\backslash cs_new_eq:Nn \backslash_draw_backend_literal:n \backslash_kernel_backend_literal_postscript:n$
 1169 $\backslash cs_generate_variant:Nn \backslash_draw_backend_literal:n \{ e \}$

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

$\backslash_draw_backend_begin:$ The $ps::[begin]$ special here deals with positioning but allows us to continue on to a
 $\backslash_draw_backend_end:$ 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.)

1170 $\backslash cs_new_protected:Npn \backslash_draw_backend_begin:$
 1171 $\{$
 1172 $\backslash_draw_backend_literal:n \{ [begin] \}$
 1173 $\backslash_draw_backend_literal:n \{ /draw.x\sim currentpoint~/draw.y\sim exch\sim def\sim def \}$
 1174 $\backslash_draw_backend_literal:n \{ @beginspecial \}$
 1175 $\}$
 1176 $\backslash cs_new_protected:Npn \backslash_draw_backend_end:$
 1177 $\{$
 1178 $\backslash_draw_backend_literal:n \{ @endspecial \}$
 1179 $\backslash_draw_backend_literal:n \{ [end] \}$
 1180 $\}$

(End of definition for $\backslash_draw_backend_begin:$ and $\backslash_draw_backend_end:.$)

$\backslash_draw_backend_scope_begin:$ Scope here may need to contain saved definitions, so the entire memory rather than just
 $\backslash_draw_backend_scope_end:$ the graphic state has to be sent to the stack.

1181 $\backslash cs_new_protected:Npn \backslash_draw_backend_scope_begin:$
 1182 $\{ \backslash_draw_backend_literal:n \{ save \} \}$
 1183 $\backslash cs_new_protected:Npn \backslash_draw_backend_scope_end:$
 1184 $\{ \backslash_draw_backend_literal:n \{ restore \} \}$

(End of definition for $\backslash_draw_backend_scope_begin:$ and $\backslash_draw_backend_scope_end:.$)

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

1185 $\backslash cs_new_protected:Npn \backslash_draw_backend_moveto:nn \#1\#2$
 1186 $\{$
 1187 $\backslash_draw_backend_literal:e$

```

1188     {
1189         \dim_to_decimal_in_bp:n {#1} ~
1190         \dim_to_decimal_in_bp:n {#2} ~ moveto
1191     }
1192 }
1193 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1194 {
1195     \__draw_backend_literal:e
1196     {
1197         \dim_to_decimal_in_bp:n {#1} ~
1198         \dim_to_decimal_in_bp:n {#2} ~ lineto
1199     }
1200 }
1201 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1202 {
1203     \__draw_backend_literal:e
1204     {
1205         \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1206         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1207         moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1208     }
1209 }
1210 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1211 {
1212     \__draw_backend_literal:e
1213     {
1214         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1215         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1216         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1217         curveto
1218     }
1219 }

```

(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
1220 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1221 { \bool_gset_true:N \g__draw_draw_eor_bool }
1222 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1223 { \bool_gset_false:N \g__draw_draw_eor_bool }
1224 \bool_new:N \g__draw_draw_eor_bool

```

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

```

\__draw_backend_closepath: Unlike PDF, PostScript doesn't track separate colors for strokes and other elements. It is
\__draw_backend_stroke: also desirable to have the clip keyword after a stroke or fill. To achieve those outcomes,
\__draw_backend_closestroke: there is some work to do. For color, the stroke color is simple but the fill one has to be
\__draw_backend_fill: inserted by hand. For clipping, the required ordering is achieved using a TEX switch.
\__draw_backend_fillstroke: All of the operations end with a new path instruction as they do not terminate (again in
\__draw_backend_clip: contrast to PDF).
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
1225 \cs_new_protected:Npn \__draw_backend_closepath:
1226 { \__draw_backend_literal:n { closepath } }
1227 \cs_new_protected:Npn \__draw_backend_stroke:

```

```

1228 {
1229   \_draw_backend_literal:n { gsave }
1230   \_draw_backend_literal:n { color.sc }
1231   \_draw_backend_literal:n { stroke }
1232   \_draw_backend_literal:n { grestore }
1233   \bool_if:NT \g__draw_draw_clip_bool
1234   {
1235     \_draw_backend_literal:e
1236     {
1237       \bool_if:NT \g__draw_draw_eor_bool { eo }
1238       clip
1239     }
1240   }
1241   \_draw_backend_literal:n { newpath }
1242   \bool_gset_false:N \g__draw_draw_clip_bool
1243 }
1244 \cs_new_protected:Npn \_draw_backend_closestroke:
1245 {
1246   \_draw_backend_closepath:
1247   \_draw_backend_stroke:
1248 }
1249 \cs_new_protected:Npn \_draw_backend_fill:
1250 {
1251   \_draw_backend_literal:e
1252   {
1253     \bool_if:NT \g__draw_draw_eor_bool { eo }
1254     fill
1255   }
1256   \bool_if:NT \g__draw_draw_clip_bool
1257   {
1258     \_draw_backend_literal:e
1259     {
1260       \bool_if:NT \g__draw_draw_eor_bool { eo }
1261       clip
1262     }
1263   }
1264   \_draw_backend_literal:n { newpath }
1265   \bool_gset_false:N \g__draw_draw_clip_bool
1266 }
1267 \cs_new_protected:Npn \_draw_backend_fillstroke:
1268 {
1269   \_draw_backend_literal:e
1270   {
1271     \bool_if:NT \g__draw_draw_eor_bool { eo }
1272     fill
1273   }
1274   \_draw_backend_literal:n { gsave }
1275   \_draw_backend_literal:n { color.sc }
1276   \_draw_backend_literal:n { stroke }
1277   \_draw_backend_literal:n { grestore }
1278   \bool_if:NT \g__draw_draw_clip_bool
1279   {
1280     \_draw_backend_literal:e
1281     {

```

```

1282         \bool_if:NT \g__draw_draw_eor_bool { eo }
1283         clip
1284     }
1285 }
1286 \__draw_backend_literal:n { newpath }
1287 \bool_gset_false:N \g__draw_draw_clip_bool
1288 }
1289 \cs_new_protected:Npn \__draw_backend_clip:
1290 { \bool_gset_true:N \g__draw_draw_clip_bool }
1291 \bool_new:N \g__draw_draw_clip_bool
1292 \cs_new_protected:Npn \__draw_backend_discardpath:
1293 {
1294     \bool_if:NT \g__draw_draw_clip_bool
1295     {
1296         \__draw_backend_literal:e
1297         {
1298             \bool_if:NT \g__draw_draw_eor_bool { eo }
1299             clip
1300         }
1301     }
1302     \__draw_backend_literal:n { newpath }
1303     \bool_gset_false:N \g__draw_draw_clip_bool
1304 }

```

(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:
1305 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1306 {
1307     \__draw_backend_literal:e
1308     {
1309         [
1310             \exp_args:Nf \use:n
1311             { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1312         ] ~
1313         \dim_to_decimal_in_bp:n {#2} ~ setdash
1314     }
1315 }
1316 \cs_new:Npn \__draw_backend_dash:n #1
1317 { ~ \dim_to_decimal_in_bp:n {#1} }
1318 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1319 {
1320     \__draw_backend_literal:e
1321     { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1322 }
1323 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1324 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1325 \cs_new_protected:Npn \__draw_backend_cap_butt:
1326 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1327 \cs_new_protected:Npn \__draw_backend_cap_round:
1328 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1329 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1330 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1331 \cs_new_protected:Npn \__draw_backend_join_miter:

```

```

1332 { \_draw_backend_literal:n { 0 ~ setlinejoin } }
1333 \cs_new_protected:Npn \_draw_backend_join_round:
1334 { \_draw_backend_literal:n { 1 ~ setlinejoin } }
1335 \cs_new_protected:Npn \_draw_backend_join_bevel:
1336 { \_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.

```

1337 \cs_new_protected:Npn \_draw_backend_transform:nnnn #1#2#3#4
1338 {
1339   \_draw_backend_literal:n
1340   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1341 }
1342 \cs_new_protected:Npn \_draw_backend_shift:nn #1#2
1343 {
1344   \_draw_backend_literal:n
1345   { [ 1 ~ 0 ~ 0 ~ 1 ~ #1 ~ #2 ] ~ concat }
1346 }

```

(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`.

```

1347 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
1348 {
1349   \_draw_backend_literal:n { save }
1350   \_draw_backend_literal:n { 72~Resolution~div~72~VResolution~div~neg~scale }
1351   \_draw_backend_literal:n { magscale { 1~DVImag~div~dup~scale } if }
1352   \_draw_backend_literal:n { draw.x~neg~draw.y~neg~translate }
1353   \_draw_backend_literal:n { [end] }
1354   \_draw_backend_literal:n { [begin] }
1355   \_draw_backend_literal:n { save }
1356   \_draw_backend_literal:n { currentpoint }
1357   \_draw_backend_literal:n { currentpoint~translate }
1358   \_draw_backend_transform:nnnn { 1 } { 0 } { 0 } { -1 }
1359   \_draw_backend_transform:nnnn { #2 } { #3 } { #4 } { #5 }
1360   \_draw_backend_transform:nnnn { 1 } { 0 } { 0 } { -1 }
1361   \_draw_backend_literal:n { neg~exch~neg~exch~translate }
1362   \_draw_backend_literal:n { [end] }
1363   \hbox_overlap_right:n { \box_use:N #1 }
1364   \_draw_backend_literal:n { [begin] }

```

```

1365     \__draw_backend_literal:n { restore }
1366     \__draw_backend_literal:n { [end] }
1367     \__draw_backend_literal:n { [begin] }
1368     \__draw_backend_literal:n { restore }
1369 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1370 </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.

```

1371 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

```

\__draw_backend_literal:n Pass data through using a dedicated interface.
\__draw_backend_literal:e
1372 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1373 \cs_new_eq:NN \__draw_backend_literal:e \__kernel_backend_literal_pdf:e

```

(End of definition for __draw_backend_literal:n.)

```

\__draw_backend_begin: No special requirements here, so simply set up a drawing scope.
\__draw_backend_end:
1374 \cs_new_protected:Npn \__draw_backend_begin:
1375 { \__draw_backend_scope_begin: }
1376 \cs_new_protected:Npn \__draw_backend_end:
1377 { \__draw_backend_scope_end: }

```

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

```

\__draw_backend_scope_begin: Use the backend-level scope mechanisms.
\__draw_backend_scope_end:
1378 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1379 \cs_new_eq:NN \__draw_backend_scope_end: \__kernel_backend_scope_end:

```

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

```

\__draw_backend_moveto:nn Path creation operations all resolve directly to PDF primitive steps, with only the need
\__draw_backend_lineto:nn to convert to bp.
\__draw_backend_curveto:nnnnnn
\__draw_backend_rectangle:nnnn

```

```

1380 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1381 {
1382     \__draw_backend_literal:e
1383     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1384 }
1385 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1386 {
1387     \__draw_backend_literal:e
1388     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1389 }
1390 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1391 {
1392     \__draw_backend_literal:e
1393     {
1394         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~

```

```

1395         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1396         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1397         c
1398     }
1399 }
1400 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1401 {
1402     \__draw_backend_literal:e
1403     {
1404         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1405         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1406         re
1407     }
1408 }

```

(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
1409 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1410 { \bool_gset_true:N \g__draw_draw_eor_bool }
1411 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1412 { \bool_gset_false:N \g__draw_draw_eor_bool }
1413 \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:
1414 \cs_new_protected:Npn \__draw_backend_closepath:
1415 { \__draw_backend_literal:n { h } }
1416 \cs_new_protected:Npn \__draw_backend_stroke:
1417 { \__draw_backend_literal:n { S } }
1418 \cs_new_protected:Npn \__draw_backend_closestroke:
1419 { \__draw_backend_literal:n { s } }
1420 \cs_new_protected:Npn \__draw_backend_fill:
1421 {
1422     \__draw_backend_literal:e
1423     { f \bool_if:NT \g__draw_draw_eor_bool * }
1424 }
1425 \cs_new_protected:Npn \__draw_backend_fillstroke:
1426 {
1427     \__draw_backend_literal:e
1428     { B \bool_if:NT \g__draw_draw_eor_bool * }
1429 }
1430 \cs_new_protected:Npn \__draw_backend_clip:
1431 {
1432     \__draw_backend_literal:e
1433     { W \bool_if:NT \g__draw_draw_eor_bool * }
1434 }
1435 \cs_new_protected:Npn \__draw_backend_discardpath:
1436 { \__draw_backend_literal:n { n } }

```

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

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

```

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

```

```

1437 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1438 {
1439   \__draw_backend_literal:e
1440   {
1441     [
1442       \exp_args:Nf \use:n
1443       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1444     ] ~
1445     \dim_to_decimal_in_bp:n {#2} ~ d
1446   }
1447 }
1448 \cs_new:Npn \__draw_backend_dash:n #1
1449 { ~ \dim_to_decimal_in_bp:n {#1} }
1450 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1451 {
1452   \__draw_backend_literal:e
1453   { \dim_to_decimal_in_bp:n {#1} ~ w }
1454 }
1455 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1456 { \__draw_backend_literal:e { #1 ~ M } }
1457 \cs_new_protected:Npn \__draw_backend_cap_butt:
1458 { \__draw_backend_literal:n { 0 ~ J } }
1459 \cs_new_protected:Npn \__draw_backend_cap_round:
1460 { \__draw_backend_literal:n { 1 ~ J } }
1461 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1462 { \__draw_backend_literal:n { 2 ~ J } }
1463 \cs_new_protected:Npn \__draw_backend_join_miter:
1464 { \__draw_backend_literal:n { 0 ~ j } }
1465 \cs_new_protected:Npn \__draw_backend_join_round:
1466 { \__draw_backend_literal:n { 1 ~ j } }
1467 \cs_new_protected:Npn \__draw_backend_join_bevel:
1468 { \__draw_backend_literal:n { 2 ~ j } }

```

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

```

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

```

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

```

1469 \cs_new_protected:Npn \__draw_backend_transform:nnnn #1#2#3#4
1470 {
1471   <*luatex | pdftex>
1472   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1473   </luatex | pdftex>
1474   <*dvipdfmx | xetex>
1475   \str_if_eq:nnTF { #2 ~ #3 } { 0 ~ 0 }
1476   {

```

```

1477     \_kernel_backend_literal:n { x:rotate~0 }
1478     \_kernel_backend_literal:n { x:scale~#1~#4 }
1479     \_kernel_backend_literal:n { x:rotate~0 }
1480   }
1481   {
1482     \_draw_backend_transform_decompose:nnnnN {#1} {#2} {#3} {#4}
1483     \_draw_backend_transform_aux:nnnn
1484   }
1485 </dviptdmx | xetex>
1486 }
1487 <*dviptdmx | xetex>
1488 \cs_new_protected:Npn \_draw_backend_transform_aux:nnnn #1#2#3#4
1489 {
1490   \_kernel_backend_literal:e
1491   {
1492     x:rotate~
1493     \fp_compare:nNnTF {#1} = \c_zero_fp
1494       { 0 }
1495       { \fp_eval:n { round ( -#1 , 5 ) } }
1496   }
1497   \_kernel_backend_literal:e
1498   {
1499     x:scale~
1500     \fp_eval:n { round ( #2 , 5 ) } ~
1501     \fp_eval:n { round ( #3 , 5 ) }
1502   }
1503   \_kernel_backend_literal:e
1504   {
1505     x:rotate~
1506     \fp_compare:nNnTF {#4} = \c_zero_fp
1507       { 0 }
1508       { \fp_eval:n { round ( -#4 , 5 ) } }
1509   }
1510 }
1511 </dviptdmx | xetex>

```

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

```

1512 \cs_new_protected:Npn \_draw_backend_shift:nn #1#2
1513 {
1514   \_draw_backend_literal:n
1515   { 1 ~ 0 ~ 0 ~ 1 ~ #1 ~ #2 ~ cm }
1516 }

```

(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.

```

1517 <*dvipdfmx | xetex>
1518 \cs_new_protected:Npn \__draw_backend_transform_decompose:nnnnN #1#2#3#4#5
1519 {
1520   \use:e
1521   {
1522     \__draw_backend_transform_decompose_auxi:nnnnN
1523     { \fp_eval:n { (#1 + #4) / 2 } }
1524     { \fp_eval:n { (#1 - #4) / 2 } }
1525     { \fp_eval:n { (#3 + #2) / 2 } }
1526     { \fp_eval:n { (#3 - #2) / 2 } }
1527   }
1528   #5
1529 }
1530 \cs_new_protected:Npn \__draw_backend_transform_decompose_auxi:nnnnN #1#2#3#4#5
1531 {
1532   \use:e
1533   {
1534     \__draw_backend_transform_decompose_auxii:nnnnN
1535     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1536     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1537     { \fp_eval:n { atand ( #3 , #2 ) } }
1538     { \fp_eval:n { atand ( #4 , #1 ) } }
1539   }
1540   #5
1541 }
1542 \cs_new_protected:Npn \__draw_backend_transform_decompose_auxii:nnnnN #1#2#3#4#5
1543 {
1544   \use:e
1545   {
1546     \__draw_backend_transform_decompose_auxiii:nnnnN
1547     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1548     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1549     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1550     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1551   }

```

```

1552         #5
1553     }
1554 \cs_new_protected:Npn \__draw_backend_transform_decompose_auxiii:nnnnN #1#2#3#4#5
1555 {
1556     \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1557     { #5 {#1} {#2} {#3} {#4} }
1558     { #5 {#1} {#3} {#2} {#4} }
1559 }
1560 </dvipdfmx | xetex>

```

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

__draw_backend_box_use:Nnnnn

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

```

1561 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1562 {
1563     \__kernel_backend_scope_begin:
1564     < *luatex | pdftex >
1565     \__kernel_backend_matrix:n { #2 ~ #3 ~ #4 ~ #5 }
1566     < /luatex | pdftex >
1567     < *dvipdfmx | xetex >
1568     \__kernel_backend_literal:n
1569     { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1570     < /dvipdfmx | xetex >
1571     \hbox_overlap_right:n { \box_use:N #1 }
1572     < *dvipdfmx | xetex >
1573     \__kernel_backend_literal:n { pdf:etrans }
1574     < /dvipdfmx | xetex >
1575     \__kernel_backend_scope_end:
1576 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1577 < /dvipdfmx | luatex | pdftex | xetex >

```

4.3 dvisvgm backend

```

1578 < *dvisvgm >

```

The same as the more general literal call.

__draw_backend_literal:n
__draw_backend_literal:e

```

1579 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1580 \cs_generate_variant:Nn \__draw_backend_literal:n { e }

```

(End of definition for __draw_backend_literal:n.)

