

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}{2021-03-02}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2021-03-02}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2021-03-02}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2021-03-02}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2021-03-02}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2021-03-02}{ }
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:n
28 {
29   \_kernel_dependency_version_check:n {2020-09-01}
30 <dvipdfmx>   {l3backend-dvipdfmx.def}
31 <dvips>      {l3backend-dvips.def}
32 <dvisvgm>    {l3backend-dvisvgm.def}
33 <luatex>     {l3backend-luatex.def}
34 <pdftex>     {l3backend-pdftex.def}
35 <xetex>      {l3backend-xetex.def}
```

```

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

```

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

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

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

```

\__kernel_backend_literal:n
\__kernel_backend_literal:x
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} } }
49 \cs_generate_variant:Nn \__kernel_backend_literal:n { x }

```

(End definition for `__kernel_backend_literal:e`.)

1.1 dvips backend

```

50 <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.

```

51 \cs_new_protected:Npn \__kernel_backend_literal_postscript:n #1
52 { \__kernel_backend_literal:n { ps:: #1 } }
53 \cs_generate_variant:Nn \__kernel_backend_literal_postscript:n { x }

```

(End 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 `"`).

```

54 \cs_new_protected:Npn \__kernel_backend_postscript:n #1
55 { \__kernel_backend_literal:n { ps: SDict ~ begin ~ #1 ~ end } }
56 \cs_generate_variant:Nn \__kernel_backend_postscript:n { x }

```

(End 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.

```

57 \bool_if:NT \g__kernel_backend_header_bool
58 {

```

```

59 \cs_if_exist:NTF \AtBeginDvi
60 { \AtBeginDvi }
61 { \use:n }
62 { \__kernel_backend_literal:n { header = l3backend-dvips.pro } }
63 }

```

`__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.

```

64 \cs_new_protected:Npn \__kernel_backend_align_begin:
65 {
66   \__kernel_backend_literal:n { ps::[begin] }
67   \__kernel_backend_literal_postscript:n { currentpoint }
68   \__kernel_backend_literal_postscript:n { currentpoint~translate }
69 }
70 \cs_new_protected:Npn \__kernel_backend_align_end:
71 {
72   \__kernel_backend_literal_postscript:n { neg-exch-neg-exch~translate }
73   \__kernel_backend_literal:n { ps::[end] }
74 }

```

(End 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.

```

75 \cs_new_protected:Npn \__kernel_backend_scope_begin:
76 { \__kernel_backend_literal:n { ps:gsave } }
77 \cs_new_protected:Npn \__kernel_backend_scope_end:
78 { \__kernel_backend_literal:n { ps:grestore } }

```

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

```

79 </dvips>

```

1.2 LuaTeX and pdfTeX backends

```

80 <*luatex | pdftex>

```

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

`__kernel_backend_literal_pdf:n` 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).

`__kernel_backend_literal_pdf:x`

```

81 \cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1
82 {
83   <*luatex>
84   \tex_pdfextension:D literal
85   </luatex>
86   <*pdftex>

```

```

87     \tex_pdfliteral:D
88   </pdftex>
89     { \exp_not:n {#1} }
90   }
91   \cs_generate_variant:Nn \__kernel_backend_literal_pdf:n { x }

```

(End definition for __kernel_backend_literal_pdf:n.)

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

```

92   \cs_new_protected:Npn \__kernel_backend_literal_page:n #1
93   {
94     <*luatex>
95       \tex_pdfextension:D literal ~
96     </luatex>
97     <*pdftex>
98       \tex_pdfliteral:D
99     </pdftex>
100       page { \exp_not:n {#1} }
101   }

```

(End definition for __kernel_backend_literal_page:n.)

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

```

\__kernel_backend_scope_end:
102 \cs_new_protected:Npn \__kernel_backend_scope_begin:
103 {
104 <*luatex>
105   \tex_pdfextension:D save \scan_stop:
106 </luatex>
107 <*pdftex>
108   \tex_pdfsave:D
109 </pdftex>
110 }
111 \cs_new_protected:Npn \__kernel_backend_scope_end:
112 {
113 <*luatex>
114   \tex_pdfextension:D restore \scan_stop:
115 </luatex>
116 <*pdftex>
117   \tex_pdfrestore:D
118 </pdftex>
119 }

```

(End 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.

```

120 \cs_new_protected:Npn \__kernel_backend_matrix:n #1
121 {
122 <*luatex>
123   \tex_pdfextension:D setmatrix
124 </luatex>
125 <*pdftex>
126   \tex_pdfsetmatrix:D
127 </pdftex>

```

```

128         { \exp_not:n {#1} }
129     }
130 \cs_generate_variant:Nn \__kernel_backend_matrix:n { x }

(End definition for \__kernel_backend_matrix:n.)

131 </luatex | pdfTeX>

```

1.3 dvipdfmx backend

```

132 <*dvipdfmx | xetex>

```

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:x

```

```

133 \cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1
134 { \__kernel_backend_literal:n { pdf:literal~ #1 } }
135 \cs_generate_variant:Nn \__kernel_backend_literal_pdf:n { x }

```

(End 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!

```

136 \cs_new_protected:Npn \__kernel_backend_literal_page:n #1
137 { \__kernel_backend_literal:n { pdf:literal~direct~ #1 } }

```

(End 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 `xdvipfmx (x:)` as these are well-tested “in the wild”.

```

138 \cs_new_protected:Npn \__kernel_backend_scope_begin:
139 { \__kernel_backend_literal:n { x:gsave } }
140 \cs_new_protected:Npn \__kernel_backend_scope_end:
141 { \__kernel_backend_literal:n { x:grestore } }

```

(End definition for __kernel_backend_scope_begin: and __kernel_backend_scope_end:.)

```

142 <@@=sys>

```

```

\c__kernel_sys_dvipdfmx_version_int

```

A short excursion into the `sys` module to set up the backend version information.

```

143 \group_begin:
144 \cs_set:Npn \__sys_tmp:w #1 Version ~ #2 ~ #3 \q_stop {#2}
145 \sys_get_shell:nnTF { extractbb---version }
146 { \char_set_catcode_space:n { ‘\ } }
147 \l__sys_internal_tl
148 {
149     \int_const:Nn \c__kernel_sys_dvipdfmx_version_int
150     {
151         \exp_after:wN \__sys_tmp:w \l__sys_internal_tl
152         \q_stop
153     }
154 }
155 { \int_const:Nn \c__kernel_sys_dvipdfmx_version_int { 0 } }
156 \group_end:

```

(End definition for \c__kernel_sys_dvipdfmx_version_int.)

```
157 <@@=)
158 </dvipdfmx | xetex>
```

1.4 dvisvgm backend

```
159 <*dvisvgm>
```

```
\_kernel_backend_literal_svg:n
\_kernel_backend_literal_svg:x
```

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.

```
160 \cs_new_protected:Npn \_kernel_backend_literal_svg:n #1
161 { \_kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }
162 \cs_generate_variant:Nn \_kernel_backend_literal_svg:n { x }
```

(End 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.

```
163 \int_new:N \g__kernel_backend_scope_int
164 \int_new:N \l__kernel_backend_scope_int
```

(End 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:x
\_kernel_backend_scope:n
\_kernel_backend_scope:x
```

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.

```
165 \cs_new_protected:Npn \_kernel_backend_scope_begin:
166 {
167   \_kernel_backend_literal_svg:n { <g> }
168   \int_set_eq:NN
169     \l__kernel_backend_scope_int
170     \g__kernel_backend_scope_int
171   \group_begin:
172     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
173 }
174 \cs_new_protected:Npn \_kernel_backend_scope_end:
175 {
176   \prg_replicate:nn
177     { \g__kernel_backend_scope_int }
178     { \_kernel_backend_literal_svg:n { </g> } }
179   \group_end:
180   \int_gset_eq:NN
181     \g__kernel_backend_scope_int
182     \l__kernel_backend_scope_int
183 }
184 \cs_new_protected:Npn \_kernel_backend_scope_begin:n #1
185 {
186   \_kernel_backend_literal_svg:n { <g ~ #1 > }
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_generate_variant:Nn \__kernel_backend_scope_begin:n { x }
194   \cs_new_protected:Npn \__kernel_backend_scope:n #1
195   {
196     \__kernel_backend_literal_svg:n { <g ~ #1 > }
197     \int_gincr:N \g__kernel_backend_scope_int
198   }
199   \cs_generate_variant:Nn \__kernel_backend_scope:n { x }

```

(End definition for __kernel_backend_scope_begin: and others.)

```

200 </dvisvgm>
201 </package>

```

2 l3backend-box Implementation

```

202 <*package>
203 <@@=box>

```

2.1 dvips backend

```

204 <*dvips>

```

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

```

205   \cs_new_protected:Npn \__box_backend_clip:N #1
206   {
207     \__kernel_backend_scope_begin:
208     \__kernel_backend_align_begin:
209     \__kernel_backend_literal_postscript:n { matrix~currentmatrix }
210     \__kernel_backend_literal_postscript:n
211     { Resolution~72~div~VResolution~72~div~scale }
212     \__kernel_backend_literal_postscript:n { DVImag~dup~scale }
213     \__kernel_backend_literal_postscript:x
214     {
215       0 ~
216       \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
217       \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
218       \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
219       rectclip
220     }
221     \__kernel_backend_literal_postscript:n { setmatrix }
222     \__kernel_backend_align_end:
223     \hbox_overlap_right:n { \box_use:N #1 }
224     \__kernel_backend_scope_end:
225     \skip_horizontal:n { \box_wd:N #1 }
226   }

```

(End 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.

```

227 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
228 { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
229 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
230 {
231   \__kernel_backend_scope_begin:
232   \__kernel_backend_align_begin:
233   \__kernel_backend_literal_postscript:x
234   {
235     \fp_compare:nNnTF {#2} = \c_zero_fp
236     { 0 }
237     { \fp_eval:n { round ( -(#2) , 5 ) } } ~
238     rotate
239   }
240   \__kernel_backend_align_end:
241   \box_use:N #1
242   \__kernel_backend_scope_end:
243 }

```

(End 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.

```

244 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
245 {
246   \__kernel_backend_scope_begin:
247   \__kernel_backend_align_begin:
248   \__kernel_backend_literal_postscript:x
249   {
250     \fp_eval:n { round ( #2 , 5 ) } ~
251     \fp_eval:n { round ( #3 , 5 ) } ~
252     scale
253   }
254   \__kernel_backend_align_end:
255   \hbox_overlap_right:n { \box_use:N #1 }
256   \__kernel_backend_scope_end:
257 }

```

(End definition for `__box_backend_scale:Nnn`.)

258 `</dvips>`

2.2 LuaTeX and pdfTeX backends

259 `<*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.

```

260 \cs_new_protected:Npn \__box_backend_clip:N #1

```

```

261 {
262   \_kernel_backend_scope_begin:
263   \_kernel_backend_literal_pdf:x
264   {
265     0~
266     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
267     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
268     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
269     re~W~n
270   }
271   \hbox_overlap_right:n { \box_use:N #1 }
272   \_kernel_backend_scope_end:
273   \skip_horizontal:n { \box_wd:N #1 }
274 }

```

(End definition for _box_backend_clip:N.)

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

```

275 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
276 { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
277 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
278 {
279   \_kernel_backend_scope_begin:
280   \box_set_wd:Nn #1 { Opt }
281   \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
282   \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
283   { \fp_zero:N \l__box_backend_cos_fp }
284   \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
285   \_kernel_backend_matrix:x
286   {
287     \fp_use:N \l__box_backend_cos_fp \c_space_tl
288     \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp
289     { 0~0 }
290     {
291       \fp_use:N \l__box_backend_sin_fp
292       \c_space_tl
293       \fp_eval:n { -\l__box_backend_sin_fp }
294     }
295     \c_space_tl
296     \fp_use:N \l__box_backend_cos_fp
297   }
298   \box_use:N #1
299   \_kernel_backend_scope_end:
300 }
301 \fp_new:N \l__box_backend_cos_fp
302 \fp_new:N \l__box_backend_sin_fp

```

(End 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.

```

303 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
304 {
305   \__kernel_backend_scope_begin:
306   \__kernel_backend_matrix:x
307   {
308     \fp_eval:n { round ( #2 , 5 ) } ~
309     0~0~
310     \fp_eval:n { round ( #3 , 5 ) }
311   }
312   \hbox_overlap_right:n { \box_use:N #1 }
313   \__kernel_backend_scope_end:
314 }

```

(End definition for __box_backend_scale:Nnn.)

```

315 </luatex | pdftex>

```

2.3 dvipdfmx/X_YTeX backend

```

316 < *dvipdfmx | xetex>

```

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

```

317 \cs_new_protected:Npn \__box_backend_clip:N #1
318 {
319   \__kernel_backend_scope_begin:
320   \__kernel_backend_literal_pdf:x
321   {
322     0~
323     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
324     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
325     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
326     re~W~n
327   }
328   \hbox_overlap_right:n { \box_use:N #1 }
329   \__kernel_backend_scope_end:
330   \skip_horizontal:n { \box_wd:N #1 }
331 }

```

(End definition for __box_backend_clip:N.)