__draw_backend_scope_begin:
__draw_backend_scope_end:

```

1581 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1582 \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

```

1583 \cs_new_protected:Npn \__draw_backend_begin:
1584 {
1585   \__kernel_backend_scope_begin:
1586   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1587 }
1588 \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`

```

1589 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1590 {
1591   \__draw_backend_add_to_path:n
1592   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1593 }
1594 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1595 {
1596   \__draw_backend_add_to_path:n
1597   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1598 }
1599 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1600 {
1601   \__draw_backend_add_to_path:n
1602   {
1603     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1604     h ~ \dim_to_decimal:n {#3} ~
1605     v ~ \dim_to_decimal:n {#4} ~
1606     h ~ \dim_to_decimal:n { -#3 } ~
1607     Z
1608   }
1609 }
1610 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1611 {
1612   \__draw_backend_add_to_path:n
1613   {
1614     C ~
1615     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1616     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1617     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1618   }
1619 }
1620 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1621 {
1622   \tl_gset:Nx \g__draw_backend_path_tl
1623   {
1624     \g__draw_backend_path_tl
1625     \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1626     #1

```

```

1627     }
1628   }
1629   \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:
1630 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1631   { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1632 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1633   { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

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

```

\__draw_backend_path:n
\__draw_backend_closepath:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int

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.
1634 \cs_new_protected:Npn \__draw_backend_closepath:
1635   { \__draw_backend_add_to_path:n { Z } }
1636 \cs_new_protected:Npn \__draw_backend_path:n #1
1637   {
1638     \bool_if:NTF \g__draw_draw_clip_bool
1639     {
1640       \int_gincr:N \g__kernel_clip_path_int
1641       \__draw_backend_literal:e
1642       {
1643         < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1644         { ?nl }
1645         <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1646         < /clipPath > { ? nl }
1647         <
1648         use~xlink:href =
1649         "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1650         #1
1651         />
1652       }
1653       \__kernel_backend_scope:e
1654       {
1655         clip-path =
1656         "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int )"
1657       }
1658     }
1659     {
1660       \__draw_backend_literal:e
1661       { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1662     }
1663     \tl_gclear:N \g__draw_backend_path_tl
1664     \bool_gset_false:N \g__draw_draw_clip_bool
1665   }
1666   \int_new:N \g__draw_backend_path_int
1667   \cs_new_protected:Npn \__draw_backend_stroke:
1668     { \__draw_backend_path:n { style="fill:none" } }

```

```

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

```

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

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_dash_aux:nn
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1702 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1703 {
1704   \use:e
1705   {
1706     \__draw_backend_dash_aux:nn
1707     { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1708     { \dim_to_decimal:n {#2} }
1709   }
1710 }
1711 \cs_new:Npn \__draw_backend_dash:n #1
1712 { , \dim_to_decimal_in_bp:n {#1} }
1713 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1714 {
1715   \__kernel_backend_scope:e
1716   {
1717     stroke-dasharray =

```

```

1718         "
1719         \tl_if_empty:nTF {#1}
1720         { none }
1721         { \use_none:n #1 }
1722     " ~
1723     stroke-offset=" #2 "
1724 }
1725 }
1726 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1727 { \__kernel_backend_scope:e { stroke-width=" \dim_to_decimal:n {#1} " } }
1728 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1729 { \__kernel_backend_scope:e { stroke-miterlimit=" #1 " } }
1730 \cs_new_protected:Npn \__draw_backend_cap_but:
1731 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1732 \cs_new_protected:Npn \__draw_backend_cap_round:
1733 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1734 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1735 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1736 \cs_new_protected:Npn \__draw_backend_join_miter:
1737 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1738 \cs_new_protected:Npn \__draw_backend_join_round:
1739 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1740 \cs_new_protected:Npn \__draw_backend_join_bevel:
1741 { \__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.

```

1742 \cs_new_protected:Npn \__draw_backend_transform:nnnn #1#2#3#4
1743 {
1744     \__kernel_backend_scope:n
1745     {
1746         transform =
1747         " matrix ( #1 , #2 , #3 , #4 , 0pt , 0pt ) "
1748     }
1749 }
1750 \cs_new_protected:Npn \__draw_backend_shift:nn #1#2
1751 {
1752     \__kernel_backend_scope:n
1753     {
1754         transform =
1755         " matrix ( 1 , 0 , 0 , 1 , #1pt , #2pt ) "
1756     }
1757 }

```

(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.

```

1758 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1759 {
1760     \__kernel_backend_scope_begin:
1761     \__draw_backend_transform:nnnn {#2} {#3} {#4} {#5}

```

```

1762 \__kernel_backend_literal_svg:n
1763 {
1764   < g~
1765     stroke="none"~
1766     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1767   >
1768   }
1769   \box_set_wd:Nn #1 { Opt }
1770   \box_set_ht:Nn #1 { Opt }
1771   \box_set_dp:Nn #1 { Opt }
1772   \box_use:N #1
1773   \__kernel_backend_literal_svg:n { </g> }
1774   \__kernel_backend_scope_end:
1775 }

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```

1776 </dvisvgm>
1777 </package>

```

5 l3backend-graphics implementation

```

1778 <*package>
1779 <@@=graphics>

```

5.1 dvips backend

```

1780 <*dvips>

```

\l_graphics_search_ext_seq

```

1781 \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.

```

1782 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1783 \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.

```

1784 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1785 {
1786   \__kernel_backend_literal:e
1787   {
1788     PSfile = #1 \c_space_tl
1789     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1790     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1791     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1792     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1793   }
1794 }
1795 \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

```
1796 \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n
```

(End of definition for _graphics_backend_get_pagecount:n.)

```
1797 </dvips>
```

5.2 LuaTeX and pdfTeX backends

```
1798 <*luatex | pdftex>
```

\l_graphics_search_ext_seq

```
1799 \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1800 { .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.

```
1801 \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.

```
1802 \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.

```
1803 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
```

```
1804 {
1805   \int_zero:N \l__graphics_page_int
1806   \tl_clear:N \l__graphics_pagebox_tl
1807   \bool_set_false:N \l__graphics_transgroup_bool
1808   \tl_set:Nx \l__graphics_attr_tl
1809   {
1810     \tl_if_empty:NF \l__graphics_decodearray_str
1811     { :D \l__graphics_decodearray_str }
1812     \bool_if:NT \l__graphics_interpolate_bool
1813     { :I }
1814     \str_if_empty:NF \l__graphics_pdf_str
1815     { :X \l__graphics_pdf_str }
1816   }
1817   \_graphics_backend_getbb_auxi:n {#1}
1818 }
```

```
1819 \cs_new_eq:NN \_graphics_backend_getbb_jpeg:n \_graphics_backend_getbb_jpg:n
```

```
1820 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
```

```

1821 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1822 {
1823   \tl_clear:N \l__graphics_decodearray_str
1824   \bool_set_true:N \l__graphics_transgroup_bool
1825   \bool_set_false:N \l__graphics_interpolate_bool
1826   \tl_set:Ne \l__graphics_attr_tl
1827   {
1828     : \l__graphics_pagebox_tl
1829     \int_compare:nNnT \l__graphics_page_int > 1
1830     { :P \int_use:N \l__graphics_page_int }
1831     \str_if_empty:NF \l__graphics_pdf_str
1832     { :X \l__graphics_pdf_str }
1833   }
1834   \__graphics_backend_getbb_auxi:n {#1}
1835 }
1836 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1837 {
1838   \__graphics_bb_restore:eF { #1 \l__graphics_attr_tl }
1839   { \__graphics_backend_getbb_auxii:n {#1} }
1840 }

```

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.

```

1841 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1842 {
1843   \exp_args:Ne \__graphics_backend_getbb_auxiii:n
1844   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1845   \int_const:cn { c__graphics_ #1 \l__graphics_attr_tl _int }
1846   { \tex_the:D \tex_pdflastximage:D }
1847   \__graphics_bb_save:e { #1 \l__graphics_attr_tl }
1848 }
1849 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1850 {
1851   \tex_immediate:D \tex_pdfximage:D
1852   \bool_lazy_any:nT
1853   {
1854     { \l__graphics_interpolate_bool }
1855     { \l__graphics_transgroup_bool }
1856     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1857     { ! \str_if_empty_p:N \l__graphics_pdf_str }
1858   }
1859   {
1860     attr ~
1861     {
1862       \tl_if_empty:NF \l__graphics_decodearray_str
1863       { /Decode~[ \l__graphics_decodearray_str ] }
1864       \bool_if:NT \l__graphics_transgroup_bool
1865       { /Group << /S /Transparency /K ~ false /I ~ false >> }
1866       \bool_if:NT \l__graphics_interpolate_bool
1867       { /Interpolate~true }
1868       \l__graphics_pdf_str

```

```

1869     }
1870   }
1871   \int_compare:nNnT \l__graphics_page_int > 0
1872     { page ~ \int_use:N \l__graphics_page_int }
1873   \tl_if_empty:NF \l__graphics_pagebox_tl
1874     { \l__graphics_pagebox_tl }
1875   {#1}
1876   \hbox_set:Nn \l__graphics_tmp_box
1877     { \tex_pdfrefximage:D \tex_pdflastximage:D }
1878   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_tmp_box }
1879   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_tmp_box }
1880 }
1881 \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.

```

1882 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1883 {
1884   \tex_pdfrefximage:D
1885   \int_use:c { c__graphics_ #1 \l__graphics_attr_tl _int }
1886 }
1887 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1888 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1889 \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.

```

1890 \sys_if_shell:T
1891 {
1892   \str_new:N \l__graphics_backend_dir_str
1893   \str_new:N \l__graphics_backend_name_str
1894   \str_new:N \l__graphics_backend_ext_str
1895   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1896   {
1897     \file_parse_full_name:nNNN {#1}
1898     \l__graphics_backend_dir_str
1899     \l__graphics_backend_name_str
1900     \l__graphics_backend_ext_str
1901     \exp_args:Ne \__graphics_backend_getbb_eps:nn
1902     {
1903       \exp_args:Ne \__kernel_file_name_quote:n
1904       {
1905         \l__graphics_backend_name_str
1906         - \str_tail:N \l__graphics_backend_ext_str
1907         -converted-to.pdf
1908       }
1909     }
1910     {#1}

```

```

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

```

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

__graphics_backend_get_pagecount:n

Simply load and store.

```

1939 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1940 {
1941     \tex_pdfximage:D {#1}
1942     \int_const:cn { c__graphics_ #1 _pages_int }
1943     { \int_use:N \tex_pdflastximagepages:D }
1944 }

```

(End of definition for __graphics_backend_get_pagecount:n.)

1945 </luatex | pdftex>

5.3 dvipdfmx backend

1946 <*dvipdfmx | xetex>

\l_graphics_search_ext_seq

```

1947 \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1948 { .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

1949 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1950 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
1951 <*dvipdfmx>
1952 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1953 {
1954   \int_zero:N \l__graphics_page_int
1955   \tl_clear:N \l__graphics_pagebox_tl
1956   \__graphics_extract_bb:n {#1}
1957 }
1958 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1959 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1960 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
1961 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1962 {
1963   \tl_clear:N \l__graphics_decodearray_str
1964   \bool_set_false:N \l__graphics_interpolate_bool
1965   \__graphics_extract_bb:n {#1}
1966 }
1967 </dvipdfmx>

```

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

(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.

```
1968 \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.

```
1969 \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_auxiii:en
\_graphics_backend_include_auxiiii:nn

1970 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1971 {
1972   \__kernel_backend_literal:e
1973   {
1974     PSfile = #1 \c_space_tl
1975     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1976     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1977     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1978     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1979   }
1980 }
1981 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n

```

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

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.

```

1982 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1983 {
1984   \bool_set_false:N \l__graphics_transgroup_bool
1985   \__graphics_backend_include_auxi:n {#1}
1986 }
1987 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1988 \cs_new_eq:NN \__graphics_backend_include_bmp:n \__graphics_backend_include_jpg:n
1989 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
1990 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1991 {
1992   \bool_set_true:N \l__graphics_transgroup_bool
1993   \__graphics_backend_include_auxi:n {#1}
1994 }
1995 \cs_new_protected:Npn \__graphics_backend_include_auxi:n #1
1996 {
1997   \__graphics_backend_include_auxii:en
1998   {
1999     \tl_if_empty:NF \l__graphics_pagebox_tl
2000     { : \l__graphics_pagebox_tl }
2001     \int_compare:nNnT \l__graphics_page_int > 1
2002     { :P \int_use:N \l__graphics_page_int }
2003     \tl_if_empty:NF \l__graphics_decodearray_str
2004     { :D \l__graphics_decodearray_str }
2005     \bool_if:NT \l__graphics_interpolate_bool
2006     { :I }
2007   }
2008   {#1}
2009 }
2010 \cs_new_protected:Npn \__graphics_backend_include_auxii:nn #1#2
2011 {
2012   \int_if_exist:cTF { c__graphics_ #2#1 _int }
2013   {
2014     \__kernel_backend_literal:e
2015     { pdf:useobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
2016   }
2017   { \__graphics_backend_include_auxiii:nn {#2} {#1} }
2018 }
2019 \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.

```

2020 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nn #1#2
2021 {
2022   \int_gincr:N \g__graphics_track_int
2023   \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
2024   \__kernel_backend_literal:e
2025   {
2026     pdf:image ~
2027     @graphic \int_use:c { c__graphics_ #1#2 _int } ~
2028     \int_compare:nNnT \l__graphics_page_int > 1
2029     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2030     \tl_if_empty:NF \l__graphics_pagebox_tl

```

```

2031     {
2032         pagebox ~ \l__graphics_pagebox_tl \c_space_tl
2033         bbox ~
2034             \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2035             \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2036             \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2037             \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
2038     }
2039     (#1)
2040     \bool_lazy_any:nT
2041     {
2042         { \l__graphics_interpolate_bool }
2043         { \l__graphics_transgroup_bool }
2044         { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
2045     }
2046     {
2047         <<
2048             \tl_if_empty:NF \l__graphics_decodearray_str
2049             { /Decode~[ \l__graphics_decodearray_str ] }
2050             \bool_if:NT \l__graphics_transgroup_bool
2051             { /Group << /S /Transparency /K ~ false /I ~ false >> }
2052             \bool_if:NT \l__graphics_interpolate_bool
2053             { /Interpolate~true }
2054         >>
2055     }
2056 }
2057 }

```

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

__graphics_backend_get_pagecount:n

```

2058 <*\dviptdmx>
2059 \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n
2060 </dviptdmx>

```

(End of definition for __graphics_backend_get_pagecount:n.)

```

2061 </dviptdmx | xetex>

```

5.4 X_YTeX backend

```

2062 <*\xetex>

```

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

```

2063 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2064 {
2065     \int_zero:N \l__graphics_page_int
2066     \tl_clear:N \l__graphics_pagebox_tl
2067     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2068 }
2069 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2070 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg: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
\__graphics_backend_getbb_auxi:nN
\__graphics_backend_getbb_auxii:nnN
\__graphics_backend_getbb_auxiii:nN
\__graphics_backend_getbb_auxiv:nnN
\__graphics_backend_getbb_auxv:nnN
\__graphics_backend_getbb_auxvi:nnN
\__graphics_backend_getbb_auxvii:nnN
\__graphics_backend_getbb_auxviii:nnN
\__graphics_backend_getbb_auxviiii:nnN
\__graphics_backend_getbb_auxv:nnN
\__graphics_backend_getbb_pagebox:w

```

```

2071 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2072 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2073 {
2074   \tl_clear:N \l__graphics_decodearray_str
2075   \bool_set_false:N \l__graphics_interpolate_bool
2076   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdf:file:D
2077 }
2078 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2079 {
2080   \int_compare:nNnTF \l__graphics_page_int > 1
2081     { \__graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2082     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2083 }
2084 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2085 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2086 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2087 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2088 {
2089   \tl_if_empty:NTF \l__graphics_pagebox_tl
2090     { \__graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2091     { \__graphics_backend_getbb_auxv:nNnn {#1} #2 {#3} {#4} }
2092 }
2093 }
2094 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2095 {
2096   \use:e
2097   {
2098     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2099     {
2100       #5
2101       \tl_if_blank:nF {#1}
2102         { \c_space_tl \__graphics_backend_getbb_pagebox:w #1 }
2103     }
2104   }
2105 }
2106 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2107 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2108 {
2109   \__graphics_bb_restore:nF {#1#3}
2110   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2111 }
2112 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2113 {
2114   \hbox_set:Nn \l__graphics_tmp_box { #2 #1 ~ #4 }
2115   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_tmp_box }
2116   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_tmp_box }
2117   \__graphics_bb_save:n {#1#3}
2118 }
2119 \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
2120 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1

```

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

```

2121 {
2122   \int_const:cn { c__graphics_ #1 _pages_int }
2123   {
2124     \int_max:nn
2125     { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2126     { 1 }
2127   }
2128 }

```

(End of definition for __graphics_backend_get_pagecount:n.)