__box_backend_rotate: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.

```

332 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
333 { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
334 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
335 {
336   \__kernel_backend_scope_begin:
337   \__kernel_backend_literal:x
338   {
339     x:rotate~

```

```

340     \fp_compare:nNnTF {#2} = \c_zero_fp
341     { 0 }
342     { \fp_eval:n { round ( #2 , 5 ) } } }
343   }
344   \box_use:N #1
345   \__kernel_backend_scope_end:
346 }

```

(End 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.

```

347 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
348 {
349   \__kernel_backend_scope_begin:
350   \__kernel_backend_literal:x
351   {
352     x:scale~
353     \fp_eval:n { round ( #2 , 5 ) } ~
354     \fp_eval:n { round ( #3 , 5 ) }
355   }
356   \hbox_overlap_right:n { \box_use:N #1 }
357   \__kernel_backend_scope_end:
358 }

```

(End definition for `__box_backend_scale:Nnn`.)

359 `</dvipdfmx | xetex>`

2.4 dvisvgm backend

360 `<*dvisvgm>`

`__box_backend_clip:N`
`\g__box_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!

```

361 \cs_new_protected:Npn \__box_backend_clip:N #1
362 {
363   \int_gincr:N \g__box_clip_path_int
364   \__kernel_backend_literal_svg:x
365   { < clipPath-id = " l3cp \int_use:N \g__box_clip_path_int " > }
366   \__kernel_backend_literal_svg:x
367   {
368     <
369     path ~ d =
370     "
371     M ~ 0 ~
372     \dim_to_decimal:n { -\box_dp:N #1 } ~
373     L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
374     \dim_to_decimal:n { -\box_dp:N #1 } ~
375     L ~ \dim_to_decimal:n { \box_wd:N #1 } ~

```

```

376         \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
377         L ~ 0 ~
378         \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
379         Z
380         "
381     />
382 }
383 \__kernel_backend_literal_svg:n
384 { < /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.

```

385 \__kernel_backend_scope_begin:n
386 {
387     transform =
388     "
389         translate ( { ?x } , { ?y } ) ~
390         scale ( 1 , -1 )
391     "
392 }
393 \__kernel_backend_scope:x
394 {
395     clip-path =
396     "url ( \c_hash_str l3cp \int_use:N \g__box_clip_path_int ) "
397 }
398 \__kernel_backend_scope:n
399 {
400     transform =
401     "
402         scale ( -1 , 1 ) ~
403         translate ( { ?x } , { ?y } ) ~
404         scale ( -1 , -1 )
405     "
406 }
407 \box_use:N #1
408 \__kernel_backend_scope_end:
409 }
410 \int_new:N \g__box_clip_path_int

```

(End definition for $\backslash_box_backend_clip:N$ and $\backslash g_box_clip_path_int$.)

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

```

411 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
412 {
413     \__kernel_backend_scope_begin:x
414     {
415         transform =
416         "
417         rotate

```

```

418         ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
419     "
420 }
421 \box_use:N #1
422 \__kernel_backend_scope_end:
423 }

```

(End 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).

```

424 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
425 {
426     \__kernel_backend_scope_begin:x
427     {
428         transform =
429         "
430             translate ( { ?x } , { ?y } ) ~
431             scale
432             (
433                 \fp_eval:n { round ( -#2 , 5 ) } ,
434                 \fp_eval:n { round ( -#3 , 5 ) }
435             ) ~
436             translate ( { ?x } , { ?y } ) ~
437             scale ( -1 )
438         "
439     }
440     \hbox_overlap_right:n { \box_use:N #1 }
441     \__kernel_backend_scope_end:
442 }

```

(End definition for `__box_backend_scale:Nnn`.)

```

443 </dvisvgm>
444 </package>

```

3 l3backend-color Implementation

```

445 <*package>
446 <@@=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 Collecting information from L^AT_EX 2_ε

3.1.1 dvips-style

```

447 <*dvisvgm | dvipdfmx | dvips | xetex>

```

`_color_backend_pickup:N` Allow for L^AT_EX 2_ε color. Here, the possible input values are limited: dvips-style colors can mainly be taken as-is with the exception spot ones (here we need a model and a tint).
`_color_backend_pickup:w` The x-type expansion is there to cover the case where xcolor is in use.

```

448 \cs_new_protected:Npn \\_color_backend_pickup:N #1 { }
449 \cs_if_exist:cT { ver@color.sty }
450 {
451   \cs_set_protected:Npn \\_color_backend_pickup:N #1
452   {
453     \exp_args:NW \tl_if_head_is_space:nTF \current@color
454     {
455       \tl_set:Nx #1
456       {
457         { \exp_after:wN \use:n \current@color }
458         { 1 }
459       }
460     }
461     {
462       \exp_last_unbraced:Nx \\_color_backend_pickup:w
463       { \current@color } \s__color_stop #1
464     }
465   }
466   \cs_new_protected:Npn \\_color_backend_pickup:w #1 ~ #2 \s__color_stop #3
467   { \tl_set:Nn #3 { {#1} {#2} } }
468 }

```

(End definition for `_color_backend_pickup:N` and `_color_backend_pickup:w`.)

```

469 </divisvgm | dvipdfmx | dvips | xetex>

```

3.1.2 Lua_TE_X and pdf_TE_X

```

470 < *luatex | pdftex>

```

`_color_backend_pickup:N` The current color in driver-dependent format: pick up the package-mode data if available. We end up converting back and forward in this route as we store our color data in dvips format. The `\current@color` needs to be x-expanded before `_color_backend_pickup:w` breaks it apart, because for instance xcolor sets it to be instructions to generate a color

```

471 \cs_new_protected:Npn \\_color_backend_pickup:N #1 { }
472 \cs_if_exist:cT { ver@color.sty }
473 {
474   \cs_set_protected:Npn \\_color_backend_pickup:N #1
475   {
476     \exp_last_unbraced:Nx \\_color_backend_pickup:w
477     { \current@color } ~ 0 ~ 0 ~ 0 \s__color_stop #1
478   }
479   \cs_new_protected:Npn \\_color_backend_pickup:w
480   #1 ~ #2 ~ #3 ~ #4 ~ #5 ~ #6 \s__color_stop #7
481   {
482     \str_if_eq:nnTF {#2} { g }
483     { \tl_set:Nn #7 { { gray } {#1} } }
484     {
485       \str_if_eq:nnTF {#4} { rg }
486       { \tl_set:Nn #7 { { rgb } { #1 ~ #2 ~ #3 } } }

```

```

487         {
488             \str_if_eq:nnTF {#5} { k }
489             { \tl_set:Nn #7 { { cmyk } { #1 ~ #2 ~ #3 ~ #4 } } }
490             {
491                 \str_if_eq:nnTF {#2} { cs }
492                 {
493                     \tl_set:Nx #7 { { \use:n #1 } { #5 } }
494                 }
495                 {
496                     \tl_set:Nn #7 { { gray } { 0 } }
497                 }
498             }
499         }
500     }
501 }
502 }

```

(End definition for `__color_backend_pickup:N` and `__color_backend_pickup:w`.)

```
503 </luatex | pdfTeX>
```

3.2 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. The exact form depends on the engine, and for `dvipdfmx/XYTeX` the backend version.

3.2.1 Common code

```
504 <*dvipdfmx | luatex | pdfTeX | xetex>
```

`\l__color_backend_stack_int` pdf_YTeX, Lua_YTeX and recent (x)dvipdfmx have multiple stacks available, and to track which one is in use a variable is required.

```
505 \int_new:N \l__color_backend_stack_int
```

(End definition for `\l__color_backend_stack_int`.)

```
506 </dvipdfmx | luatex | pdfTeX | xetex>
```

3.2.2 dvipdfmx/X_YTeX

```
507 <*dvipdfmx | xetex>
```

`__kernel_color_backend_stack_init:Nnn` In (x)dvipdfmx, the base color stack is not set up, so we have to force that, as well as providing a mechanism more generally.

```

\g__color_backend_stack_int
\c__color_backend_main_stack_int
508 \int_compare:nNnTF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
509 { \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3 { } }
510 {
511     \int_new:N \g__color_backend_stack_int
512     \cs_new_protected:Npx \__kernel_color_backend_stack_init:Nnn #1#2#3
513     {
514         \int_gincr:N \exp_not:N \g__color_backend_stack_int
515         \int_const:Nn #1 { \exp_not:N \g__color_backend_stack_int }
516         \use:x
517         {
518             \cs_if_exist:NTF \AtBeginDvi

```

```

519         { \exp_not:N \AtBeginDvi }
520         { \exp_not:N \use:n }
521         {
522             \__kernel_backend_literal:n
523             {
524                 pdfcolorstackinit ~
525                 \exp_not:N \int_use:N \exp_not:N \g__color_backend_stack_int
526                 \c_space_tl
527                 \exp_not:N \tl_if_blank:nF {#2} { #2 ~ }
528                 (#3)
529             }
530         }
531     }
532 }
533 \cs_if_exist:cTF { main@pdfcolorstack }
534 {
535     \int_set:Nn \l__color_backend_stack_int
536     { \int_use:c { main@pdfcolorstack } }
537 }
538 {
539     \__kernel_color_backend_stack_init:Nnn \c__color_backend_main_stack_int
540     { page ~ direct } { 0 ~ g ~ 0 ~ G }
541     \int_set_eq:NN \l__color_backend_stack_int
542     \c__color_backend_main_stack_int
543     \int_const:cn { main@pdfcolorstack } { \c__color_backend_main_stack_int }
544 }
545 }

```

(End definition for __kernel_color_backend_stack_init:Nnn, \g__color_backend_stack_int, and \c__color_backend_main_stack_int.)

__kernel_color_backend_stack_push:nn
__kernel_color_backend_stack_push:nx
__kernel_color_backend_stack_pop:n

Simple enough but needs a version check.

```

546 \int_compare:nNnF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
547 {
548     \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
549     {
550         \__kernel_backend_literal:x
551         {
552             pdfcolorstack ~
553             \int_eval:n {#1} ~
554             push ~ (#2)
555         }
556     }
557 \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
558 \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
559 {
560     \__kernel_backend_literal:x
561     {
562         pdfcolorstack ~
563         \int_eval:n {#1} ~
564         pop
565     }
566 }
567 }

```

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

568 `</dvipdfmx | xetex>`

3.2.3 Lua_T_EX and pdf_T_EX

569 `<*luatex | pdftex>`

`_kernel_color_backend_stack_init:Nnn`

```
570 \cs_new_protected:Npn \_kernel_color_backend_stack_init:Nnn #1#2#3
571 {
572   \int_const:Nn #1
573   {
574     <*luatex>
575     \tex_pdffeedback:D colorstackinit ~
576     </luatex>
577     <*pdftex>
578     \tex_pdfcolorstackinit:D
579     </pdftex>
580     \tl_if_blank:nF {#2} { #2 ~ }
581     {#3}
582   }
583 }
```

(End definition for `_kernel_color_backend_stack_init:Nnn`.)

`_kernel_color_backend_stack_push:nn`

`_kernel_color_backend_stack_push:nx`

`_kernel_color_backend_stack_pop:n`

```
584 \cs_new_protected:Npn \_kernel_color_backend_stack_push:nn #1#2
585 {
586   <*luatex>
587   \tex_pdfextension:D colorstack ~
588   </luatex>
589   <*pdftex>
590   \tex_pdfcolorstack:D
591   </pdftex>
592   \int_eval:n {#1} ~ push ~ {#2}
593 }
594 \cs_generate_variant:Nn \_kernel_color_backend_stack_push:nn { nx }
595 \cs_new_protected:Npn \_kernel_color_backend_stack_pop:n #1
596 {
597   <*luatex>
598   \tex_pdfextension:D colorstack ~
599   </luatex>
600   <*pdftex>
601   \tex_pdfcolorstack:D
602   </pdftex>
603   \int_eval:n {#1} ~ pop \scan_stop:
604 }
```

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

605 `</luatex | pdftex>`

3.3 General color

3.3.1 dvips-style

606 `<*dvips | dvisvgm>`

Push the data to the stack. In the case of dvips also saves the drawing color in raw PostScript.

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc
607 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
608 { \__color_backend_select:n { cmyk ~ #1 } }
609 \cs_new_protected:Npn \__color_backend_select_gray:n #1
610 { \__color_backend_select:n { gray ~ #1 } }
611 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
612 { \__color_backend_select:n { rgb ~ #1 } }
613 \cs_new_protected:Npn \__color_backend_select:n #1
614 {
615   \__kernel_backend_literal:n { color~push~ #1 }
616   <*dvips>
617   \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
618   </dvips>
619   \group_insert_after:N \__color_backend_reset:
620 }
621 \cs_new_protected:Npn \__color_backend_reset:
622 { \__kernel_backend_literal:n { color~pop } }
```

(End definition for __color_backend_select_cmyk:n and others. This function is documented on page ??.)

623 `</dvips | dvisvgm>`

3.3.2 LuaTeX and pdfTeX

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

```

\l__color_backend_fill_tl
\l__color_backend_stroke_tl
625 \tl_new:N \l__color_backend_fill_tl
626 \tl_new:N \l__color_backend_stroke_tl
```

(End 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.
627 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
628 { \__color_backend_select:nn { #1 ~ k } { #1 ~ K } }
629 \cs_new_protected:Npn \__color_backend_select_gray:n #1
630 { \__color_backend_select:nn { #1 ~ g } { #1 ~ G } }
631 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
632 { \__color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
633 \cs_new_protected:Npn \__color_backend_select:nn #1#2
634 {
635   \tl_set:Nn \l__color_backend_fill_tl {#1}
636   \tl_set:Nn \l__color_backend_stroke_tl {#2}
637   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int { #1 ~ #2 }
638   \group_insert_after:N \__color_backend_reset:
639 }
640 \cs_new_protected:Npn \__color_backend_reset:
641 { \__kernel_color_backend_stack_pop:n \l__color_backend_stack_int }
```

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

```
642 </dvipdfmx | luatex | pdftex | xetex>
```

3.3.3 dvipdfmx/X₃TeX

```
643 <*dvipdfmx | xetex>
```

These backends have the most possible approaches: it recognises both dvips-based color specials and it's 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. Thus it is used in preference to the dvips-style interface or the “native” color specials (which have only one stack).