```
2129 </xetex>
```

5.5 dvisvgm backend

```
2130 <*dvisvgm>
```

```
\l_graphics_search_ext_seq
```

```

2131 \seq_set_from_clist:Nn \l_graphics_search_ext_seq
2132 { .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.

```

2133 \cs_new_protected:Npn \__graphics_backend_getbb_svg:n #1
2134 {
2135   \__graphics_bb_restore:nF {#1}
2136   {
2137     \ior_open:Nn \l__graphics_tmp_ior {#1}
2138     \ior_if_eof:NTF \l__graphics_tmp_ior
2139     { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2140     {
2141       \dim_zero:N \l__graphics_llx_dim
2142       \dim_zero:N \l__graphics_lly_dim
2143       \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2144       \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2145       \ior_str_map_inline:Nn \l__graphics_tmp_ior
2146       {
2147         \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2148         {
2149           \__graphics_backend_getbb_svg_auxi:nNn
2150           { width } \l__graphics_urx_dim {##1}
2151         }
2152         \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2153         {
2154           \__graphics_backend_getbb_svg_auxi:nNn
2155           { height } \l__graphics_ury_dim {##1}
2156         }
2157         \bool_lazy_and:nnF
2158         { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2159         { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2160         { \ior_map_break: }

```

```

2161         }
2162         \__graphics_bb_save:n {#1}
2163     }
2164     \ior_close:N \l__graphics_tmp_ior
2165 }
2166 }
2167 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2168 {
2169     \use:e
2170     {
2171         \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxii:w
2172             ##1 \tl_to_str:n {#1} = ##2 \tl_to_str:n {#1} = ##3
2173         \s__graphics_stop
2174     }
2175     {
2176         \tl_if_blank:nF {##2}
2177         {
2178             \peek_remove_spaces:n
2179             {
2180                 \peek_meaning:NTF ' % '
2181                 { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2182                 {
2183                     \peek_meaning:NTF " % "
2184                     { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2185                     { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2186                 }
2187             }
2188             ##2 \s__graphics_stop
2189         }
2190     }
2191     \use:e
2192     {
2193         \__graphics_backend_getbb_svg_auxii:w #3
2194         \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2195         \s__graphics_stop
2196     }
2197 }
2198 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2199 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop
2200 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2201 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2202 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2203 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2204 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2205 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2206 {
2207     \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2208     \l__graphics_tmp_dim #2 bp \scan_stop:
2209     \dim_set_eq:NN #1 \l__graphics_tmp_dim
2210 }
2211 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvii:w #1 \scan_stop: { }

```

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

`_graphics_backend_getbb_eps:n` Simply use the generic function.
`_graphics_backend_getbb_ps:n` 2212 `\cs_new_eq:NN _graphics_backend_getbb_eps:n _graphics_read_bb:n`
2213 `\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` These can be included by extracting the bounding box data.
`_graphics_backend_getbb_jpg:n` 2214 `\cs_new_protected:Npn _graphics_backend_getbb_jpg:n #1`
`_graphics_backend_getbb_jpeg:n` 2215 `{`
2216 `\int_zero:N \l__graphics_page_int`
2217 `\tl_clear:N \l__graphics_pagebox_tl`
2218 `_graphics_extract_bb:n {#1}`
2219 `}`
2220 `\cs_new_eq:NN _graphics_backend_getbb_jpeg:n _graphics_backend_getbb_jpg:n`
2221 `\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
2222 `\cs_new_protected:Npn _graphics_backend_getbb_pdf:n #1`
2223 `{`
2224 `\tl_clear:N \l__graphics_decodearray_str`
2225 `\bool_set_false:N \l__graphics_interpolate_bool`
2226 `_graphics_extract_bb:n {#1}`
2227 `}`
(End of definition for `_graphics_backend_getbb_pdf:n`.)

`_graphics_backend_include_eps:n` The special syntax is relatively clear here: remember we need PostScript sizes here. (This
`_graphics_backend_include_ps:n` is the same as the `dvips` code.)
`_graphics_backend_include_pdf:n` 2228 `\cs_new_protected:Npn _graphics_backend_include_eps:n #1`
`_graphics_backend_include:nn` 2229 `{ _graphics_backend_include:nn { PSfile } {#1} }`
2230 `\cs_new_eq:NN _graphics_backend_include_ps:n _graphics_backend_include_eps:n`
2231 `\cs_new_protected:Npn _graphics_backend_include_pdf:n #1`
2232 `{ _graphics_backend_include:nn { pdffile } {#1} }`
2233 `\cs_new_protected:Npn _graphics_backend_include:nn #1#2`
2234 `{`
2235 `__kernel_backend_literal:e`
2236 `{`
2237 `#1 = #2 \c_space_tl`
2238 `llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl`
2239 `lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl`
2240 `urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl`
2241 `ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim`
2242 `}`
2243 `}`
(End of definition for `_graphics_backend_include_eps:n` and others.)

`_graphics_backend_include_svg:n` The backend here has built-in support for basic graphic inclusion (see `dvissvgm.def` for a
`_graphics_backend_include_png:n` more complex approach, needed if clipping, etc., is covered at the graphic backend level).
`_graphics_backend_include_jpg:n` We have to deal with the fact that the image reference point is at the *top*, so there is a
`_graphics_backend_include_jpeg:n` need for a vertical shift to put it in the right place. The other issue is that `#1` must be
`_graphics_backend_include_dequote:w`

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.

```

2244 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2245 {
2246   \box_move_up:nn { \l__graphics_ury_dim }
2247   {
2248     \hbox:n
2249     {
2250       \__kernel_backend_literal:e
2251       {
2252         dvisvgm:img~
2253         \dim_to_decimal:n { \l__graphics_urx_dim } ~
2254         \dim_to_decimal:n { \l__graphics_ury_dim } ~
2255         \__graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2256       }
2257     }
2258   }
2259 }
2260 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_svg:n
2261 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_svg:n
2262 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_svg:n
2263 \cs_new:Npn \__graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2264 {#2}

```

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

`__graphics_backend_get_pagecount:n`

```

2265 \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n
2266 \end{document}
2267 \end{package}

```

6 l3backend-pdf implementation

```

2268 \begin{package}
2269 \begin{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

```

2270 \begin{dvips}

```

`__pdf_backend_pdfmark:n`
`__pdf_backend_pdfmark:e`

Used often enough it should be a separate function.

```

2271 \cs_new_protected:Npn \__pdf_backend_pdfmark:n #1
2272 { \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2273 \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
2274 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2275 { \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2276 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2277 { \__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
2278 \cs_new_protected:Npn \__pdf_backend_object_new:
2279 { \int_gincr:N \g__pdf_backend_object_int }
2280 \cs_new:Npn \__pdf_backend_object_ref:n #1 { { pdf.obj #1 } }
2281 \cs_new_eq:NN \__pdf_backend_object_id:n \__pdf_backend_object_ref:n

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

```

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

```

\__pdf_backend_object_write:nnn
\__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
2282 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2283 {
2284   \__pdf_backend_object_write_aux:nnn
2285   { \__pdf_backend_object_ref:n {#1} }
2286   {#2} {#3}
2287 }
2288 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2289 \cs_new_protected:Npn \__pdf_backend_object_write_aux:nnn #1#2#3
2290 {
2291   \__pdf_backend_pdfmark:e
2292   {
2293     /objdef ~ #1
2294     /type
2295     \str_case:nn {#2}
2296     {
2297       { array } { /array }
2298       { dict } { /dict }
2299       { fstream } { /stream }
2300       { stream } { /stream }
2301     }
2302     /OBJ
2303   }
2304   \use:c { __pdf_backend_object_write_ #2 :nn } {#1} {#3}
2305 }
2306 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2307 {
2308   \__pdf_backend_pdfmark:e
2309   { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2310 }
2311 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2312 {

```

```

2313     \__pdf_backend_pdfmark:e
2314     { #1 << \exp_not:n {#2} >> /PUT }
2315 }
2316 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2317 {
2318     \exp_args:Ne
2319     \__pdf_backend_object_write_fstream:nnn {#1} #2
2320 }
2321 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2322 {
2323     \__kernel_backend_postscript:n
2324     {
2325         SDict ~ begin ~
2326         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2327         mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2328         end
2329     }
2330 }
2331 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2332 {
2333     \exp_args:Ne
2334     \__pdf_backend_object_write_stream:nnn {#1} #2
2335 }
2336 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2337 {
2338     \__kernel_backend_postscript:n
2339     {
2340         mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2341         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2342     }
2343 }

```

(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
2344 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2345 {
2346     \int_gincr:N \g__pdf_backend_object_int
2347     \__pdf_backend_object_write_aux:nnn
2348     { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2349     {#1} {#2}
2350 }
2351 \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.

```

2352 \cs_new:Npn \__pdf_backend_object_last:
2353 { { 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.

```

2354 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2355 { { Page #1 } }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.1.3 Destinations

`_pdf_backend_destination:nn`
`_pdf_backend_destination:nnnn`
`_pdf_backend_destination_aux:nnnn`

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

```

2356 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2357 {
2358   \__kernel_backend_postscript:n { pdf.dest.anchor }
2359   \_pdf_backend_pdfmark:e
2360   {
2361     /View
2362     [
2363       \str_case:nnF {#2}
2364       {
2365         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2366         { fit } { /Fit }
2367         { fitb } { /FitB }
2368         { fitbh } { /FitBH ~ pdf.dest.y }
2369         { fitbv } { /FitBV ~ pdf.dest.x }
2370         { fith } { /FitH ~ pdf.dest.y }
2371         { fitv } { /FitV ~ pdf.dest.x }
2372         { fitr } { /Fit }
2373       }
2374       {
2375         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2376       }
2377     ]
2378     /Dest ( \exp_not:n {#1} ) cvn
2379     /DEST
2380   }
2381 }
2382 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2383 {
2384   \exp_args:Ne \_pdf_backend_destination_aux:nnnn
2385   { \dim_eval:n {#2} } {#1} {#3} {#4}
2386 }
2387 \cs_new_protected:Npn \_pdf_backend_destination_aux:nnnn #1#2#3#4
2388 {
2389   \vbox_to_zero:n
2390   {
2391     \__kernel_kern:n {#4}
2392     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2393     \tex_vss:D
2394   }
2395   \__kernel_kern:n {#1}
2396   \vbox_to_zero:n
2397   {
2398     \__kernel_kern:n { -#3 }
2399     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2400     \tex_vss:D
2401   }
2402   \__kernel_kern:n { -#1 }
2403   \_pdf_backend_pdfmark:n

```

```

2404     {
2405         /View
2406         [
2407             /FitR ~
2408             pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2409             pdf.urx ~ pdf.ury ~ pdf.dest2device
2410         ]
2411         /Dest ( #2 ) cvn
2412         /DEST
2413     }
2414 }

```

(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`

```

2415 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2416 {
2417     \int_compare:nNtT {#1} = 0
2418     {
2419         \__kernel_backend_literal_postscript:n
2420         {
2421             /setdistillerparams ~ where
2422             { pop << /CompressPages ~ false >> setdistillerparams }
2423             if
2424         }
2425     }
2426 }
2427 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2428 {
2429     \bool_if:nF {#1}
2430     {
2431         \__kernel_backend_literal_postscript:n
2432         {
2433             /setdistillerparams ~ where
2434             { pop << /CompressStreams ~ false >> setdistillerparams }
2435             if
2436         }
2437     }
2438 }

```

(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`

```

2439 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2440 {
2441     \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2442 }
2443 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2444 {
2445     \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2446 }

```

(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:` 2447 `\cs_new:Npn __pdf_backend_version_major: { -1 }`
 2448 `\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:` 2449 `\cs_new_protected:Npn __pdf_backend_bdc:nn #1#2`
 2450 `{ __pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }`
 2451 `\cs_new_protected:Npn __pdf_backend_emc:`
 2452 `{ __pdf_backend_pdfmark:n { /EMC } }`

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

2453 `</dvips>`

6.2 LuaTeX and pdfTeX backend

2454 `<*luatex | pdftex>`

6.2.1 Destinations

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

2455 `\cs_new_protected:Npn __pdf_backend_destination:nn #1#2`
 2456 `{`
 2457 `<*luatex>`
 2458 `\tex_pdfextension:D dest ~`
 2459 `</luatex>`
 2460 `<*pdftex>`
 2461 `\tex_pdfdest:D`
 2462 `</pdftex>`
 2463 `name {#1}`
 2464 `\str_case:nnF {#2}`
 2465 `{`
 2466 `{ xyz } { xyz }`
 2467 `{ fit } { fit }`
 2468 `{ fitb } { fitb }`
 2469 `{ fitbh } { fitbh }`
 2470 `{ fitbv } { fitbv }`
 2471 `{ fith } { fith }`
 2472 `{ fitv } { fitv }`
 2473 `{ fitr } { fitr }`
 2474 `}`
 2475 `{ xyz ~ zoom \fp_eval:n { #2 * 10 } }`
 2476 `\scan_stop:`
 2477 `}`
 2478 `\cs_new_protected:Npn __pdf_backend_destination:nnnn #1#2#3#4`
 2479 `{`

```

2480 <*luatex>
2481   \tex_pdfextension:D dest ~
2482 </luatex>
2483 <*pdftex>
2484   \tex_pdfdest:D
2485 </pdftex>
2486   name {#1}
2487   fitr ~
2488   width \dim_eval:n {#2} ~
2489   height \dim_eval:n {#3} ~
2490   depth \dim_eval:n {#4} \scan_stop:
2491 }

```

(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
2492 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2493 {
2494 <*luatex>
2495   \tex_pdfextension:D catalog
2496 </luatex>
2497 <*pdftex>
2498   \tex_pdfcatalog:D
2499 </pdftex>
2500   { / #1 ~ #2 }
2501 }
2502 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2503 {
2504 <*luatex>
2505   \tex_pdfextension:D info
2506 </luatex>
2507 <*pdftex>
2508   \tex_pdfinfo:D
2509 </pdftex>
2510   { / #1 ~ #2 }
2511 }

```

(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.

```

2512 \prop_new:N \g__pdf_backend_object_prop

```

(End of definition for \g__pdf_backend_object_prop.)

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

```

\__pdf_backend_object_ref:n
\__pdf_backend_object_id:n
2513 \cs_new_protected:Npn \__pdf_backend_object_new:
2514 {
2515 <*luatex>
2516   \tex_pdfextension:D obj ~
2517 </luatex>

```

```

2518 <*pdfTeX>
2519   \tex_pdfobj:D
2520 </pdfTeX>
2521   reserveobjnum ~
2522   \int_gset:Nn \g__pdf_backend_object_int
2523 <*luaTeX>
2524   { \tex_pdffeedback:D lastobj }
2525 </luaTeX>
2526 <*pdfTeX>
2527   { \tex_pdflastobj:D }
2528 </pdfTeX>
2529 }
2530 \cs_new:Npn \__pdf_backend_object_ref:n #1 { #1 ~ 0 ~ R }
2531 \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 2532 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
\__pdf_backend_object_write:nn 2533 {
\__pdf_exp_not_i:nn 2534 <*luaTeX>
\__pdf_exp_not_ii:nn 2535   \tex_immediate:D \tex_pdfextension:D obj ~
2536 </luaTeX>
2537 <*pdfTeX>
2538   \tex_immediate:D \tex_pdfobj:D
2539 </pdfTeX>
2540   useobjnum ~ #1
2541   \__pdf_backend_object_write:nn {#2} {#3}
2542 }
2543 \cs_new:Npn \__pdf_backend_object_write:nn #1#2
2544 {
2545   \str_case:nn {#1}
2546   {
2547     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2548     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2549     { fstream }
2550     {
2551       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2552       file ~ { \__pdf_exp_not_ii:nn #2 }
2553     }
2554     { stream }
2555     {
2556       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2557       { \__pdf_exp_not_ii:nn #2 }
2558     }
2559   }
2560 }
2561 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2562 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2563 \cs_new:Npn \__pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

```

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

__pdf_backend_object_now:nn Much like writing, but direct creation.
 __pdf_backend_object_now:ne

```

2564 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2565 {
2566   \*luatex
2567     \tex_immediate:D \tex_pdfextension:D obj ~
2568   \*luatex
2569   \*pdfTeX
2570     \tex_immediate:D \tex_pdfobj:D
2571   \*pdfTeX
2572     \__pdf_backend_object_write:nn {#1} {#2}
2573 }
2574 \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.

```

2575 \cs_new:Npe \__pdf_backend_object_last:
2576 {
2577   \exp_not:N \int_value:w
2578 \*luatex
2579   \exp_not:N \tex_pdffeedback:D lastobj ~
2580 \*luatex
2581 \*pdfTeX
2582   \exp_not:N \tex_pdflastobj:D
2583 \*pdfTeX
2584   \c_space_tl 0 ~ R
2585 }

```

(End of definition for __pdf_backend_object_last:.)

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

```

2586 \cs_new:Npe \_pdf_backend_pageobject_ref:n #1
2587 {
2588   \exp_not:N \int_value:w
2589 \*luatex
2590   \exp_not:N \tex_pdffeedback:D pageref
2591 \*luatex
2592 \*pdfTeX
2593   \exp_not:N \tex_pdfpageref:D
2594 \*pdfTeX
2595   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2596 }

```

(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 2597 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
\_pdf_backend_objcompresslevel:n 2598 {
2599   \tex_global:D
2600 \*luatex
2601   \tex_pdfvariable:D compresslevel
2602 \*luatex
2603 \*pdfTeX
2604   \tex_pdfcompresslevel:D

```

```

2605 </pdftex>
2606     \int_value:w \int_eval:n {#1} \scan_stop:
2607 }
2608 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2609 {
2610     \bool_if:nTF {#1}
2611     { \__pdf_backend_objcompresslevel:n { 2 } }
2612     { \__pdf_backend_objcompresslevel:n { 0 } }
2613 }
2614 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2615 {
2616     \tex_global:D
2617     <*luatex>
2618     \tex_pdfvariable:D objcompresslevel
2619 </luatex>
2620 <*pdftex>
2621     \tex_pdfobjcompresslevel:D
2622 </pdftex>
2623     #1 \scan_stop:
2624 }

```

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

__pdf_backend_version_major_gset:n
__pdf_backend_version_minor_gset:n

The availability of the primitive is not universal, so we have to test at load time.

```

2625 \cs_new_protected:Npe \__pdf_backend_version_major_gset:n #1
2626 {
2627 <*luatex>
2628     \int_compare:nNnT \tex_luatexversion:D > { 106 }
2629     {
2630         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2631         \exp_not:N \int_eval:n {#1} \scan_stop:
2632     }
2633 </luatex>
2634 <*pdftex>
2635     \cs_if_exist:NT \tex_pdfmajorversion:D
2636     {
2637         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2638         \exp_not:N \int_eval:n {#1} \scan_stop:
2639     }
2640 </pdftex>
2641 }
2642 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2643 {
2644     \tex_global:D
2645 <*luatex>
2646     \tex_pdfvariable:D minorversion
2647 </luatex>
2648 <*pdftex>
2649     \tex_pdfminorversion:D
2650 </pdftex>
2651     \int_eval:n {#1} \scan_stop:
2652 }

```

(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:
2653 \cs_new:Npe \__pdf_backend_version_major:
2654 {
2655   \*luatex
2656     \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2657     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2658     { 1 }
2659   \*pdfTeX
2660     \cs_if_exist:NTF \tex_pdfmajorversion:D
2661     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2662     { 1 }
2663   \*pdfTeX
2664 }
2665 \cs_new:Npn \__pdf_backend_version_minor:
2666 {
2667   \tex_the:D
2668   \*luatex
2669     \tex_pdfvariable:D minorversion
2670   \*pdfTeX
2671     \tex_pdfminorversion:D
2672   \*pdfTeX
2673   \tex_pdfminorversion:D
2674 }
2675

```

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

6.2.5 Marked content

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

```

\__pdf_backend_emc:
2676 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2677 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2678 \cs_new_protected:Npn \__pdf_backend_emc:
2679 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2680 \*luatex | pdfTeX

```

6.3 dvipdfmx backend

```

2681 \*dvipdfmx | xetex

```

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

```

\__pdf_backend:e
2682 \cs_new_protected:Npe \__pdf_backend:n #1
2683 { \__kernel_backend_literal:n { pdf: #1 } }
2684 \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
2685 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2686 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2687 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2688 { \__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.

```

2689 \prop_new:N \g__pdf_backend_object_prop

(End of definition for \g__pdf_backend_object_prop.)