Push the data to the stack.

```
\_color_backend_select_cmyk:n
\_color_backend_select_gray:n
\_color_backend_select_rgb:n
\_color_backend_reset:
644 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
645 {
646   \cs_gset_protected:Npn \_color_backend_select_cmyk:n #1
647   {
648     \__kernel_backend_literal:n { pdf: bc ~ [#1] }
649     \group_insert_after:N \_color_backend_reset:
650   }
651   \cs_gset_eq:NN \_color_backend_select_gray:n \_color_backend_select_cmyk:n
652   \cs_gset_eq:NN \_color_backend_select_rgb:n \_color_backend_select_cmyk:n
653   \cs_gset_protected:Npn \_color_backend_reset:
654   { \__kernel_backend_literal:n { pdf: ec } }
655 }
```

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

```
656 </dvipdfmx | xetex>
```

3.4 Separations

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

```
657 <*dvips>
```

```
\_color_backend_select_separation:nn
\_color_backend_select_devicen:nn
658 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
659 { \_color_backend_select:n { separation ~ #1 ~ #2 } }
660 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
```

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

Initialising here means creating a small header set up plus massaging some data. This comes about as we have to deal with PDF-focussed data, which makes most sense “higher-up”. The approach is based on ideas from <https://tex.stackexchange.com/q/560093> plus using the PostScript manual for other aspects.

```
\_color_backend_separation_init:nnnnn
\_color_backend_separation_init:nxxnn
\_color_backend_separation_init_aux:nnnnn
\_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
661 \cs_new_protected:Npx \_color_backend_separation_init:nnnnn #1#2#3#4#5
662 {
663   \bool_if:NT \g__kernel_backend_header_bool
664   {
665     \cs_if_exist:NTF \AtBeginDvi
666     { \exp_not:N \AtBeginDvi }

```

```

667         { \use:n }
668         {
669             \exp_not:N \__color_backend_separation_init_aux:nnnnn
670             {#1} {#2} {#3} {#4} {#5}
671         }
672     }
673 }
674 \cs_generate_variant:Nn \__color_backend_separation_init:nnnnn { nxx }
675 \cs_new_protected:Npn \__color_backend_separation_init_aux:nnnnn #1#2#3#4#5
676 {
677     \__kernel_backend_literal:e
678     {
679         !
680         TeXDict ~ begin ~
681         /color \int_use:N \g__color_model_int
682         {
683             [ ~
684             /Separation ~ ( \str_convert_pdfname:n {#1} ) ~
685             [ ~ #2 ~ ] ~
686             {
687                 \cs_if_exist_use:cF { __color_backend_separation_init_ #2 :nnn }
688                 { \__color_backend_separation_init:nnn }
689                 {#3} {#4} {#5}
690             }
691             ] ~ setcolorspace
692         } ~ def ~
693     end
694 }
695 }
696 \cs_new:cpn { __color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
697 { \__color_backend_separation_init_Device:Nn 4 {#3} }
698 \cs_new:cpn { __color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
699 { \__color_backend_separation_init_Device:Nn 1 {#3} }
700 \cs_new:cpn { __color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
701 { \__color_backend_separation_init_Device:Nn 2 {#3} }
702 \cs_new:Npn \__color_backend_separation_init_Device:Nn #1#2
703 {
704     #2 ~
705     \prg_replicate:nn {#1}
706     { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
707     \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
708 }

```

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.

```

709 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3
710 {
711     \exp_args:Ne \__color_backend_separation_init:nnnn
712     { \__color_backend_separation_init_count:n {#2} }
713     {#1} {#2} {#3}
714 }
715 \cs_new:Npn \__color_backend_separation_init_count:n #1

```

```

716 { \int_eval:n { 0 \_color_backend_separation_init_count:w #1 ~ \s\_color_stop } }
717 \cs_new:Npn \_color_backend_separation_init_count:w #1 ~ #2 \s\_color_stop
718 {
719   +1
720   \tl_if_blank:nF {#2}
721   { \_color_backend_separation_init_count:w #2 \s\_color_stop }
722 }

```

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

```

723 \cs_new:Npn \_color_backend_separation_init:nnnn #1#2#3#4
724 {
725   \_color_backend_separation_init:w #3 ~ \s\_color_stop #4 ~ \s\_color_stop
726   \prg_replicate:nn {#1}
727   {
728     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
729     \int_eval:n { 3 * #1 } ~ index ~ mul ~
730     2 ~ index ~ add ~
731     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
732   }
733   \int_step_function:nnnN {#1} { -1 } { 1 }
734   \_color_backend_separation_init:n
735   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
736   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
737   \tl_if_blank:nF {#2}
738   { \_color_backend_separation_init:nw {#1} #2 ~ \s\_color_stop }
739 }
740 \cs_new:Npn \_color_backend_separation_init:w
741 #1 ~ #2 \s\_color_stop #3 ~ #4 \s\_color_stop
742 {
743   #1 ~ #3 ~ 0 ~
744   \tl_if_blank:nF {#2}
745   { \_color_backend_separation_init:w #2 \s\_color_stop #4 \s\_color_stop }
746 }
747 \cs_new:Npn \_color_backend_separation_init:n #1
748 { \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.

```

749 \cs_new:Npn \_color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s\_color_stop
750 {
751   #2 ~ #3 ~
752   2 ~ index ~ 2 ~ index ~ lt ~
753   { ~ pop ~ exch ~ pop ~ } ~

```

```

754     { ~
755       2 ~ index ~ 1 ~ index ~ gt ~
756       { ~ exch ~ pop ~ exch ~ pop ~ } ~
757       { ~ pop ~ pop ~ } ~
758     }
759   }
760   ifelse ~
761   #1 ~ 1 ~ roll ~
762   \tl_if_blank:nF {#4}
763   { \_color_backend_separation_init:nw {#1} #4 \s_color_stop }
764 }

```

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.

```

765 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
766 {
767   \_color_backend_separation_init:nxxxnn
768   {#2}
769   {
770     /CIEBasedABC ~
771     << ~
772     /RangeABC ~ [ ~ \c_color_model_range_CIELAB_tl \c_space_tl ] ~
773     /DecodeABC ~
774     [ ~
775       { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
776       { ~ 500 ~ div ~ } ~ bind ~
777       { ~ 200 ~ div ~ } ~ bind ~
778     ] ~
779     /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
780     /DecodeLMN ~
781     [ ~
782     { ~
783       dup ~ 6 ~ 29 ~ div ~ ge ~
784       { ~ dup ~ dup ~ mul ~ mul ~ } ~
785       { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
786     }
787     ifelse ~
788     0.9505 ~ mul ~
789   } ~ bind ~
790   { ~
791     dup ~ 6 ~ 29 ~ div ~ ge ~
792     { ~ dup ~ dup ~ mul ~ mul ~ } ~
793     { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
794   }
795   ifelse ~
796   } ~ bind ~
797   { ~
798     dup ~ 6 ~ 29 ~ div ~ ge ~
799     { ~ dup ~ dup ~ mul ~ mul ~ } ~
800     { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
801   }
802   ifelse ~
803   1.0890 ~ mul ~
804   } ~ bind
805 ] ~
806 /WhitePoint ~
807 [ ~ \tl_use:c { c_color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~

```

```

805         >>
806     }
807     { \c__color_model_range_CIELAB_t1 }
808     { 100 ~ 0 ~ 0 }
809     {#3}
810 }

```

(End 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.

```

811 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
812 {
813     \_kernel_backend_literal:e
814     {
815         !
816         TeXDict ~ begin ~
817         /color \int_use:N \g__color_model_int
818         {
819             [ ~
820                 /DeviceN ~
821                 [ ~ #1 ~ ] ~
822                 #2 ~
823                 { ~ #3 ~ } ~
824             ] ~ setcolorspace
825         } ~ def ~
826     end
827 }
828 }

```

(End definition for _color_backend_devicen_init:nnn.)

```

829 </dvips>
830 <*dvisvgm>

```

_color_backend_select_separation:nn No support at present.

```

\_color_backend_select_devicen:nn
831 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2 { }
832 \cs_new_protected:Npn \_color_backend_select_devicen:nn #1#2 { }

```

(End 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
833 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }
834 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnnnnn #1#2#3 { }

```

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

```

835 </dvisvgm>
836 <*dvipdfmx | luatex | pdftex | xetex>

```

_color_backend_select_separation:nn Although (x)dvipdfmx has a built-in approach to color spaces, that can't be used with the generic color stacks. So we take an approach in which we share the same code as for pdfTeX.

```

\_color_backend_select_devicen:nn
837 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
838 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
839 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn

```

(End definition for `_color_backend_select_separation:nn` and `_color_backend_select_device:nn`.)

`_color_backend_separation_init:nnnn`
`_color_backend_separation_init:n`
`_color_backend_separation_init_CIELAB:nnn`

Initialising the PDF structures needs two parts: creating an object containing the “real” name of the Separation, then adding a reference to that to each page. We use a separate object for the tint transformation following the model in the PDF reference.

```

840 \cs_new_protected:Npn \_color_backend_separation_init:nnnn #1#2#3#4#5
841 {
842   \pdf_object_unnamed_write:nx { dict }
843   {
844     /FunctionType ~ 2
845     /Domain ~ [ 0 ~ 1]
846     \tl_if_blank:nF {#3} { /Range ~ [#3] }
847     /C0 ~ [#4] ~
848     /C1 ~ [#5] /N ~ 1
849   }
850   \_color_backend_separation_init:n
851   {
852     /Separation ~
853     / \str_convert_pdfname:n {#1} ~ #2 ~
854     \pdf_object_ref_last:
855   }
856   \cs_if_exist:NT \pdfmanagement_add:nnn
857   {
858     \use:x
859     {
860       \pdfmanagement_add:nnn
861       { Page / Resources / ColorSpace }
862       { color \int_use:N \g__color_model_int }
863       { \pdf_object_ref_last: }
864     }
865   }
866 }
867 \cs_new_protected:Npn \_color_backend_separation_init:n #1
868 {
869   \pdf_object_unnamed_write:nx { array } {#1}
870 }

```

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

```

871 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
872 {
873   \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
874   {
875     \pdf_object_new:nn { __color_illuminant_CIELAB_ #1 } { array }
876     \pdf_object_write:nx { __color_illuminant_CIELAB_ #1 }
877     {
878       /Lab ~
879       <<
880       /WhitePoint ~
881       [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
882       /Range ~ [ \c__color_model_range_CIELAB_tl ]
883     }
884   }
885 }

```

```

886     \_color_backend_separation_init:nnnnn
887     {#2}
888     { \pdf_object_ref:n { \_color_illuminant_CIELAB_ #1 } }
889     { \c\_color_model_range_CIELAB_t1 }
890     { 100 ~ 0 ~ 0 }
891     {#3}
892 }
893 \cs_if_exist:NF \pdf_object_unnamed_write:nn
894 {
895     \cs_gset_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
896     { }
897 }

```

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

_color_backend_devicen_init:nnn
_color_backend_devicen_init:w
_color_backend_devicen_init:n

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

```

898 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
899 {
900     \pdf_object_unnamed_write:nx { stream }
901     {
902         {
903             /FunctionType ~ 4 ~
904             /Domain ~
905             [ ~
906                 \prg_replicate:nn
907                 { 0 \_color_backend_devicen_init:w #1 ~ \s_color_stop }
908                 { 0 ~ 1 ~ } ~
909             ] ~
910             /Range ~
911             [ ~
912                 \str_case:nn {#2}
913                 {
914                     { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
915                     { /DeviceGray } { 0 ~ 1 }
916                     { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
917                 } ~
918             ]
919         }
920         {#3}
921     }
922     \_color_backend_separation_init:n
923     {
924         /DeviceN ~
925         [ ~ #1 ~ ] ~
926         #2 ~
927         \pdf_object_ref_last:
928     }
929     \cs_if_exist:NT \pdfmanagement_add:nnn
930     {
931         \use:x
932         {
933             \pdfmanagement_add:nnn

```

```

934         { Page / Resources / ColorSpace }
935         { color \int_use:N \g__color_model_int }
936         { \pdf_object_ref_last: }
937     }
938 }
939 }
940 \cs_new:Npn \__color_backend_devicen_init:w #1 ~ #2 \s__color_stop
941 {
942     + 1
943     \tl_if_blank:nF {#2}
944     { \__color_backend_devicen_init:w #2 \s__color_stop }
945 }
946 \cs_new_eq:NN \__color_backend_devicen_init:n \__color_backend_separation_init:n
(End definition for \__color_backend_devicen_init:nnn, \__color_backend_devicen_init:w, and \_
_color_backend_devicen_init:n.)
947 </dvipdfmx | luatex | pdftex | xetex>
948 <*dvipdfmx | xetex>