```

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

```

\__pdf_backend_object_ref:n
\__pdf_backend_object_id:n
2690 \cs_new_protected:Npn \__pdf_backend_object_new:
2691 { \int_gincr:N \g__pdf_backend_object_int }
2692 \cs_new:Npn \__pdf_backend_object_ref:n #1 { @pdf.obj #1 }
2693 \cs_new_eq:NN \__pdf_backend_object_id:n \__pdf_backend_object_ref:n

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

```

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

```

\__pdf_backend_object_write: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
2694 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2695 {
2696   \use:c { __pdf_backend_object_write_ #2 :nn }
2697   { \__pdf_backend_object_ref:n {#1} } {#3}
2698 }
2699 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2700 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2701 {
2702   \__pdf_backend:e
2703   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2704 }
2705 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2706 {
2707   \__pdf_backend:e
2708   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2709 }
2710 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2711 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2712 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2713 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2714 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2715 {
2716   \__pdf_backend:e
2717   {
2718     #1 stream ~ #2 ~
2719     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2720   }
2721 }

```

(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.

```

2722 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2723 {
2724   \int_gincr:N \g__pdf_backend_object_int
2725   \exp_args:Nne \use:c { __pdf_backend_object_write_ #1 :nn }
2726   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2727   {#2}
2728 }
2729 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

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

`__pdf_backend_object_last:`

```

2730 \cs_new:Npn \__pdf_backend_object_last:
2731 { @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.

```

2732 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2733 { @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
`__pdf_backend_destination:nnnn` Grahn: the idea is to avoid needing to do any calculations in T_EX by using the backend
`__pdf_backend_destination_aux:nnnn` data for @xpos and @ypos. /FitR without rule spec doesn't work, so it falls back to /Fit
here.

```

2734 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2735 {
2736   \__pdf_backend:e
2737   {
2738     dest ~ ( \exp_not:n {#1} )
2739     [
2740       @thispage
2741       \str_case:nnF {#2}
2742       {
2743         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2744         { fit } { /Fit }
2745         { fitb } { /FitB }
2746         { fitbh } { /FitBH }
2747         { fitbv } { /FitBV ~ @xpos }
2748         { fith } { /FitH ~ @ypos }
2749         { fitv } { /FitV ~ @xpos }
2750         { fitr } { /Fit }
2751       }
2752       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2753     ]
2754   }
2755 }

```

```

2756 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2757 {
2758   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2759   { \dim_eval:n {#2} } {#1} {#3} {#4}
2760 }
2761 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2762 {
2763   \vbox_to_zero:n
2764   {
2765     \__kernel_kern:n {#4}
2766     \hbox:n
2767     {
2768       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2769       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2770     }
2771     \tex_vss:D
2772   }
2773   \__kernel_kern:n {#1}
2774   \vbox_to_zero:n
2775   {
2776     \__kernel_kern:n { -#3 }
2777     \hbox:n
2778     {
2779       \__pdf_backend:n
2780       {
2781         dest ~ (#2)
2782         [
2783           @thispage
2784           /FitR ~
2785           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2786           @xpos ~ @ypos
2787         ]
2788       }
2789     }
2790     \tex_vss:D
2791   }
2792   \__kernel_kern:n { -#1 }
2793 }

```

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

```

2794 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2795 { \__kernel_backend_literal:e { dvipdfmx:config~z~ \int_eval:n {#1} } }
2796 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2797 {
2798   \bool_if:nF {#1}
2799   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2800 }

```

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

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

_pdf_backend_version_minor_gset:n

```

2801 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2802 {
2803   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2804   \__kernel_backend_literal:e { pdf:majorversion~ \__pdf_backend_version_major: }
2805 }
2806 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2807 {
2808   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2809   \__kernel_backend_literal:e { pdf:minorversion~ \__pdf_backend_version_minor: }
2810 }

```

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

_pdf_backend_version_major: We start with the assumption that the default is active.

_pdf_backend_version_minor:

```

2811 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2812 \cs_new:Npn \__pdf_backend_version_minor: { 7 }

```

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

6.3.5 Marked content

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

_pdf_backend_emc:

```

2813 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2814 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2815 \cs_new_protected:Npn \__pdf_backend_emc:
2816 { \__kernel_backend_literal_page:n { EMC } }

```

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

2817 </dviPDFmx | xetex>

6.4 dvisvgm backend

2818 <*dvisvgm>

6.4.1 Destinations

_pdf_backend_destination:nn

_pdf_backend_destination:nnnn

```

2819 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2 { }
2820 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4 { }

```

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

6.4.2 Catalogue entries

_pdf_backend_catalog_gput:nn No-op.

__pdf_backend_info_gput:nn

```

2821 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2 { }
2822 \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

All no-ops here.

```

\__pdf_backend_object_new:
\__pdf_backend_object_ref:n
\__pdf_backend_object_id:n
    \__pdf_backend_object_write:nnn
    \__pdf_backend_object_write:ne
\__pdf_backend_object_now:nn
\__pdf_backend_object_now:ne
\__pdf_backend_object_last:
    \__pdf_backend_pageobject_ref:n
2823 \cs_new_protected:Npn \__pdf_backend_object_new: { }
2824 \cs_new:Npn \__pdf_backend_object_ref:n #1 { }
2825 \cs_new:Npn \__pdf_backend_object_id:n #1 { }
2826 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3 { }
2827 \cs_new_protected:Npn \__pdf_backend_object_write:nne #1#2#3 { }
2828 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2 { }
2829 \cs_new_protected:Npn \__pdf_backend_object_now:ne #1#2 { }
2830 \cs_new:Npn \__pdf_backend_object_last: { }
2831 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1 { }

```

(End of definition for __pdf_backend_object_new: and others.)

6.4.4 Structure

These are all no-ops.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2832 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2833 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1 { }

```

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

Data not available!

```

\__pdf_backend_version_major_gset:n
\__pdf_backend_version_minor_gset:n
2834 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2835 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1 { }

```

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

Data not available!

```

\__pdf_backend_version_major:
\__pdf_backend_version_minor:
2836 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2837 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

```

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

More no-ops.

```

\__pdf_backend_bdc:nn
\__pdf_backend_emc:
2838 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2839 \cs_new_protected:Npn \__pdf_backend_emc: { }

```

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

2840 \langle /dvisvgm \rangle

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.

2841 \langle *dvi_{pdf}mx | dvips \rangle

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

```

2842 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
2843 {
2844   \_kernel_backend_first_shipout:n
2845   {
2846     \_kernel_backend_literal:e
2847     {
2848       <*dviPDFmx>
2849         pdf:pagesize ~
2850         width ~ \dim_eval:n {#1} ~
2851         height ~ \dim_eval:n {#2}
2852       </dviPDFmx>
2853       <*dvips>
2854         papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
2855       </dvips>
2856     }
2857   }
2858 }

(End of definition for \_pdf_backend_pagesize_gset:nn.)

2859 </dviPDFmx | dvips>
2860 <*luatex | pdftex | xetex>

```

`_pdf_backend_pagesize_gset:nn` Pass to the primitives.

```

2861 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
2862 {
2863   \dim_gset:Nn \tex_pagewidth:D {#1}
2864   \dim_gset:Nn \tex_pageheight:D {#2}
2865 }

(End of definition for \_pdf_backend_pagesize_gset:nn.)

2866 </luatex | pdftex | xetex>
2867 <*dvipsgm>

```

`_pdf_backend_pagesize_gset:nn` A no-op.

```

2868 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2 { }

(End of definition for \_pdf_backend_pagesize_gset:nn.)

2869 </dvipsgm>
2870 </package>

```

7 l3backend-pdfannot implementation

```

2871 <*package>
2872 <@@=pdfannot>

```

7.1 dvips backend

```

2873 <*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.

2874 `\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.

2875 `\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.

2876 `\int_new:N \g__pdfannot_backend_int`

(End of definition for `\g__pdfannot_backend_int`.)

`__pdfannot_backend_generic:nnnn` Annotations are objects but they are not in the object data lists. Here, to get the
`__pdfannot_backend_generic_aux:nnnn` 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.

```
2877 \cs_new_protected:Npn \__pdfannot_backend_generic:nnnn #1#2#3#4
2878 {
2879   \exp_args:Nf \__pdfannot_backend_generic_aux:nnnn
2880   { \dim_eval:n {#1} } {#2} {#3} {#4}
2881 }
2882 \cs_new_protected:Npn \__pdfannot_backend_generic_aux:nnnn #1#2#3#4
2883 {
2884   \box_move_down:nn {#3}
2885   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2886   \box_move_up:nn {#2}
2887   {
2888     \hbox:n
2889     {
2890       \__kernel_kern:n {#1}
2891       \__kernel_backend_postscript:n { pdf.save.ur }
2892       \__kernel_kern:n { -#1 }
2893     }
2894   }
2895   \int_gincr:N \g__pdfannot_backend_int
2896   \__kernel_backend_postscript:e
2897   {
2898     mark
2899     /_objdef { pdf.annot \int_use:N \g__pdfannot_backend_int }
2900     pdf.rect
2901     #4 ~
2902     /ANN ~
2903     pdfmark
2904   }
2905 }
```

(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.

```
2906 \cs_new:Npn \__pdfannot_backend_last:
2907 { { 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.

```
2908 \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.

```
2909 \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.

```
2910 \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.

```
2911 \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.

```
2912 \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.

```
2913 \tl_new:N \l__pdfannot_backend_breaklink_pdfmark_tl
2914 \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.

```
2915 \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.

```
2916 \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.

```

2917 \cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nnw #1#2
2918 {
2919   \_pdfannot_backend_link_begin:nw
2920   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2921 }
2922 \cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nnw #1#2
2923 { \_pdfannot_backend_link_begin:nw {#1#2} }
2924 \cs_new_protected:Npn \_pdfannot_backend_link_begin:nw #1
2925 {
2926   \bool_if:NF \g__pdfannot_backend_link_bool
2927   { \_pdfannot_backend_link_begin_aux:nw {#1} }
2928 }

```

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.

```

2929 \cs_new_protected:Npn \_pdfannot_backend_link_begin_aux:nw #1
2930 {
2931   \bool_gset_true:N \g__pdfannot_backend_link_bool
2932   \_kernel_backend_postscript:n
2933   { /pdf.link.dict ( #1 ) def }
2934   \tl_gset:Nn \g__pdfannot_backend_link_dict_tl {#1}
2935   \_pdfannot_backend_link_sf_save:
2936   \mode_if_math:TF
2937   { \bool_gset_true:N \g__pdfannot_backend_link_math_bool }
2938   { \bool_gset_false:N \g__pdfannot_backend_link_math_bool }
2939   \hbox_set:Nw \l__pdfannot_backend_content_box
2940   \_pdfannot_backend_link_sf_restore:
2941   \bool_if:NT \g__pdfannot_backend_link_math_bool
2942   { \c_math_toggle_token }
2943 }
2944 \cs_new_protected:Npn \_pdfannot_backend_link_end:
2945 {
2946   \bool_if:NT \g__pdfannot_backend_link_bool
2947   { \_pdfannot_backend_link_end_aux: }
2948 }
2949 \cs_new_protected:Npn \_pdfannot_backend_link_end_aux:

```

```

2950 {
2951   \bool_if:NT \g__pdfannot_backend_link_math_bool
2952   { \c_math_toggle_token }
2953   \__pdfannot_backend_link_sf_save:
2954   \hbox_set_end:
2955   \__pdfannot_backend_link_minima:
2956   \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
2957   \exp_args:Ne \__pdfannot_backend_link_outerbox:n
2958   {
2959     \int_if_odd:nTF { \value { page } }
2960     { \oddsidemargin }
2961     { \evensidemargin }
2962   }
2963   \box_move_down:nn { \box_dp:N \l__pdfannot_backend_content_box }
2964   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2965   \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.begin }
2966   \__pdfannot_backend_breaklink_usebox:N \l__pdfannot_backend_content_box
2967   \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.end }
2968   \box_move_up:nn { \box_ht:N \l__pdfannot_backend_content_box }
2969   {
2970     \hbox:n
2971     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2972   }
2973   \int_gincr:N \g__pdfannot_backend_int
2974   \int_gset_eq:NN \g__pdfannot_backend_link_int \g__pdfannot_backend_int
2975   \__kernel_backend_postscript:e
2976   {
2977     mark
2978     /_objdef { pdf.annot \int_use:N \g__pdfannot_backend_link_int }
2979     \g__pdfannot_backend_link_dict_tl \c_space_tl
2980     pdf.rect
2981     /ANN ~ \l__pdfannot_backend_breaklink_pdfmark_tl
2982   }
2983   \__pdfannot_backend_link_sf_restore:
2984   \bool_gset_false:N \g__pdfannot_backend_link_bool
2985   }
2986   \cs_new_protected:Npn \__pdfannot_backend_link_minima:
2987   {
2988     \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
2989     \__kernel_backend_postscript:e
2990     {
2991       /pdf.linkdp.pad ~
2992       \dim_to_decimal:n
2993       {
2994         \dim_max:nn
2995         {
2996           \box_dp:N \l__pdfannot_backend_model_box
2997           - \box_dp:N \l__pdfannot_backend_content_box
2998         }
2999         { Opt }
3000       } ~
3001       pdf.pt.dvi ~ def
3002       /pdf.linkht.pad ~
3003       \dim_to_decimal:n

```

```

3004         {
3005             \dim_max:nn
3006             {
3007                 \box_ht:N \l__pdfannot_backend_model_box
3008                 - \box_ht:N \l__pdfannot_backend_content_box
3009             }
3010             { Opt }
3011         } ~
3012         pdf.pt.dvi ~ def
3013     }
3014 }
3015 \cs_new_protected:Npn \__pdfannot_backend_link_outerbox:n #1
3016 {
3017     \__kernel_backend_postscript:e
3018     {
3019         /pdf.outerbox
3020         [
3021             \dim_to_decimal:n {#1} ~
3022             \dim_to_decimal:n { -\box_dp:N \l__pdfannot_backend_model_box } ~
3023             \dim_to_decimal:n { #1 + \textwidth } ~
3024             \dim_to_decimal:n { \box_ht:N \l__pdfannot_backend_model_box }
3025         ]
3026         [ exch { pdf.pt.dvi } forall ] def
3027     /pdf.baselineskip ~
3028     \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
3029     { pdf.pt.dvi ~ def }
3030     { pop ~ pop }
3031     ifelse
3032 }
3033 }
3034 \cs_new_protected:Npn \__pdfannot_backend_link_sf_save:
3035 {
3036     \int_gset:Nn \g__pdfannot_backend_link_sf_int
3037     {
3038         \mode_if_horizontal:TF
3039         { \tex_spacefactor:D }
3040         { 0 }
3041     }
3042 }
3043 \cs_new_protected:Npn \__pdfannot_backend_link_sf_restore:
3044 {
3045     \mode_if_horizontal:T
3046     {
3047         \int_compare:nNnT \g__pdfannot_backend_link_sf_int > { 0 }
3048         { \int_set:Nn \tex_spacefactor:D \g__pdfannot_backend_link_sf_int }
3049     }
3050 }

```

(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.

```

3051 \use_none:nnn
3052 \cs_if_exist:NT \hook_gput_code:nnn
3053 {

```

```

3054 \hook_gput_code:nnn { build/column/after } { backend }
3055 {
3056   \box_if_empty:NF \l_shipout_box
3057   {
3058     \vbox_set:Nn \l_shipout_box
3059     {
3060       \__kernel_backend_postscript:n
3061       {
3062         pdf.globaldict /pdf.brokenlink.rect ~ known
3063         { pdf.bordertracking.continue }
3064         if
3065       }
3066       \vbox_unpack_drop:N \l_shipout_box
3067       \__kernel_backend_postscript:n
3068       { pdf.bordertracking.endpage }
3069     }
3070   }
3071 }
3072 \tl_set:Nn \l__pdfannot_backend_breaklink_pdfmark_tl { pdf.pdfmark }
3073 \cs_set_eq:NN \__pdfannot_backend_breaklink_postscript:n
3074 \__kernel_backend_postscript:n
3075 \cs_set_eq:NN \__pdfannot_backend_breaklink_usebox:N \hbox_unpack:N
3076 }

```

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

```

3077 \cs_new:Npn \__pdfannot_backend_link_last:
3078 { { 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.

```

3079 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1
3080 {
3081   \__kernel_backend_postscript:e
3082   {
3083     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
3084   }
3085 }

```

(End of definition for __pdfannot_backend_link_margin:n.)

_pdfannot_backend_link_on:

```

\__pdfannot_backend_link_off:
3086 \cs_new_protected:Npn \__pdfannot_backend_link_on: { }
3087 \cs_new_protected:Npn \__pdfannot_backend_link_off: { }

```

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

```

3088 </dvips>

```

7.2 LuaTeX and pdfTeX backend

3089 `<*luatex | pdftex>`

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

```
3090 \cs_new_protected:Npn \_pdfannot_backend_generic:nnnn #1#2#3#4
3091 {
3092   <*luatex>
3093     \tex_pdfextension:D annot ~
3094   </luatex>
3095   <*pdftex>
3096     \tex_pdfannot:D
3097   </pdftex>
3098     width ~ \dim_eval:n {#1} ~
3099     height ~ \dim_eval:n {#2} ~
3100     depth ~ \dim_eval:n {#3} ~
3101     {#4}
3102 }
```

(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.

```
3103 \cs_new:Npe \_pdfannot_backend_last:
3104 {
3105   \exp_not:N \int_value:w
3106   <*luatex>
3107     \exp_not:N \tex_pdffeedback:D lastannot ~
3108   </luatex>
3109   <*pdftex>
3110     \exp_not:N \tex_pdflastannot:D
3111   </pdftex>
3112   \c_space_tl 0 ~ R
3113 }
```

(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:
3114 \cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nnw #1#2
3115 { \_pdfannot_backend_link_begin:nnnw {#1} { goto~name } {#2} }
3116 \cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nnw #1#2
3117 { \_pdfannot_backend_link_begin:nnnw {#1} { user } {#2} }
3118 \cs_new_protected:Npn \_pdfannot_backend_link_begin:nnnw #1#2#3
3119 {
3120   <*luatex>
3121     \tex_pdfextension:D startlink ~
3122   </luatex>
3123   <*pdftex>
3124     \tex_pdfstartlink:D
3125   </pdftex>
3126     attr {#1}
3127     #2 {#3}
3128   }
3129 \cs_new_protected:Npn \_pdfannot_backend_link_end:
```

```

3130 {
3131 <*luatex>
3132 \tex_pdfextension:D endlink \scan_stop:
3133 </luatex>
3134 <*pdftex>
3135 \tex_pdfendlink:D
3136 </pdftex>
3137 }

```

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

_pdfannot_backend_link_last: Formatted for direct use.