```

__color_backend_select_separation:nn
__color_backend_select_devicen:nn

For older (x)dvipdfmx, we *could* support separations using a dedicated mechanism, but it was not added that long before the color stacks. So instead of having two complex paths, just disable here.

```

949 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
950 {
951     \cs_gset_protected:Npn \__color_backend_select_separation:nn #1#2 { }
952     \cs_gset_eq:NN \__color_backend_select_devicen:nn
953         \__color_backend_select_separation:nn
954 }
(End definition for \__color_backend_select_separation:nn and \__color_backend_select_devicen:nn.)
955 </dvipdfmx | xetex>

```

3.5 Fill and stroke color

Here, dvipdfmx/X_YTeX follows LuaTeX and pdfTeX, while for dvips we have to manage fill and stroke color ourselves. We also handle dvisvgm independently, as there we can create SVG directly.

```

956 <*dvipdfmx | luatex | pdftex | xetex>

```

__color_backend_fill_cmyk:n
__color_backend_fill_gray:n
__color_backend_fill_rgb:n
__color_backend_fill:n
__color_backend_stroke_cmyk:n
__color_backend_stroke_gray:n
__color_backend_stroke_rgb:n
__color_backend_stroke:n

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

```

957 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
958 { \__color_backend_fill:n { #1 ~ k } }
959 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
960 { \__color_backend_fill:n { #1 ~ g } }
961 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
962 { \__color_backend_fill:n { #1 ~ rg } }
963 \cs_new_protected:Npn \__color_backend_fill:n #1
964 {
965     \tl_set:Nn \l__color_backend_fill_tl {#1}

```

```

966 \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
967 { #1 ~ \l__color_backend_stroke_tl }
968 \group_insert_after:N \__color_backend_reset:
969 }
970 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
971 { \__color_backend_stroke:n { #1 ~ K } }
972 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
973 { \__color_backend_stroke:n { #1 ~ G } }
974 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
975 { \__color_backend_stroke:n { #1 ~ RG } }
976 \cs_new_protected:Npn \__color_backend_stroke:n #1
977 {
978 \tl_set:Nn \l__color_backend_stroke_tl {#1}
979 \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
980 { \l__color_backend_fill_tl \c_space_tl #1 }
981 \group_insert_after:N \__color_backend_reset:
982 }

```

(End 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
983 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
984 { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
985 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
986 { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
987 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
988 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

989 </dvipdfmx | luatex | pdftex | xetex>
990 <*dvipdfmx | xetex>

```

Deal with older (x)dvipdfmx.

```

991 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
992 {
993 \cs_gset_protected:Npn \__color_backend_fill_cmyk:n #1
994 {
995 \__kernel_backend_literal:n { pdf: bc ~ [#1] }
996 \group_insert_after:N \__color_backend_reset:
997 }
998 \cs_gset_eq:NN \__color_backend_fill_gray:n \__color_backend_fill_cmyk:n
999 \cs_gset_eq:NN \__color_backend_fill_rgb:n \__color_backend_fill_cmyk:n
1000 \cs_gset_protected:Npn \__color_backend_reset:
1001 { \__kernel_backend_literal:n { pdf: ec } }
1002 \cs_gset_protected:Npn \__color_backend_stroke:n #1
1003 { \__kernel_backend_literal:n {#1} }
1004 \cs_gset_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1005 \cs_gset_eq:NN \__color_backend_fill_devicen:nn
1006 \__color_backend_fill_separation:nn
1007 \cs_gset_eq:NN \__color_backend_stroke_separation:nn
1008 \__color_backend_fill_separation:nn
1009 \cs_gset_eq:NN \__color_backend_stroke_devicen:nn
1010 \__color_backend_stroke_separation:nn
1011 }

```

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

1012 `</dviptdpmx | xetex>`

1013 `<*dvips>`

`_color_backend_fill_cmyk:n` Fill color here is the same as general color *except* we skip the stroke part.

```

1014 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1015 { \_color_backend_fill:n { cmyk ~ #1 } }
1016 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1017 { \_color_backend_fill:n { gray ~ #1 } }
1018 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1019 { \_color_backend_fill:n { rgb ~ #1 } }
1020 \cs_new_protected:Npn \_color_backend_fill:n #1
1021 {
1022   \_kernel_backend_literal:n { color~push~ #1 }
1023   \group_insert_after:N \_color_backend_reset:
1024 }
1025 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1026 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1027 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1028 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1029 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1030 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }

```

(End 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
1031 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1032 { \_color_backend_fill:n { separation ~ #1 ~ #2 } }
1033 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1034 { \_kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1035 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1036 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

1037 `</dvips>`

1038 `<*dvisvgm>`

`_color_backend_fill_cmyk:n` Fill color here is the same as general color *except* we skip the stroke part.

```

1039 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1040 { \_color_backend_fill:n { cmyk ~ #1 } }
1041 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1042 { \_color_backend_fill:n { gray ~ #1 } }
1043 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1044 { \_color_backend_fill:n { rgb ~ #1 } }
1045 \cs_new_protected:Npn \_color_backend_fill:n #1
1046 {
1047   \_kernel_backend_literal:n { color~push~ #1 }
1048   \group_insert_after:N \_color_backend_reset:
1049 }

```

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

For drawings in SVG, we use scopes for all stroke colors. That requires using RGB values, which luckily are easy to convert here (cmyk to RGB is a fixed function).

```

1050 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1051 { \__color_backend_cmyk:w #1 \s__color_stop }
1052 \cs_new_protected:Npn \__color_backend_stroke_cmyk:w
1053 #1 ~ #2 ~ #3 ~ #4 \s__color_stop
1054 {
1055   \use:x
1056   {
1057     \__color_backend:nnn
1058     { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1059     { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1060     { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }
1061   }
1062 }
1063 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1064 {
1065   \use:x
1066   {
1067     \__color_backend_stroke_gray_aux:n
1068     { \fp_eval:n { 100 * (#1) } }
1069   }
1070 }
1071 \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1072 { \__color_backend:nnn {#1} {#1} {#1} }
1073 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1074 { \__color_backend_rgb:w #1 \s__color_stop }
1075 \cs_new_protected:Npn \__color_backend_stroke_rgb:w
1076 #1 ~ #2 ~ #3 \s__color_stop
1077 {
1078   \use:x
1079   {
1080     \__color_backend:nnn
1081     { \fp_eval:n { 100 * (#1) } }
1082     { \fp_eval:n { 100 * (#2) } }
1083     { \fp_eval:n { 100 * (#3) } }
1084   }
1085 }
1086 \cs_new_protected:Npx \__color_backend:nnn #1#2#3
1087 {
1088   \__kernel_backend_scope:n
1089   {
1090     stroke =
1091     "
1092     rgb
1093     (
1094       #1 \c_percent_str ,
1095       #2 \c_percent_str ,
1096       #3 \c_percent_str
1097     )
1098     "
1099   }
1100 }

```

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

At present, these are no-ops.