```

3138 \cs_new:Npe \_pdfannot_backend_link_last:
3139 {
3140 \exp_not:N \int_value:w
3141 <*luatex>
3142 \exp_not:N \tex_pdffeedback:D lastlink ~
3143 </luatex>
3144 <*pdftex>
3145 \exp_not:N \tex_pdflastlink:D
3146 </pdftex>
3147 \c_space_tl 0 ~ R
3148 }

```

(End of definition for _pdfannot_backend_link_last:.)

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

```

3149 \cs_new_protected:Npn \_pdfannot_backend_link_margin:n #1
3150 {
3151 <*luatex>
3152 \tex_pdfvariable:D linkmargin
3153 </luatex>
3154 <*pdftex>
3155 \tex_pdflinkmargin:D
3156 </pdftex>
3157 \dim_eval:n {#1} \scan_stop:
3158 }

```

(End of definition for _pdfannot_backend_link_margin:n.)

_pdfannot_backend_link_on: Separate definitions for the two engines.

```

\_pdfannot_backend_link_off: 3159 \cs_new_protected:Npn \_pdfannot_backend_link_on:
3160 <*luatex>
3161 { \tex_pdfextension:D linkstate 0 ~ }
3162 </luatex>
3163 <*pdftex>
3164 { \tex_pdfrunninglinkon:D }
3165 </pdftex>
3166 \cs_new_protected:Npn \_pdfannot_backend_link_off:
3167 <*luatex>
3168 { \tex_pdfextension:D linkstate 1 ~ }
3169 </luatex>
3170 <*pdftex>
3171 { \tex_pdfrunninglinkoff:D }
3172 </pdftex>

```

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

3173 `</luatex | pdftex>`

7.3 dvipdfmx backend

3174 `<*dvipdfmx | xetex>`

`_pdfannot_backend:n`
`_pdfannot_backend:e`

A generic function for the backend PDF specials

3175 `\cs_new_protected:Npe _pdfannot_backend:n #1`
3176 `{ _kernel_backend_literal:n { pdf: #1 } }`
3177 `\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.

3178 `\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.

3179 `\cs_new_protected:Npn _pdfannot_backend_generic:nnnn #1#2#3#4`
3180 `{`
3181 `\int_gincr:N \g_pdfannot_backend_int`
3182 `_pdfannot_backend:e`
3183 `{`
3184 `ann ~ @pdfannot \int_use:N \g_pdfannot_backend_int \c_space_tl`
3185 `width ~ \dim_eval:n {#1} ~`
3186 `height ~ \dim_eval:n {#2} ~`
3187 `depth ~ \dim_eval:n {#3} ~`
3188 `<< /Type /Annot #4 >>`
3189 `}`
3190 `}`

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

`_pdfannot_backend_last:`

3191 `\cs_new:Npn _pdfannot_backend_last:`
3192 `{ @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.

3193 `\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:`

3194 `\cs_new_protected:Npn _pdfannot_backend_link_begin_goto:nnw #1#2`
3195 `{`
3196 `_pdfannot_backend_link_begin:n`
3197 `{ #1 /Subtype /Link /A << /S /GoTo /D (#2) >> }`
3198 `}`
3199 `\cs_new_protected:Npn _pdfannot_backend_link_begin_user:nnw #1#2`
3200 `{ _pdfannot_backend_link_begin:n {#1#2} }`
3201 `\cs_new_protected:Npe _pdfannot_backend_link_begin:n #1`

```

3202 {
3203   \int_gincr:N \exp_not:N \g__pdfannot_backend_int
3204   \int_gset_eq:NN \exp_not:N \g__pdfannot_backend_link_int
3205   \exp_not:N \g__pdfannot_backend_int
3206   \__pdfannot_backend:e
3207   {
3208     bann ~
3209     @pdfannot
3210     \exp_not:N \int_use:N \exp_not:N \g__pdfannot_backend_link_int
3211     \c_space_tl
3212     <<
3213     /Type /Annot
3214     #1
3215     >>
3216   }
3217 }
3218 \cs_new_protected:Npn \__pdfannot_backend_link_end:
3219 { \__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.

```

3220 \cs_new:Npn \__pdfannot_backend_link_last:
3221 { @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.

```

3222 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1
3223 { \__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:

```

3224 \cs_new_protected:Npn \__pdfannot_backend_link_on: { \__pdfannot_backend:n { link } }
3225 \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:.)

```

3226 </dvipdfmx | xetex>

```

7.4 dvisvgm backend

```

3227 <{*dvisvgm}

```

_pdfannot_backend_generic:nnnn

```

3228 \cs_new_protected:Npn \__pdfannot_backend_generic:nnnn #1#2#3#4 { }

```

(End of definition for __pdfannot_backend_generic:nnnn.)

__pdfannot_backend_last:

```

3229 \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:
3230 \cs_new_protected:Npn \__pdfannot_backend_link_begin_goto:nnw #1#2 { }
3231 \cs_new_protected:Npn \__pdfannot_backend_link_begin_user:nnw #1#2 { }
3232 \cs_new_protected:Npn \__pdfannot_backend_link_begin:nnnw #1#2#3 { }
3233 \cs_new_protected:Npn \__pdfannot_backend_link_end: { }

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

\__pdfannot_backend_link_last:
3234 \cs_new:Npe \__pdfannot_backend_link_last: { }

(End of definition for \__pdfannot_backend_link_last:.)

\__pdfannot_backend_link_margin:n
3235 \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:
3236 \cs_new_protected:Npn \__pdfannot_backend_link_on: { }
3237 \cs_new_protected:Npn \__pdfannot_backend_link_off: { }

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

3238 </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.

```

3239 <@@=)

3240 \cs_new_eq:NN \__pdf_backend_annotation:nnnn \__pdfannot_backend_generic:nnnn
3241 \cs_new_eq:NN \__pdf_backend_annotation_last: \__pdfannot_backend_last:
3242 \clist_map_inline:nn
3243 {
3244   begin_goto:nnw ,
3245   begin_user:nnw ,
3246   begin:nnnw ,
3247   end: ,
3248   last: ,
3249   margin:n
3250 }
3251 { \cs_new_eq:cc { __pdf_backend_link_ #1 } { __pdfannot_backend_link_ #1 } }
3252 </package>

```

8 l3backend-opacity implementation

```
3253 <*package>
3254 <@@=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.

```
3255 <*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_fill:
  __opacity_backend_reset_stroke:
3256 \cs_new_protected:Npn __opacity_backend_select:n #1
3257 {
3258   __opacity_backend:nnn {#1} { fill } { ca }
3259   __opacity_backend:nnn {#1} { stroke } { CA }
3260   \group_insert_after:N __opacity_backend_reset_fill:
3261   \group_insert_after:N __opacity_backend_reset_stroke:
3262 }
3263 \cs_new_protected:Npn __opacity_backend_fill:n #1
3264 {
3265   __opacity_backend:nnn
3266   { #1 }
3267   { fill }
3268   { ca }
3269   \group_insert_after:N __opacity_backend_reset_fill:
3270 }
3271 \cs_new_protected:Npn __opacity_backend_stroke:n #1
3272 {
3273   __opacity_backend:nnn
3274   { #1 }
3275   { stroke }
3276   { CA }
3277   \group_insert_after:N __opacity_backend_reset_stroke:
3278 }
3279 \cs_new_protected:Npn __opacity_backend:nnn #1#2#3
3280 {
3281   __kernel_backend_postscript:n
3282   {
3283     product ~ (Ghostscript) ~ search
3284     {
3285       pop ~ pop ~ pop ~
3286       #1 ~ .set #2 constantalpha
3287     }
3288     {
3289       pop ~
3290       mark ~
3291       /#3 ~ #1
3292       /SetTransparency ~

```

```

3293         pdfmark
3294     }
3295     ifelse
3296 }
3297 }
3298 \cs_new_protected:Npn \__opacity_backend_reset_fill:
3299 {
3300     \__opacity_backend:nnn
3301     { 1 }
3302     { fill }
3303     { ca }
3304 }
3305 \cs_new_protected:Npn \__opacity_backend_reset_stroke:
3306 {
3307     \__opacity_backend:nnn
3308     { 1 }
3309     { stroke }
3310     { CA }
3311 }

```

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

```

3312 </dvips>
3313 <*dvipdfmx | luatex | pdftex | xetex>

```

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

```

3314 \bool_lazy_and:nnT
3315 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3316 { \pdfmanagement_if_active_p: }
3317 {
3318     <*luatex | pdftex>
3319     \__kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3320     { page ~ direct } { /opacity 1 ~ gs }
3321 </luatex | pdftex>
3322     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3323     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3324 }

```

(End of definition for \c__opacity_backend_stack_int.)

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

\l__opacity_backend_stroke_tl

```

3325 \tl_new:N \l__opacity_backend_fill_tl
3326 \tl_new:N \l__opacity_backend_stroke_tl
3327 \tl_set:Nn \l__opacity_backend_fill_tl { 1 }
3328 \tl_set:Nn \l__opacity_backend_stroke_tl { 1 }

```

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

__opacity_backend_select:n Much the same as color.

__opacity_backend_reset:

```

3329 \cs_new_protected:Npn \__opacity_backend_select:n #1
3330 {
3331     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3332     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3333     \pdfmanagement_add:nnn { Page / Resources / ExtGState }

```

```

3334     { opacity #1 }
3335     { << /ca ~ #1 /CA ~ #1 >> }
3336 <*dvipdfmx | xetex>
3337     \__kernel_backend_literal_pdf:n
3338 </dvipdfmx | xetex>
3339 <*luatex | pdftex>
3340     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3341 </luatex | pdftex>
3342     { /opacity #1 ~ gs }
3343     \group_insert_after:N \__opacity_backend_reset:
3344 }
3345 \cs_new_protected:Npn \__opacity_backend_reset:
3346 {
3347 <*dvipdfmx | xetex>
3348     \__kernel_backend_literal_pdf:n
3349     { /opacity1 ~ gs }
3350 </dvipdfmx | xetex>
3351 <*luatex | pdftex>
3352     \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int
3353 </luatex | pdftex>
3354 }

```

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

__opacity_backend_fill:n
__opacity_backend_stroke:n
__opacity_backend_fill_stroke:nn

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

```

3355 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3356 {
3357     \exp_args:Nno \__opacity_backend_fill_stroke:nn
3358     { #1 }
3359     { \l__opacity_backend_stroke_tl }
3360 }
3361 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3362 {
3363     \exp_args:No \__opacity_backend_fill_stroke:nn
3364     { \l__opacity_backend_fill_tl }
3365     { #1 }
3366 }
3367 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3368 {
3369     \str_if_eq:nnTF {#1} {#2}
3370     { \__opacity_backend_select:n {#1} }
3371     {
3372         \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3373         \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3374         \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3375         { opacity.fill #1 }
3376         { << /ca ~ #1 >> }
3377         \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3378         { opacity.stroke #2 }
3379         { << /CA ~ #2 >> }
3380 <*dvipdfmx | xetex>
3381     \__kernel_backend_literal_pdf:n
3382 </dvipdfmx | xetex>

```

```

3383 <*luatex | pdftex>
3384     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3385 </luatex | pdftex>
3386     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3387     \group_insert_after:N \__opacity_backend_reset:
3388 }
3389 }

```

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

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

```

\__opacity_backend_fill_stroke:nn
3390 \bool_lazy_and:nnF
3391 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3392 { \pdfmanagement_if_active_p: }
3393 {
3394     \cs_gset_protected:Npn \__opacity_backend_select:n #1 { }
3395     \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2 { }
3396 }

```

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

```

3397 </dvipdfmx | luatex | pdftex | xetex>
3398 <*dvisvgm>

```

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

```

\__opacity_backend_fill:n
\__opacity_backend_stroke:n
\__opacity_backend:nn
3399 \cs_new_protected:Npn \__opacity_backend_select:n #1
3400 { \__opacity_backend:nn {#1} { } }
3401 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3402 { \__opacity_backend:nn {#1} { fill- } }
3403 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3404 { \__opacity_backend:nn {#1} { stroke- } }
3405 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3406 { \__kernel_backend_scope:e { #2 opacity = " #1 " } }

```

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

```

3407 </dvisvgm>
3408 </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.

```

3409 <*lua>

```

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

```

3410 local pdfmanagement_active do
3411     local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3412     local cmd = pdfmanagement_if_active_p.cmdname
3413     if cmd == 'undefined_cs' then
3414         pdfmanagement_active = false
3415     else
3416         token.put_next(pdfmanagement_if_active_p)

```

```

3417     pdfmanagement_active = token.scan_int() ~= 0
3418   end
3419 end
3420
3421 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3422   luaotfload.set_transparent_colorstack(function() return token.create'c__opacity_backend_st
3423
3424   local transparent_register = {
3425     token.create'pdfmanagement_add:nnn',
3426     token.new(0, 1),
3427     'Page/Resources/ExtGState',
3428     token.new(0, 2),
3429     token.new(0, 1),
3430     '',
3431     token.new(0, 2),
3432     token.new(0, 1),
3433     '<</ca ',
3434     '',
3435     '/CA ',
3436     '',
3437     '>>',
3438     token.new(0, 2),
3439   }
3440   luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)
3441     value = (octet * -1):match(value)
3442     if not value then
3443       tex.error'Invalid transparency value'
3444       return
3445     end
3446     value = value:sub(1, -2)
3447     local result = 'opacity' .. value
3448     tex.runtoks(function()
3449       transparent_register[6], transparent_register[10], transparent_register[12] = result,
3450       tex.sprint(-2, transparent_register)
3451     end)
3452     return '/' .. result .. ' gs'
3453   end, 'l3opacity')
3454 end
3455 </lua>

```

9 l3backend-header implementation

```

3456 <*dvips & header>

```

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

```

3457 /color.sc { } def

```

(End of definition for color.sc.)

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

```

3458 TeXDict begin
3459 /TeXcolorseparation { setcolor } def
3460 end

```

(End of definition for TeXcolorseparation and separation.)

pdf.globaldict A small global dictionary for backend use.

```
3461 true setglobal
3462 /pdf.globaldict 4 dict def
3463 false setglobal
```

(End of definition for pdf.globaldict.)

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

```
pdf.rect.ht 3464 /pdf.cvs { 65534 string cvs } def
3465 /pdf.dvi.pt { 72.27 mul Resolution div } def
3466 /pdf.pt.dvi { 72.27 div Resolution mul } def
3467 /pdf.rect.ht { dup 1 get neg exch 3 get add } def
```

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

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

```
pdf.linkdp.pad 3468 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad 3469 /pdf.linkdp.pad { 0 } def
3470 /pdf.linkht.pad { 0 } def
```

(End of definition for pdf.linkmargin, pdf.linkdp.pad, and pdf.linkht.pad.)

pdf.rect Functions for marking the limits of an annotation/link, plus drawing the border. We
pdf.save.ll separate links for generic annotations to support adding a margin and setting a minimal
pdf.save.ur size.

```
pdf.save.linkll 3471 /pdf.rect
pdf.save.linkur 3472 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.llx 3473 /pdf.save.ll
pdf.lly 3474 {
pdf.urx 3475 currentpoint
pdf.ury 3476 /pdf.lly exch def
3477 /pdf.llx exch def
3478 }
3479 def
3480 /pdf.save.ur
3481 {
3482 currentpoint
3483 /pdf.ury exch def
3484 /pdf.urx exch def
3485 }
3486 def
3487 /pdf.save.linkll
3488 {
3489 currentpoint
3490 pdf.linkmargin add
3491 pdf.linkdp.pad add
3492 /pdf.lly exch def
3493 pdf.linkmargin sub
3494 /pdf.llx exch def
3495 }
```

```

3496     def
3497 /pdf.save.linkur
3498 {
3499     currentpoint
3500     pdf.linkmargin sub
3501     pdf.linkht.pad sub
3502     /pdf.ury exch def
3503     pdf.linkmargin add
3504     /pdf.urx exch def
3505 }
3506     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 3507 /pdf.dest.anchor
pdf.dev.y 3508 {
pdf.tmpa 3509     currentpoint exch
pdf.tmpb 3510     pdf.dvi.pt 72 add
pdf.tmpc 3511     /pdf.dest.x exch def
pdf.tmpd 3512     pdf.dvi.pt
3513     vsize 72 sub exch sub
3514     /pdf.dest.y exch def
3515 }
3516     def
3517 /pdf.dest.point
3518 { pdf.dest.x pdf.dest.y } def
3519 /pdf.dest2device
3520 {
3521     /pdf.dest.y exch def
3522     /pdf.dest.x exch def
3523     matrix currentmatrix
3524     matrix defaultmatrix
3525     matrix invertmatrix
3526     matrix concatmatrix
3527     cvx exec
3528     /pdf.dev.y exch def
3529     /pdf.dev.x exch def
3530     /pdf.tmpd exch def
3531     /pdf.tmpc exch def
3532     /pdf.tmpb exch def
3533     /pdf.tmpa exch def
3534     pdf.dest.x pdf.tmpa mul
3535     pdf.dest.y pdf.tmpc mul add
3536     pdf.dev.x add
3537     pdf.dest.x pdf.tmpb mul
3538     pdf.dest.y pdf.tmpd mul add
3539     pdf.dev.y add
3540 }
3541     def

```

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

```

pdf.bordertracking To know where a breakable link can go, we need to track the boundary rectangle. That
pdf.bordertracking.begin can be done by hooking into a and x operations: those names have to be retained. The
pdf.bordertracking.end boundary is stored at the end of the operation. Special effort is needed at the start and
pdf.leftboundary end of pages (or rather galleys), such that everything works properly.
pdf.rightboundary
pdf.brokenlink.rect 3542 /pdf.bordertracking false def
pdf.brokenlink.skip 3543 /pdf.bordertracking.begin
pdf.brokenlink.dict 3544 {
3545   SDict /pdf.bordertracking true put
3546   SDict /pdf.leftboundary undef
3547   SDict /pdf.rightboundary undef
pdf.bordertracking.endpage 3548   /a where
pdf.bordertracking.continue 3549   {
3550     /a
3551     {
3552       currentpoint pop
3553       SDict /pdf.rightboundary known dup
3554       {
3555         SDict /pdf.rightboundary get 2 index lt
3556         { not }
3557         if
3558       }
3559       if
3560       { pop }
3561       { SDict exch /pdf.rightboundary exch put }
3562       ifelse
3563       moveto
3564       currentpoint pop
3565       SDict /pdf.leftboundary known dup
3566       {
3567         SDict /pdf.leftboundary get 2 index gt
3568         { not }
3569         if
3570       }
3571       if
3572       { pop }
3573       { SDict exch /pdf.leftboundary exch put }
3574       ifelse
3575     }
3576     put
3577   }
3578   if
3579 }
3580 def
3581 /pdf.bordertracking.end
3582 {
3583   /a where { /a { moveto } put } if
3584   /x where { /x { 0 exch rmoveto } put } if
3585   SDict /pdf.leftboundary known
3586   { pdf.outerbox 0 pdf.leftboundary put }
3587   if
3588   SDict /pdf.rightboundary known
3589   { pdf.outerbox 2 pdf.rightboundary put }

```

```

3590     if
3591     SDict /pdf.bordertracking false put
3592   }
3593   def
3594   /pdf.bordertracking.endpage
3595   {
3596   pdf.bordertracking
3597   {
3598     pdf.bordertracking.end
3599     true setglobal
3600     pdf.globaldict
3601       /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3602     pdf.globaldict
3603       /pdf.brokenlink.skip pdf.baselineskip put
3604     pdf.globaldict
3605       /pdf.brokenlink.dict
3606         pdf.link.dict pdf.cvs put
3607     false setglobal
3608     mark pdf.link.dict cvx exec /Rect
3609     [
3610       pdf.llx
3611       pdf.lly
3612       pdf.outerbox 2 get pdf.linkmargin add
3613       currentpoint exch pop
3614       pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3615     ]
3616     /ANN pdf.pdfmark
3617   }
3618   if
3619   }
3620   def
3621   /pdf.bordertracking.continue
3622   {
3623     /pdf.link.dict pdf.globaldict
3624     /pdf.brokenlink.dict get def
3625     /pdf.outerbox pdf.globaldict
3626     /pdf.brokenlink.rect get def
3627     /pdf.baselineskip pdf.globaldict
3628     /pdf.brokenlink.skip get def
3629     pdf.globaldict dup dup
3630     /pdf.brokenlink.dict undef
3631     /pdf.brokenlink.skip undef
3632     /pdf.brokenlink.rect undef
3633     currentpoint
3634     /pdf.originy exch def
3635     /pdf.originx exch def
3636     /a where
3637     {
3638       /a
3639       {
3640         moveto
3641         SDict
3642         begin
3643         currentpoint pdf.originy ne exch

```

```

3644         pdf.originx ne or
3645         {
3646             pdf.save.linkll
3647             /pdf.lly
3648             pdf.lly pdf.outerbox 1 get sub def
3649             pdf.bordertracking.begin
3650         }
3651     if
3652     end
3653 }
3654 put
3655 }
3656 if
3657 /x where
3658 {
3659     /x
3660     {
3661         0 exch rmoveto
3662         SDict
3663         begin
3664             currentpoint
3665             pdf.originy ne exch pdf.originx ne or
3666             {
3667                 pdf.save.linkll
3668                 /pdf.lly
3669                 pdf.lly pdf.outerbox 1 get sub def
3670                 pdf.bordertracking.begin
3671             }
3672             if
3673             end
3674         }
3675         put
3676     }
3677     if
3678 }
3679 def

```

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

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

```

3680 /pdf.breaklink
3681 {
3682     pop
3683     counttomark 2 mod 0 eq
3684     {
3685         counttomark /pdf.count exch def
3686         {
3687             pdf.count 0 eq { exit } if
3688             counttomark 2 roll
3689             1 index /Rect eq

```

```

3690 {
3691     dup 4 array copy
3692     dup dup
3693     1 get
3694     pdf.outerbox pdf.rect.ht
3695     pdf.linkmargin 2 mul add sub
3696     3 exch put
3697     dup
3698     pdf.outerbox 2 get
3699     pdf.linkmargin add
3700     2 exch put
3701     dup dup
3702     3 get
3703     pdf.outerbox pdf.rect.ht
3704     pdf.linkmargin 2 mul add add
3705     1 exch put
3706     /pdf.currentrect exch def
3707     pdf.breaklink.write
3708     {
3709         pdf.currentrect
3710         dup
3711         pdf.outerbox 0 get
3712         pdf.linkmargin sub
3713         0 exch put
3714         dup
3715         pdf.outerbox 2 get
3716         pdf.linkmargin add
3717         2 exch put
3718         dup dup
3719         1 get
3720         pdf.baselineskip add
3721         1 exch put
3722         dup dup
3723         3 get
3724         pdf.baselineskip add
3725         3 exch put
3726         /pdf.currentrect exch def
3727         pdf.breaklink.write
3728     }
3729     1 index 3 get
3730     pdf.linkmargin 2 mul add
3731     pdf.outerbox pdf.rect.ht add
3732     2 index 1 get sub
3733     pdf.baselineskip div round cvi 1 sub
3734     exch
3735     repeat
3736     pdf.currentrect
3737     dup
3738     pdf.outerbox 0 get
3739     pdf.linkmargin sub
3740     0 exch put
3741     dup dup
3742     1 get
3743     pdf.baselineskip add

```

```

3744         1 exch put
3745     dup dup
3746     3 get
3747     pdf.baselineskip add
3748     3 exch put
3749     dup 2 index 2 get 2 exch put
3750     /pdf.currentrect exch def
3751     pdf.breaklink.write
3752     SDict /pdf.pdfmark.good false put
3753     exit
3754 }
3755 { pdf.count 2 sub /pdf.count exch def }
3756 ifelse
3757 }
3758 loop
3759 }
3760 if
3761 /ANN
3762 }
3763 def
3764 /pdf.breaklink.write
3765 {
3766     counttomark 1 sub
3767     index /_objdef eq
3768     {
3769         counttomark -2 roll
3770         dup wcheck
3771         {
3772             readonly
3773             counttomark 2 roll
3774         }
3775         { pop pop }
3776     } ifelse
3777 }
3778 if
3779 counttomark 1 add copy
3780 pop pdf.currentrect
3781 /ANN pdfmark
3782 }
3783 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 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.

```

3784 /pdf.pdfmark
3785 {
3786     SDict /pdf.pdfmark.good true put
3787     dup /ANN eq
3788     {
3789         pdf.pdfmark.store

```

```

3790     pdf.pdfmark.dict
3791     begin
3792         Subtype /Link eq
3793         currentdict /Rect known and
3794         SDict /pdf.outerbox known and
3795         SDict /pdf.baselineskip known and
3796         {
3797             Rect 3 get
3798             pdf.linkmargin 2 mul add
3799             pdf.outerbox pdf.rect.ht add
3800             Rect 1 get sub
3801             pdf.baselineskip div round cvi 0 gt
3802             { pdf.breaklink }
3803             if
3804         }
3805         if
3806         end
3807         SDict /pdf.outerbox undef
3808         SDict /pdf.baselineskip undef
3809         currentdict /pdf.pdfmark.dict undef
3810     }
3811     if
3812     pdf.pdfmark.good
3813     { pdfmark }
3814     { cleartomark }
3815     ifelse
3816 }
3817 def
3818 /pdf.pdfmark.store
3819 {
3820     /pdf.pdfmark.dict 65534 dict def
3821     counttomark 1 add copy
3822     pop
3823     {
3824         dup mark eq
3825         {
3826             pop
3827             exit
3828         }
3829         {
3830             pdf.pdfmark.dict
3831             begin def end
3832         }
3833     } ifelse
3834 }
3835 loop
3836 }
3837 def

```

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

```

3838 </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>	1137
A	
<code>\AtBeginDvi</code>	56
B	
bool commands:	
<code>\bool_gset_false:N</code>	1223, 1242, 1265, 1287, 1303, 1412, 1664, 1700, 2938, 2984
<code>\bool_gset_true:N</code>	1221, 1290, 1410, 1679, 2931, 2937
<code>\bool_if:NTF</code>	66, 596, 1233, 1237, 1253, 1256, 1260, 1271, 1278, 1282, 1294, 1298, 1423, 1428, 1433, 1638, 1683, 1812, 1864, 1866, 2005, 2050, 2052, 2926, 2941, 2946, 2951
<code>\bool_if:nTF</code>	2429, 2610, 2798
<code>\bool_lazy_and:nnTF</code>	809, 2157, 3314, 3390
<code>\bool_lazy_any:nTF</code>	1852, 2040
<code>\bool_new:N</code>	1224, 1291, 1413, 1680, 1802, 1968, 2911, 2912
<code>\bool_set_false:N</code>	1807, 1825, 1964, 1984, 2075, 2225
<code>\bool_set_true:N</code>	1824, 1992
box commands:	
<code>\box_dp:N</code>	235, 237, 285, 287, 342, 344, 391, 393, 395, 397, 2963, 2996, 2997, 3022
<code>\box_ht:N</code>	237, 287, 344, 395, 397, 1879, 2116, 2968, 3007, 3008, 3024
<code>\box_if_empty:N</code>	3056
<code>\box_move_down:nn</code>	2884, 2963
<code>\box_move_up:nn</code>	2246, 2886, 2968
<code>\box_new:N</code>	2874, 2875
<code>\box_set_dp:Nn</code>	1771
<code>\box_set_ht:Nn</code>	1770
<code>\box_set_wd:Nn</code>	299, 1769
<code>\box_use:N</code>	242, 260, 274, 290, 317, 331, 347, 363, 375, 426, 440, 459, 1363, 1571, 1772, 2916
<code>\box_wd:N</code>	236, 244, 286, 292, 343, 349, 392, 394, 1878, 2115
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>	1311, 1443, 1707
<code>\clist_map_inline:nn</code>	3242
color internal commands:	
<code>__color_backend:nnn</code>	1045, 1060, 1068, 1074
<code>\g__color_backend_colorant_prop</code>	562, 581, 584, 604, 845
<code>__color_backend_devicen_colorants:n</code>	563, 563, 765, 903
<code>__color_backend_devicen_colorants:w</code>	563, 571, 578, 586
<code>__color_backend_devicen_init:nnn</code>	752, 752, 870, 870, 1095, 1095
<code>__color_backend_devicen_init:w</code>	870, 879, 908, 912
<code>__color_backend_fill:n</code>	949, 949, 951, 952, 953, 975, 976, 978, 980, 981, 1000, 1009, 1010, 1012, 1014, 1015, 1026, 1035, 1036, 1038, 1040, 1041
<code>__color_backend_fill_cmyk:n</code>	949, 951, 975, 975, 1009, 1009, 1035, 1035, 1047
<code>__color_backend_fill_devicen:nn</code>	959, 969, 999, 1003, 1025, 1029, 1089, 1091
<code>__color_backend_fill_gray:n</code>	949, 952, 975, 977, 1009, 1011, 1035, 1037
<code>__color_backend_fill_reset</code>	971, 971, 1005, 1005, 1031, 1031, 1093, 1093
<code>__color_backend_fill_rgb:n</code>	949, 953, 975, 979, 1009, 1013, 1035, 1039
<code>__color_backend_fill_separation:nn</code>	959, 959, 969, 999, 999, 1003, 1025, 1025, 1029, 1089, 1089, 1091

\l_color_backend_fill_tl	_color_backend_separation_
..... 525, 537, 983, 997	init_aux:nnnnnn 594, 600, 616
_color_backend_iccbased_	_color_backend_separation_
device:nnn	init_CIELAB:nnn
932, 932 594, 706, 776, 823, 848
_color_backend_iccbased_	_color_backend_separation_
init:nnn	init_CIELAB:nnnnnn
..... 771, 771, 914, 914, 1095, 1096	777
_color_backend_init_resource:n	_color_backend_separation_
..... 806, 806, 835, 906, 930, 945	init_count:n
_color_backend_reset:	594, 653, 656
..... 506, 521, 529, 541, 545, 550,	_color_backend_separation_
971, 972, 1005, 1006, 1031, 1049, 1093	init_count:w ... 594, 657, 658, 662
_color_backend_rgb:w	_color_backend_separation_
1062	init_Device:Nn
_color_backend_select:n 594, 638, 640, 642, 643
..... 506, 507, 509, 511,	\l_color_backend_stack_int
513, 514, 545, 545, 547, 548, 549, 591 467, 539, 542, 984, 996
_color_backend_select:nn	_color_backend_stroke:n
..... 529, 530, 532, 534, 535, 802 949, 954, 956,
_color_backend_select_cmyk:n ..	957, 958, 975, 988, 990, 992, 993, 1002
..... 506, 506, 529, 529, 545, 547	_color_backend_stroke_cmyk:n ..
_color_backend_select_devicen:nn 949,
..... 590, 592, 774, 775, 796, 804	956, 975, 987, 1009, 1019, 1045, 1045
_color_backend_select_gray:n ..	_color_backend_stroke_devicen:nn
..... 506, 508, 529, 531, 545, 548, 555 959,
_color_backend_select_iccbased:nn	970, 999, 1004, 1025, 1030, 1089, 1092
..... 593, 593, 778, 778, 796, 805	_color_backend_stroke_gray:n ..
_color_backend_select_named:n 949,
..... 506, 510, 552, 552	957, 975, 989, 1009, 1021, 1045, 1051
_color_backend_select_rgb:n ...	_color_backend_stroke_gray_
..... 506, 512, 529, 533, 545, 549	aux:n
_color_backend_select_separation:nn	1045, 1055, 1059
..... 590, 590, 592,	_color_backend_stroke_reset: ..
774, 774, 775, 796, 797, 801, 804, 805 971,
_color_backend_separation_	972, 1005, 1006, 1031, 1032, 1093, 1094
init:n	_color_backend_stroke_rgb:n ...
594, 675, 688 949,
_color_backend_separation_	958, 975, 991, 1009, 1023, 1045, 1061
init:nn	_color_backend_stroke_rgb:w ...
823, 833, 837 1045, 1063
_color_backend_separation_	_color_backend_stroke_separation:nn
init:nnn 959, 964, 970, 999, 1001, 1004,
594, 629, 650	1025, 1027, 1030, 1089, 1090, 1092
_color_backend_separation_	\l_color_backend_stroke_tl
init:nnnn 525, 538, 985, 995
594, 615, 708, 776, 776, 823, 823, 863	\g_color_model_int 601, 610, 758,
_color_backend_separation_	786, 835, 841, 842, 896, 897, 906, 930
init:nw	\c_color_model_range_CIELAB_tl .
594, 679, 690, 704 713, 748, 859, 866
_color_backend_separation_	color.sc
init:w	3457
594, 666, 681, 686	cs commands:
_color_backend_separation_	\cs_generate_variant:Nn .. 62, 65,
init_/DeviceCMYK:nnn	170, 181, 212, 218, 615, 1169, 1580,
594	2019, 2086, 2106, 2273, 2288, 2351,
_color_backend_separation_	2561, 2574, 2684, 2699, 2729, 3177
init_/DeviceGray:nnn	\cs_gset:Npe .. 2441, 2445, 2803, 2808
594	
_color_backend_separation_	
init_/DeviceRGB:nnn	
594	

<code>\cs_gset_protected:Npn</code> . . .	3394, 3395	1430, 1435, 1437, 1450, 1455, 1457,
<code>\cs_if_exist:NTF</code>		1459, 1461, 1463, 1465, 1467, 1469,
	27, 49, 2635, 2661, 3052	1488, 1512, 1518, 1530, 1542, 1554,
<code>\cs_if_exist_p:N</code>	810, 3315, 3391	1561, 1583, 1589, 1594, 1599, 1610,
<code>\cs_if_exist_use:NTF</code>	38, 628	1620, 1630, 1632, 1634, 1636, 1667,
<code>\cs_new:Npe</code>		1669, 1674, 1676, 1678, 1681, 1702,
	563, 2575, 2586, 2653, 3103, 3138, 3234	1713, 1726, 1728, 1730, 1732, 1734,
<code>\cs_new:Npn</code>	578, 637, 639,	1736, 1738, 1740, 1742, 1750, 1758,
	641, 643, 650, 656, 658, 664, 681,	1784, 1803, 1821, 1836, 1841, 1849,
	688, 690, 908, 1316, 1448, 1711,	1882, 1895, 1913, 1923, 1939, 1952,
	1881, 2119, 2263, 2280, 2352, 2354,	1961, 1970, 1982, 1990, 1995, 2010,
	2447, 2448, 2530, 2531, 2543, 2562,	2020, 2063, 2072, 2078, 2084, 2087,
	2563, 2666, 2692, 2730, 2732, 2811,	2094, 2107, 2112, 2120, 2133, 2167,
	2812, 2824, 2825, 2830, 2831, 2836,	2198, 2199, 2201, 2203, 2205, 2211,
	2837, 2906, 3077, 3191, 3220, 3229	2214, 2222, 2228, 2231, 2233, 2244,
<code>\cs_new_eq:NN</code>	46, 56, 58, 547,	2271, 2274, 2276, 2278, 2282, 2289,
	548, 549, 592, 775, 804, 805, 951,	2306, 2311, 2316, 2321, 2331, 2336,
	952, 953, 956, 957, 958, 969, 970,	2344, 2356, 2382, 2387, 2415, 2427,
	971, 972, 1003, 1004, 1005, 1006,	2439, 2443, 2449, 2451, 2455, 2478,
	1029, 1030, 1031, 1091, 1092, 1093,	2492, 2502, 2513, 2532, 2564, 2597,
	1168, 1372, 1373, 1378, 1379, 1579,	2608, 2614, 2642, 2676, 2678, 2685,
	1581, 1582, 1588, 1782, 1783, 1795,	2687, 2690, 2694, 2700, 2705, 2710,
	1796, 1819, 1820, 1887, 1888, 1889,	2712, 2714, 2722, 2734, 2756, 2761,
	1912, 1937, 1949, 1950, 1958, 1959,	2794, 2796, 2801, 2806, 2813, 2815,
	1960, 1981, 1987, 1988, 1989, 2059,	2819, 2820, 2821, 2822, 2823, 2826,
	2069, 2070, 2071, 2212, 2213, 2220,	2827, 2828, 2829, 2832, 2833, 2834,
	2221, 2230, 2260, 2261, 2262, 2265,	2835, 2838, 2839, 2842, 2861, 2868,
	2281, 2693, 2916, 3240, 3241, 3251	2877, 2882, 2915, 2917, 2922, 2924,
<code>\cs_new_protected:Npe</code>		2929, 2944, 2949, 2986, 3015, 3034,
	. . . 594, 1074, 2625, 2682, 3175, 3201	3043, 3079, 3086, 3087, 3090, 3114,
<code>\cs_new_protected:Npn</code> 47, 53, 60, 63,		3116, 3118, 3129, 3149, 3159, 3166,
	71, 77, 82, 84, 88, 98, 108, 118, 128,	3179, 3194, 3199, 3218, 3222, 3224,
	137, 146, 156, 168, 171, 173, 175,	3225, 3228, 3230, 3231, 3232, 3233,
	179, 184, 193, 203, 213, 224, 246,	3235, 3236, 3237, 3256, 3263, 3271,
	248, 263, 279, 294, 296, 322, 336,	3279, 3298, 3305, 3329, 3345, 3355,
	351, 353, 366, 380, 430, 443, 470,	3361, 3367, 3399, 3401, 3403, 3405
	484, 494, 506, 508, 510, 512, 514,	<code>\cs_set_eq:NN</code>
	521, 529, 531, 533, 535, 541, 545,	3073, 3075
	550, 552, 590, 593, 616, 706, 752,	<code>\cs_set_protected:Npn</code>
	771, 774, 776, 777, 778, 797, 801,	2171
	806, 823, 837, 848, 870, 914, 932,	
	949, 954, 959, 964, 975, 977, 979,	
	981, 987, 989, 991, 993, 999, 1001,	
	1009, 1011, 1013, 1015, 1019, 1021,	
	1023, 1025, 1027, 1032, 1035, 1037,	
	1039, 1041, 1045, 1051, 1059, 1061,	
	1063, 1089, 1090, 1094, 1095, 1096,	
	1170, 1176, 1181, 1183, 1185, 1193,	
	1201, 1210, 1220, 1222, 1225, 1227,	
	1244, 1249, 1267, 1289, 1292, 1305,	
	1318, 1323, 1325, 1327, 1329, 1331,	
	1333, 1335, 1337, 1342, 1347, 1374,	
	1376, 1380, 1385, 1390, 1400, 1409,	
	1411, 1414, 1416, 1418, 1420, 1425,	

D

dim commands:

<code>\dim_compare:nNnTF</code>	2147, 2152
<code>\dim_compare_p:nNn</code>	2158, 2159
<code>\dim_eval:n</code>	
	. . . 2385, 2488, 2489, 2490, 2759,
	2850, 2851, 2854, 2880, 3098, 3099,
	3100, 3157, 3185, 3186, 3187, 3223
<code>\dim_gset:Nn</code>	2863, 2864
<code>\dim_max:nn</code>	2994, 3005
<code>\dim_set:Nn</code>	
	. . 1878, 1879, 2115, 2116, 2143, 2144
<code>\dim_set_eq:NN</code>	2209
<code>\dim_to_decimal:n</code>	391, 392, 393,
	394, 395, 397, 1592, 1597, 1603,

```

1604, 1605, 1606, 1615, 1616, 1617,
1708, 1727, 2253, 2254, 2992, 3003,
3021, 3022, 3023, 3024, 3028, 3083
\dim_to_decimal_in_bp:n .....
.... 235, 236, 237, 285, 286, 287,
342, 343, 344, 1189, 1190, 1197,
1198, 1205, 1206, 1214, 1215, 1216,
1313, 1317, 1321, 1383, 1388, 1394,
1395, 1396, 1404, 1405, 1445, 1449,
1453, 1712, 1789, 1790, 1791, 1792,
1975, 1976, 1977, 1978, 2034, 2035,
2036, 2037, 2238, 2239, 2240, 2241
\dim_zero:N ..... 2141, 2142
\c_max_dim .....
.. 2143, 2144, 2147, 2152, 2158, 2159
draw internal commands:
\__draw_backend_add_to_path:n ...
..... 1589,
1591, 1596, 1601, 1612, 1620, 1635
\__draw_backend_begin: .....
.. 1170, 1170, 1374, 1374, 1583, 1583
\__draw_backend_box_use:Nnnnn ...
.. 1347, 1347, 1561, 1561, 1758, 1758
\__draw_backend_cap_but: .....
.. 1305, 1325, 1437, 1457, 1702, 1730
\__draw_backend_cap_rectangle: ..
.. 1305, 1329, 1437, 1461, 1702, 1734
\__draw_backend_cap_round: .....
.. 1305, 1327, 1437, 1459, 1702, 1732
\__draw_backend_clip: .....
.. 1225, 1289, 1414, 1430, 1634, 1678
\__draw_backend_closepath: .....
..... 1225, 1225,
1246, 1414, 1414, 1634, 1634, 1671
\__draw_backend_closestroke: ...
.. 1225, 1244, 1414, 1418, 1634, 1669
\__draw_backend_curveto:nnnnnn ..
.. 1185, 1210, 1380, 1390, 1589, 1610
\__draw_backend_dash:n .....
..... 1305, 1311, 1316,
1437, 1443, 1448, 1702, 1707, 1711
\__draw_backend_dash_aux:nn ....
..... 1702, 1706, 1713
\__draw_backend_dash_pattern:nn .
.. 1305, 1305, 1437, 1437, 1702, 1702
\__draw_backend_discardpath: ...
.. 1225, 1292, 1414, 1435, 1634, 1681
\__draw_backend_end: .....
.. 1170, 1176, 1374, 1376, 1583, 1588
\__draw_backend_evenodd_rule: ...
.. 1220, 1220, 1409, 1409, 1630, 1630
\__draw_backend_fill: .....
.. 1225, 1249, 1414, 1420, 1634, 1674
\__draw_backend_fillstroke: ....
.. 1225, 1267, 1414, 1425, 1634, 1676
\__draw_backend_join_bevel: ....
.. 1305, 1335, 1437, 1467, 1702, 1740
\__draw_backend_join_miter: ....
.. 1305, 1331, 1437, 1463, 1702, 1736
\__draw_backend_join_round: ....
.. 1305, 1333, 1437, 1465, 1702, 1738
\__draw_backend_lineto:nn .....
.. 1185, 1193, 1380, 1385, 1589, 1594
\__draw_backend_linewidth:n ....
.. 1305, 1318, 1437, 1450, 1702, 1726
\__draw_backend_literal:n .....
1168, 1168, 1169, 1172, 1173, 1174,
1178, 1179, 1182, 1184, 1187, 1195,
1203, 1212, 1226, 1229, 1230, 1231,
1232, 1235, 1241, 1251, 1258, 1264,
1269, 1274, 1275, 1276, 1277, 1280,
1286, 1296, 1302, 1307, 1320, 1324,
1326, 1328, 1330, 1332, 1334, 1336,
1339, 1344, 1349, 1350, 1351, 1352,
1353, 1354, 1355, 1356, 1357, 1361,
1362, 1364, 1365, 1366, 1367, 1368,
1372, 1372, 1373, 1382, 1387, 1392,
1402, 1415, 1417, 1419, 1422, 1427,
1432, 1436, 1439, 1452, 1456, 1458,
1460, 1462, 1464, 1466, 1468, 1514,
1579, 1579, 1580, 1641, 1660, 1686
\__draw_backend_miterlimit:n ...
.. 1305, 1323, 1437, 1455, 1702, 1728
\__draw_backend_moveto:nn .....
.. 1185, 1185, 1380, 1380, 1589, 1589
\__draw_backend_nonzero_rule: ...
.. 1220, 1222, 1409, 1411, 1630, 1632
\__draw_backend_path:n .....
..... 1634, 1636, 1668, 1675, 1677
\g__draw_backend_path_int 1649, 1666
\g__draw_backend_path_tl .....
.. 1589, 1645, 1661, 1663, 1690, 1699
\__draw_backend_rectangle:nnnn ..
.. 1185, 1201, 1380, 1400, 1589, 1599
\__draw_backend_scope_begin: 1181,
1181, 1375, 1378, 1378, 1581, 1581
\__draw_backend_scope_end: 1181,
1183, 1377, 1378, 1379, 1581, 1582
\__draw_backend_shift:nn .....
.. 1337, 1342, 1469, 1512, 1742, 1750
\__draw_backend_stroke: 1225, 1227,
1247, 1414, 1416, 1634, 1667, 1672
\__draw_backend_transform:nnnn ..
..... 1337, 1337, 1358, 1359,
1360, 1469, 1469, 1742, 1742, 1761
\__draw_backend_transform_-
aux:nnnn ..... 1469, 1483, 1488

```

`__draw_backend_transform_-`
`decompose:nnnnN` . 1482, 1517, 1518
`__draw_backend_transform_-`
`decompose_auxi:nnnnN`
..... 1517, 1522, 1530
`__draw_backend_transform_-`
`decompose_auxii:nnnnN`
..... 1517, 1534, 1542
`__draw_backend_transform_-`
`decompose_auxiii:nnnnN`
..... 1517, 1546, 1554
`\g__draw_draw_clip_bool` .. 1225, 1634
`\g__draw_draw_eor_bool`
... 1220, 1237, 1253, 1260, 1271,
1282, 1298, 1409, 1423, 1428, 1433
`\g__draw_draw_path_int` 1634

E

`\errmessage` 38
`\evensidemargin` 2961
exp commands:
`\exp_args:Ne` 598,
652, 833, 1843, 1901, 1903, 1927,
1929, 2318, 2333, 2384, 2758, 2957
`\exp_args:Nf` 1310, 1442, 2879
`\exp_args:Nne` 2725
`\exp_args:NNf` 247, 295, 352
`\exp_args:Nno` 3357
`\exp_args:No` 3363
`\exp_not:N` 565,
571, 572, 573, 598, 600, 601, 604,
605, 610, 2577, 2579, 2582, 2588,
2590, 2593, 2630, 2631, 2637, 2638,
2657, 2662, 3105, 3107, 3110, 3140,
3142, 3145, 3203, 3204, 3205, 3210
`\exp_not:n` 48, 96, 116, 154,
922, 2309, 2314, 2378, 2547, 2548,
2562, 2563, 2703, 2708, 2719, 2738
`\ExplBackendFileDate` 1

F

file commands:
`\file_compare_timestamp:nNnTF` . 1915
`\file_parse_full_name:nNNN` 1897, 1925
`\fmtversion` 51
fp commands:
`\fp_compare:nNnTF`
. 254, 301, 307, 359, 1493, 1506, 1556
`\fp_eval:n`
. 247, 256, 269, 270, 295, 312, 327,
329, 352, 361, 372, 373, 437, 452,
453, 1056, 1069, 1070, 1071, 1495,
1500, 1501, 1508, 1523, 1524, 1525,

1526, 1535, 1536, 1537, 1538, 1547,
1548, 1549, 1550, 2375, 2475, 2752
`\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, 1493, 1506

G

graphics commands:
`\l_graphics_search_ext_seq`
..... 1781, 1799, 1947, 2131
graphics internal commands:
`\l_graphics_attr_tl` 1801,
1808, 1826, 1838, 1845, 1847, 1885
`__graphics_backend_dequote:w` ...
..... 1803, 1844, 1881
`\l_graphics_backend_dir_str` . 1890
`\l_graphics_backend_ext_str` . 1890
`__graphics_backend_get_pagecount:n`
..... 1796, 1796, 1939, 1939,
2058, 2059, 2120, 2120, 2265, 2265
`__graphics_backend_getbb_auxi:n`
..... 1803, 1817, 1834, 1836
`__graphics_backend_getbb_-`
`auxi:nN` 2063, 2067, 2076, 2078
`__graphics_backend_getbb_-`
`auxii:n` 1803, 1839, 1841
`__graphics_backend_getbb_-`
`auxiii:nnN` .. 2063, 2081, 2084, 2086
`__graphics_backend_getbb_-`
`auxiiii:n` 1803, 1843, 1849
`__graphics_backend_getbb_-`
`auxiii:nNnn` . 2063, 2082, 2085, 2087
`__graphics_backend_getbb_-`
`auxiv:nnNnn` . 2063, 2090, 2094, 2106
`__graphics_backend_getbb_-`
`auxv:nNnn` .. 2063, 2091, 2098, 2107
`__graphics_backend_getbb_-`
`auxvi:nNnn` 2110, 2112
`__graphics_backend_getbb_bmp:n` .
..... 1949, 1960, 2063, 2071
`__graphics_backend_getbb_eps:n` .
..... 1782, 1782, 1890,
1895, 1912, 1949, 1949, 2212, 2212
`__graphics_backend_getbb_eps:nm`
..... 1890
`__graphics_backend_getbb_eps:n`
..... 1901, 1913
`__graphics_backend_getbb_jpeg:n`
..... 1803, 1819,
1949, 1958, 2063, 2069, 2214, 2220
`__graphics_backend_getbb_jpg:n` .
..... 1803, 1803, 1819, 1820, 1949, 1952,

1958, 1959, 1960, 2063, 2063, 2069,
 2070, 2071, 2214, 2214, 2220, 2221
 _graphics_backend_getbb_-
 pagebox:w 2063, 2102, 2119
 _graphics_backend_getbb_pdf:n .
 1803, 1821, 1921,
 1949, 1961, 2063, 2072, 2222, 2222
 _graphics_backend_getbb_png:n .
 1803, 1820,
 1949, 1959, 2063, 2070, 2214, 2221
 _graphics_backend_getbb_ps:n ..
 1782, 1783,
 1890, 1912, 1949, 1950, 2212, 2213
 _graphics_backend_getbb_svg:n .
 2133, 2133
 _graphics_backend_getbb_svg_-
 auxi:nNn ... 2133, 2149, 2154, 2167
 _graphics_backend_getbb_svg_-
 auxii:w 2133, 2171, 2193, 2198
 _graphics_backend_getbb_svg_-
 auxiii:Nw 2133, 2181, 2199
 _graphics_backend_getbb_svg_-
 auxiv:Nw 2133, 2184, 2201
 _graphics_backend_getbb_svg_-
 auxv:Nw 2133, 2185, 2203
 _graphics_backend_getbb_svg_-
 auxvi:Nn 2133, 2200, 2202, 2204, 2205
 _graphics_backend_getbb_svg_-
 auxvii:w 2133, 2207, 2211
 _graphics_backend_include:nn ..
 2228, 2229, 2232, 2233
 _graphics_backend_include_-
 auxi:n 1970, 1985, 1993, 1995
 _graphics_backend_include_-
 auxii:nn ... 1970, 1997, 2010, 2019
 _graphics_backend_include_-
 auxiii:nn 1970, 2017, 2020
 _graphics_backend_include_-
 bmp:n 1970, 1988
 _graphics_backend_include_-
 dequote:w 2244, 2255, 2263
 _graphics_backend_include_-
 eps:n 1784,
 1784, 1795, 1890, 1923, 1937,
 1970, 1970, 1981, 2228, 2228, 2230
 _graphics_backend_include_-
 jpeg:n . 1882, 1887, 1987, 2244, 2261
 _graphics_backend_include_-
 jpg:n 1882,
 1882, 1887, 1888, 1889, 1970,
 1982, 1987, 1988, 1989, 2244, 2262
 _graphics_backend_include_-
 jpseg:n 1970
 _graphics_backend_include_-
 pdf:n 1882,
 1888, 1927, 1970, 1990, 2228, 2231
 _graphics_backend_include_-
 png:n
 .. 1882, 1889, 1970, 1989, 2244, 2260
 _graphics_backend_include_ps:n
 1784, 1795,
 1890, 1937, 1970, 1981, 2228, 2230
 _graphics_backend_include_-
 svg:n .. 2244, 2244, 2260, 2261, 2262
 \l_graphics_backend_name_str . 1890
 _graphics_bb_restore:nTF
 1838, 2109, 2135
 _graphics_bb_save:n 1847, 2117, 2162
 \l_graphics_decodearray_str ...
 1810, 1811,
 1823, 1856, 1862, 1863, 1963, 2003,
 2004, 2044, 2048, 2049, 2074, 2224
 _graphics_extract_bb:n
 1956, 1965, 2218, 2226
 \l_graphics_final_name_str .. 1920
 _graphics_get_pagecount:n
 1796, 2059, 2265
 \l_graphics_interpolate_bool ...
 1812, 1825, 1854, 1866,
 1964, 2005, 2042, 2052, 2075, 2225
 \l_graphics_llx_dim
 1789, 1975, 2034, 2141, 2238
 \l_graphics_lly_dim
 1790, 1976, 2035, 2142, 2239
 \l_graphics_page_int 1805, 1829,
 1830, 1871, 1872, 1954, 2001, 2002,
 2028, 2029, 2065, 2080, 2081, 2216
 \l_graphics_pagebox_tl ... 1806,
 1828, 1873, 1874, 1955, 1999, 2000,
 2030, 2032, 2066, 2089, 2090, 2217
 \l_graphics_pdf_str
 .. 1814, 1815, 1831, 1832, 1857, 1868
 _graphics_read_bb:n
 .. 1782, 1783, 1949, 1950, 2212, 2213
 \l_graphics_tmp_box
 .. 1876, 1878, 1879, 2114, 2115, 2116
 \l_graphics_tmp_dim 2208, 2209
 \l_graphics_tmp_ior
 2137, 2138, 2145, 2164
 \g_graphics_track_int
 1969, 2022, 2023
 \l_graphics_transgroup_bool ...
 1802, 1807, 1824, 1855,
 1864, 1968, 1984, 1992, 2043, 2050
 \l_graphics_urx_dim
 ... 1791, 1878, 1977, 2036, 2115,
 2143, 2147, 2150, 2158, 2240, 2253

`\l_graphics_ury_dim`
 1792, 1879, 1978, 2037, 2116, 2144,
 2152, 2155, 2159, 2241, 2246, 2254
 group commands:
`\group_begin:` 190, 209
`\group_end:` 198
`\group_insert_after:N`
 .. 3260, 3261, 3269, 3277, 3343, 3387

H

hbox commands:
`\hbox:n` 2248, 2392, 2399,
 2766, 2777, 2885, 2888, 2964, 2970
`\hbox_overlap_right:n` 242,
 274, 290, 331, 347, 375, 459, 1363, 1571
`\hbox_set:Nn` .. 1876, 2114, 2956, 2988
`\hbox_set:Nw` 2939
`\hbox_set_end:` 2954
`\hbox_unpack:N` 3075
 hook commands:
`\hook_gput_code:nnn` .. 54, 3052, 3054

I

int commands:
`\int_compare:nNnTF` 1829, 1871, 2001,
 2028, 2080, 2417, 2628, 2656, 3047
`\int_const:Nn`
 472, 1845, 1942, 2023, 2122
`\int_eval:n` 492, 502, 648, 657, 670,
 672, 676, 689, 2441, 2445, 2606,
 2631, 2638, 2651, 2795, 2803, 2808
`\int_gincr:N` 216,
 382, 1640, 1685, 2022, 2279, 2346,
 2691, 2724, 2895, 2973, 3181, 3203
`\int_gset:Nn` 191, 210, 2522, 3036
`\int_gset_eq:NN` 199, 2974, 3204
`\int_if_exist:NTF` 2012
`\int_if_odd:nTF` 2959
`\int_max:nn` 2124
`\int_new:N` 182, 183, 429, 467, 1666,
 1969, 2876, 2908, 2910, 3178, 3193
`\int_set:Nn` 3048
`\int_set_eq:NN` 187, 206
`\int_step_function:nnnN` 674
`\int_use:N` 384,
 415, 601, 610, 758, 786, 835, 841,
 842, 896, 897, 906, 930, 1643, 1649,
 1656, 1688, 1696, 1830, 1872, 1885,
 1943, 2002, 2015, 2027, 2029, 2125,
 2348, 2353, 2726, 2731, 2899, 2907,
 2978, 3078, 3184, 3192, 3210, 3221
`\int_value:w`
 2577, 2588, 2606, 3105, 3140
`\int_zero:N` ... 1805, 1954, 2065, 2216

ior commands:
`\ior_close:N` 2164
`\ior_if_eof:NTF` 2138
`\ior_map_break:` 2160
`\ior_open:Nn` 2137
`\ior_str_map_inline:Nn` 2145

K

kernel internal commands:
`__kernel_backend_align_begin:` ..
 71, 71, 227, 251, 266
`__kernel_backend_align_end:` ...
 71, 77, 241, 259, 273
`__kernel_backend_first_shipout:n`
 49, 53, 56, 58, 68, 598, 2844
`\g_kernel_backend_header_bool` ..
 66, 596
`__kernel_backend_literal:n`
 46, 46, 47, 48, 61, 64, 69,
 73, 80, 83, 85, 169, 172, 174, 176,
 180, 356, 369, 516, 522, 546, 551,
 618, 754, 798, 950, 955, 961, 966,
 1017, 1043, 1477, 1478, 1479, 1490,
 1497, 1503, 1568, 1573, 1786, 1972,
 2014, 2024, 2235, 2250, 2683, 2795,
 2799, 2804, 2809, 2846, 3176, 3223
`__kernel_backend_literal_page:n`
 108, 108,
 118, 171, 171, 2677, 2679, 2814, 2816
`__kernel_backend_literal_pdf:n` ..
 88, 88, 98, 168, 168, 170,
 282, 339, 1372, 1373, 3337, 3348, 3381
`__kernel_backend_literal_-
 postscript:n` 60,
 60, 62, 74, 75, 79, 228, 229, 231,
 232, 240, 252, 267, 1168, 2419, 2431
`__kernel_backend_literal_svg:n` ..
 . 179, 179, 181, 186, 197, 205, 215,
 383, 385, 402, 780, 1579, 1762, 1773
`__kernel_backend_matrix:n`
 .. 146, 146, 156, 304, 325, 1472, 1565
`__kernel_backend_postscript:n` ..
 63, 63, 65, 518,
 1020, 1022, 1024, 1028, 2272, 2323,
 2338, 2358, 2392, 2399, 2885, 2891,
 2896, 2932, 2964, 2971, 2975, 2989,
 3017, 3060, 3067, 3074, 3081, 3281
`__kernel_backend_scope:n`
 ... 184, 213, 218, 412, 417, 1048,
 1076, 1586, 1631, 1633, 1653, 1693,
 1715, 1727, 1729, 1731, 1733, 1735,
 1737, 1739, 1741, 1744, 1752, 3406
`__kernel_backend_scope_begin:` ..
 82, 82, 128, 128, 173, 173, 184, 184,

226, 250, 265, 281, 298, 324, 338,
355, 368, 1378, 1563, 1581, 1585, 1760
`_kernel_backend_scope_begin:n` .
 184, 203, 212, 404, 432, 445
`_kernel_backend_scope_end:` ...
 82, 84, 128, 137,
 173, 175, 184, 193, 243, 261, 275,
 291, 318, 332, 348, 364, 376, 427,
 441, 460, 1379, 1575, 1582, 1588, 1774
`\g_kernel_backend_scope_int` ...
 182, 189, 191, 196, 200, 208, 210, 216
`\l_kernel_backend_scope_int` ...
 182, 188, 201, 207
`\g_kernel_clip_path_int`
 380, 1640, 1643, 1656, 1685, 1688, 1696
`_kernel_color_backend_stack-`
`init:Nnn` 470, 470, 3319
`_kernel_color_backend_stack-`
`pop:n` 484, 494, 542, 3352
`_kernel_color_backend_stack-`
`push:nn`
 .. 484, 484, 539, 984, 996, 3340, 3384
`_kernel_dependency_version-`
`check:Nn` 1
`_kernel_dependency_version-`
`check:nn` 27, 29
`_kernel_file_name_quote:n`
 1903, 1929
`_kernel_kern:n`
 2391, 2395, 2398, 2402,
 2765, 2773, 2776, 2792, 2890, 2892

L

lua commands:
`\lua_load_module:n` 1162

M

`\MessageBreak` 40
 mode commands:
`\mode_if_horizontal:TF` ... 3038, 3045
`\mode_if_math:TF` 2936
 msg commands:
`\msg_error:nnn` 556, 2139
`\msg_new:nnn` 558

O

`\oddsidemargin` 2960
 opacity internal commands:
`_opacity_backend:nn`
 3399, 3400, 3402, 3404, 3405
`_opacity_backend:nnn` 3256, 3258,
 3259, 3265, 3273, 3279, 3300, 3307
`_opacity_backend_fill:n`
 .. 3256, 3263, 3355, 3355, 3399, 3401

`_opacity_backend_fill_stroke:nn`
 .. 3355, 3357, 3363, 3367, 3390, 3395
`\l_opacity_backend_fill_tl`
 3325, 3331, 3364, 3372
`_opacity_backend_reset:`
 3329, 3343, 3345, 3387
`_opacity_backend_reset_fill:` ..
 3256, 3260, 3269, 3298
`_opacity_backend_reset_stroke:`
 3256, 3261, 3277, 3305
`_opacity_backend_select:n`
 3256, 3256, 3329,
 3329, 3370, 3390, 3394, 3399, 3399
`\c_opacity_backend_stack_int` ...
 3314, 3340, 3352, 3384
`_opacity_backend_stroke:n`
 .. 3256, 3271, 3355, 3361, 3399, 3403
`\l_opacity_backend_stroke_tl` ...
 3325, 3332, 3359, 3373

P

pdf commands:
`\pdf_object_if_exist:nTF` 850, 916, 934
`\pdf_object_new:n`
 841, 852, 896, 918, 936
`\pdf_object_ref:n`
 798, 865, 929, 944, 962, 967
`\pdf_object_ref_last:`
 818, 843, 846, 902
`\pdf_object_unnamed_write:nn` ...
 825, 872, 928, 943
`\pdf_object_write:nnn`
 842, 853, 897, 919, 937
 pdf internal commands:
`_pdf_backend:n`
 2682, 2682, 2684, 2686, 2688, 2702,
 2707, 2716, 2736, 2768, 2769, 2779
`_pdf_backend_annotation:nnnn` 3240
`_pdf_backend_annotation_last:` 3241
`_pdf_backend_bdc:nn` 2449, 2449,
 2676, 2676, 2813, 2813, 2838, 2838
`_pdf_backend_catalog_gput:nn` ..
 2274, 2274,
 2492, 2492, 2685, 2685, 2821, 2821
`_pdf_backend_compress_objects:n`
 2415, 2427,
 2597, 2608, 2794, 2796, 2832, 2833
`_pdf_backend_compresslevel:n` ..
 2415, 2415,
 2597, 2597, 2794, 2794, 2832, 2832
`_pdf_backend_destination:nn` ...
 2356, 2356,
 2455, 2455, 2734, 2734, 2819, 2819

_pdf_backend_destination:nnnn .	_pdf_backend_pageobject_ref:n .
..... 2356, 2382, 2354, 2354,
2455, 2478, 2734, 2756, 2819, 2820	2586, 2586, 2732, 2732, 2823, 2831
_pdf_backend_destination_-	_pdf_backend_pagesize_gset:nn .
aux:nnnn .. 2356, 2384, 2387, 2734, 2758, 2761	.. 2842, 2842, 2861, 2861, 2868, 2868
_pdf_backend_emc: .. 2449, 2451,	_pdf_backend_pdfmark:n
2676, 2678, 2813, 2815, 2838, 2839	2271, 2271, 2273, 2275, 2277, 2291,
_pdf_backend_info_gput:nn	2308, 2313, 2359, 2403, 2450, 2452
..... 2274, 2276,	_pdf_backend_version_major:...
2492, 2502, 2685, 2687, 2821, 2822	... 2441, 2447, 2447, 2653, 2653,
_pdf_backend_objcompresslevel:n	2803, 2804, 2811, 2811, 2836, 2836
..... 2597, 2611, 2612, 2614	_pdf_backend_version_major_-
_pdf_backend_object_id:n	gset:n
..... 2278, 2281,	2439, 2439,
2513, 2531, 2690, 2693, 2823, 2825	2625, 2625, 2801, 2801, 2834, 2834
_pdf_backend_object_int	_pdf_backend_version_minor:...
..... 2279, 2346, 2348,	... 2445, 2447, 2448, 2653, 2666,
2353, 2522, 2691, 2724, 2726, 2731	2808, 2809, 2811, 2812, 2836, 2837
_pdf_backend_object_last:	_pdf_backend_version_minor_-
..... 2352, 2352,	gset:n
2575, 2575, 2730, 2730, 2823, 2830	2439, 2443,
_pdf_backend_object_new:	2625, 2642, 2801, 2806, 2834, 2835
..... 2278, 2278,	_pdf_exp_not_i:nn
2513, 2513, 2690, 2690, 2823, 2823 2532, 2551, 2556, 2562
_pdf_backend_object_now:nn ...	_pdf_exp_not_ii:nn
2344, 2344, 2351, 2564, 2564, 2574, 2532, 2552, 2557, 2563
2722, 2722, 2729, 2823, 2828, 2829	pdf.baselineskip
_pdf_backend_object_prop	3784
..... 2512, 2689	pdf.bordertracking
_pdf_backend_object_ref:n	3542
2278, 2280, 2281, 2285, 2513, 2530,	pdf.bordertracking.begin
2690, 2692, 2693, 2697, 2823, 2824	3542
_pdf_backend_object_write:nn ..	pdf.bordertracking.continue
..... 2532, 2541, 2543, 2572, 2823	3542
_pdf_backend_object_write:nnn .	pdf.bordertracking.end
2282, 2282, 2288, 2532, 2532, 2561,	3542
2694, 2694, 2699, 2823, 2826, 2827	pdf.breaklink
_pdf_backend_object_write_-	3680
array:nn ... 2282, 2306, 2694, 2700	pdf.breaklink.write
_pdf_backend_object_write_-	3680
aux:nnn 2282, 2284, 2289, 2347	pdf.brokenlink.dict
_pdf_backend_object_write_-	3542
dict:nn 2282, 2311, 2694, 2705	pdf.brokenlink.rect
_pdf_backend_object_write_-	3542
fstream:nn . 2282, 2316, 2694, 2710	pdf.brokenlink.skip
_pdf_backend_object_write_-	3542
fstream:nnn	3680
2319, 2321	pdf.count
_pdf_backend_object_write_-	3680
stream:nn .. 2282, 2331, 2694, 2712	pdf.currentrect
_pdf_backend_object_write_-	3464
stream:nnn	pdf.cvs
2282, 2334, 2336	pdf.dest.anchor
_pdf_backend_object_write_-	3507
stream:nnnn . 2694, 2711, 2713, 2714	pdf.dest.point
	3507
	pdf.dest.x
	3507
	pdf.dest.y
	3507
	pdf.dest2device
	3507
	pdf.dev.x
	3507
	pdf.dev.y
	3507
	pdf.dvi.pt
	3464
	pdf.globaldict
	3461
	pdf.leftboundary
	3542
	pdf.linkdp.pad
	3468
	pdf.linkht.pad
	3468
	pdf.linkmargin
	3468
	pdf.llx
	3471
	pdf.lly
	3471
	pdf.originx
	3542

pdf.originy	3542	_pdfannot_backend_link_begin_-	
pdf.outerbox	3784	goto:nnw	2917, 2917,
pdf.pdfmark	3784		3114, 3114, 3194, 3194, 3230, 3230
pdf.pdfmark.dict	3784	_pdfannot_backend_link_begin_-	
pdf.pdfmark.good	3784	user:nnw	2917, 2922,
pdf.pt.dvi	3464		3114, 3116, 3194, 3199, 3230, 3231
pdf.rect	3471	\\g__pdfannot_backend_link_bool	..
pdf.rect.ht	3464		2912, 2926, 2931, 2946, 2984
pdf.rightboundary	3542	\\g__pdfannot_backend_link_dict_-	
pdf.save.linkll	3471	tl	2909, 2934, 2979
pdf.save.linkur	3471	_pdfannot_backend_link_end:	...
pdf.save.ll	3471		2917, 2944,
pdf.save.ur	3471		3114, 3129, 3194, 3218, 3230, 3233
pdf.tmpa	3507	_pdfannot_backend_link_end_-	
pdf.tmpb	3507	aux:	2917, 2947, 2949
pdf.tmpc	3507	\\g__pdfannot_backend_link_int	...
pdf.tmpd	3507		2908, 2974,
pdf.urx	3471		2978, 3078, 3193, 3204, 3210, 3221
pdf.ury	3471	_pdfannot_backend_link_last:	..
pdfannot internal commands:			3077, 3077,
_pdfannot_backend:n	3175, 3175,		3138, 3138, 3220, 3220, 3234, 3234
	3177, 3182, 3206, 3219, 3224, 3225	_pdfannot_backend_link_-	
\\l__pdfannot_backend_breaklink_-		margin:n	3079, 3079,
pdfmark_tl	2913, 2981, 3072		3149, 3149, 3222, 3222, 3235, 3235
_pdfannot_backend_breaklink_-		\\g__pdfannot_backend_link_math_-	
postscript:n	2915, 2915, 2965, 2967, 3073	bool	2911, 2937, 2938, 2941, 2951
_pdfannot_backend_breaklink_-		_pdfannot_backend_link_minima:	2917, 2955, 2986
usebox:N	2916, 2916, 2966, 3075	_pdfannot_backend_link_off:	...
\\l__pdfannot_backend_content_box			3086, 3087,
	2874,		3159, 3166, 3224, 3225, 3236, 3237
	2939, 2963, 2966, 2968, 2997, 3008	_pdfannot_backend_link_on:	...
_pdfannot_backend_generic:nnnn			3086, 3086,
	2877, 2877, 3090,		3159, 3159, 3224, 3224, 3236, 3236
	3090, 3179, 3179, 3228, 3228, 3240	_pdfannot_backend_link_-	
_pdfannot_backend_generic_-		outerbox:n	2917, 2957, 3015
aux:nnnn	2877, 2879, 2882	\\g__pdfannot_backend_link_sf_int	
\\g__pdfannot_backend_int			2910, 3036, 3047, 3048
	2876, 2895, 2899, 2907, 2973, 2974,	_pdfannot_backend_link_sf_-	
	3178, 3181, 3184, 3192, 3203, 3205	restore:	2917, 2940, 2983, 3043
_pdfannot_backend_last:		_pdfannot_backend_link_sf_-	
	2906, 2906, 3103,	save:	2917, 2935, 2953, 3034
	3103, 3191, 3191, 3229, 3229, 3241	\\l__pdfannot_backend_model_box	..
_pdfannot_backend_link:nw	2917		2875,
_pdfannot_backend_link_aux:nw	2917		2956, 2988, 2996, 3007, 3022, 3024
_pdfannot_backend_link_begin:n		pdfmanagement commands:	
	3194, 3196, 3200, 3201	\\pdfmanagement_add:nnn	...
_pdfannot_backend_link_-			815, 3322, 3333, 3374, 3377
begin:nnnw	3114, 3115, 3117, 3118, 3230, 3232	\\pdfmanagement_if_active_p:	...
_pdfannot_backend_link_-			810, 811, 3315, 3316, 3391, 3392
begin:nw	2919, 2923, 2924	peek commands:	
_pdfannot_backend_link_begin_-		\\peek_meaning:NTF	2180, 2183
aux:nw	2927, 2929	\\peek_remove_spaces:n	2178

<code>\tex_XeTeXpdfpagecount:D</code>	2125	526, 1629, 1801, 2909, 2913, 3325, 3326
<code>\tex_XeTeXpicfile:D</code>	2067	<code>\tl_set:Nn</code> . 527, 528, 537, 538, 983,
<code>TeXcolorseparation</code>	3458	995, 1808, 1826, 1920, 2914, 3072,
<code>\textwidth</code>	3023	3327, 3328, 3331, 3332, 3372, 3373
tl commands:		<code>\tl_to_str:n</code> 2172, 2194
<code>\c_space_tl</code>		<code>\tl_use:N</code> 745, 858
. 306, 311, 314, 567, 572, 610,		token commands:
713, 787, 997, 1625, 1788, 1789,		<code>\c_math_toggle_token</code> 2942, 2952
1790, 1791, 1974, 1975, 1976, 1977,		
2029, 2032, 2034, 2035, 2036, 2037,		U
2102, 2237, 2238, 2239, 2240, 2584,		use commands:
2595, 2979, 3112, 3147, 3184, 3211		<code>\use:N</code> 43, 2304, 2696, 2725
<code>\tl_clear:N</code> 1806, 1823,		<code>\use:n</code> 58, 813, 839,
1955, 1963, 2066, 2074, 2217, 2224		894, 1053, 1066, 1310, 1442, 1520,
<code>\tl_gclear:N</code> 1663, 1699		1532, 1544, 1704, 2096, 2169, 2191
<code>\tl_gset:Nn</code> 1622, 2934		<code>\use_none:n</code> 1721
<code>\tl_if_blank:nTF</code> 480, 565,		<code>\use_none:nnn</code> 3051
661, 678, 685, 703, 829, 911, 2101, 2176		
<code>\tl_if_empty:NTF</code> . 1625, 1810, 1862,		V
1873, 1999, 2003, 2030, 2048, 2089		<code>\value</code> 2959
<code>\tl_if_empty:nTF</code> 923, 1719		vbox commands:
<code>\tl_if_empty_p:N</code> 1856, 2044		<code>\vbox_set:Nn</code> 3058
<code>\tl_new:N</code> 525,		<code>\vbox_to_zero:n</code> 2389, 2396, 2763, 2774
		<code>\vbox_unpack_drop:N</code> 3066