```

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

(End definition for \_color_backend_fill_separation:nn and others.)

1105 </dvisvgm>
1106 </package>

```

4 l3backend-draw Implementation

```

1107 <*package>
1108 <@@=draw>

```

4.1 dvips backend

```

1109 <*dvips>

```

The same as literal PostScript: same arguments about positioning apply her.

```

1110 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
1111 \cs_generate_variant:Nn \_draw_backend_literal:n { x }

```

(End definition for `_draw_backend_literal:n`.)

The `ps::[begin]` special here deals with positioning but allows us to continue on to a matching `ps::[end]`: contrast with `ps:`, which positions but where we can't split material between separate calls. The `@beginspecial/@endspecial` pair are from `special.pro` and correct the scale and y -axis direction. In contrast to `pgf`, we don't save the current point: discussion with Tom Rokici suggested a better way to handle the necessary translations (see `_draw_backend_box_use:Nnnnn`). (Note that `@beginspecial/@endspecial` forms a backend scope.) The `[begin]/[end]` lines are handled differently from the rest as they are conceptually different: not really drawing literals but instructions to `dvips` itself.

```

1112 \cs_new_protected:Npn \_draw_backend_begin:
1113 {
1114   \_kernel_backend_literal:n { ps::[begin] }
1115   \_draw_backend_literal:n { @beginspecial }
1116 }
1117 \cs_new_protected:Npn \_draw_backend_end:
1118 {
1119   \_draw_backend_literal:n { @endspecial }
1120   \_kernel_backend_literal:n { ps::[end] }
1121 }

```

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

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

```

1122 \cs_new_protected:Npn \_draw_backend_scope_begin:
1123 { \_draw_backend_literal:n { save } }
1124 \cs_new_protected:Npn \_draw_backend_scope_end:
1125 { \_draw_backend_literal:n { restore } }

```

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

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

```

1126 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1127 {
1128   \_draw_backend_literal:x
1129   {
1130     \dim_to_decimal_in_bp:n {#1} ~
1131     \dim_to_decimal_in_bp:n {#2} ~ moveto
1132   }
1133 }
1134 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1135 {
1136   \_draw_backend_literal:x
1137   {
1138     \dim_to_decimal_in_bp:n {#1} ~
1139     \dim_to_decimal_in_bp:n {#2} ~ lineto
1140   }
1141 }
1142 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1143 {
1144   \_draw_backend_literal:x
1145   {
1146     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1147     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1148     moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1149   }
1150 }
1151 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1152 {
1153   \_draw_backend_literal:x
1154   {
1155     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1156     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1157     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1158     curveto
1159   }
1160 }

```

(End 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`

```

1161 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1162 { \bool_gset_true:N \g__draw_draw_eor_bool }
1163 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1164 { \bool_gset_false:N \g__draw_draw_eor_bool }
1165 \bool_new:N \g__draw_draw_eor_bool

```

(End 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`

```

1166 \cs_new_protected:Npn \__draw_backend_closepath:
1167 { \__draw_backend_literal:n { closepath } }
1168 \cs_new_protected:Npn \__draw_backend_stroke:
1169 {
1170   \__draw_backend_literal:n { gsave }
1171   \__draw_backend_literal:n { color.sc }
1172   \__draw_backend_literal:n { stroke }
1173   \__draw_backend_literal:n { grestore }
1174   \bool_if:NT \g__draw_draw_clip_bool
1175   {
1176     \__draw_backend_literal:x
1177     {
1178       \bool_if:NT \g__draw_draw_eor_bool { eo }
1179       clip
1180     }
1181   }
1182   \__draw_backend_literal:n { newpath }
1183   \bool_gset_false:N \g__draw_draw_clip_bool
1184 }
1185 \cs_new_protected:Npn \__draw_backend_closestroke:
1186 {
1187   \__draw_backend_closepath:
1188   \__draw_backend_stroke:
1189 }
1190 \cs_new_protected:Npn \__draw_backend_fill:
1191 {
1192   \__draw_backend_literal:x
1193   {
1194     \bool_if:NT \g__draw_draw_eor_bool { eo }
1195     fill
1196   }
1197   \bool_if:NT \g__draw_draw_clip_bool
1198   {
1199     \__draw_backend_literal:x
1200     {
1201       \bool_if:NT \g__draw_draw_eor_bool { eo }
1202       clip
1203     }
1204   }
1205   \__draw_backend_literal:n { newpath }
1206   \bool_gset_false:N \g__draw_draw_clip_bool
1207 }
1208 \cs_new_protected:Npn \__draw_backend_fillstroke:
1209 {
1210   \__draw_backend_literal:x
1211   {
1212     \bool_if:NT \g__draw_draw_eor_bool { eo }

```

```

1213         fill
1214     }
1215     \__draw_backend_literal:n { gsave }
1216     \__draw_backend_literal:n { color.sc }
1217     \__draw_backend_literal:n { stroke }
1218     \__draw_backend_literal:n { grestore }
1219     \bool_if:NT \g__draw_draw_clip_bool
1220     {
1221         \__draw_backend_literal:x
1222         {
1223             \bool_if:NT \g__draw_draw_eor_bool { eo }
1224             clip
1225         }
1226     }
1227     \__draw_backend_literal:n { newpath }
1228     \bool_gset_false:N \g__draw_draw_clip_bool
1229 }
1230 \cs_new_protected:Npn \__draw_backend_clip:
1231 { \bool_gset_true:N \g__draw_draw_clip_bool }
1232 \bool_new:N \g__draw_draw_clip_bool
1233 \cs_new_protected:Npn \__draw_backend_discardpath:
1234 {
1235     \bool_if:NT \g__draw_draw_clip_bool
1236     {
1237         \__draw_backend_literal:x
1238         {
1239             \bool_if:NT \g__draw_draw_eor_bool { eo }
1240             clip
1241         }
1242     }
1243     \__draw_backend_literal:n { newpath }
1244     \bool_gset_false:N \g__draw_draw_clip_bool
1245 }

```

(End 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:
1246 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1247 {
1248     \__draw_backend_literal:x
1249     {
1250         [
1251             \exp_args:Nf \use:n
1252             { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1253         ] ~
1254         \dim_to_decimal_in_bp:n {#2} ~ setdash
1255     }
1256 }
1257 \cs_new:Npn \__draw_backend_dash:n #1
1258 { ~ \dim_to_decimal_in_bp:n {#1} }
1259 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1260 {
1261     \__draw_backend_literal:x
1262     { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }

```

```

1263 }
1264 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1265 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1266 \cs_new_protected:Npn \__draw_backend_cap_butt:
1267 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1268 \cs_new_protected:Npn \__draw_backend_cap_round:
1269 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1270 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1271 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1272 \cs_new_protected:Npn \__draw_backend_join_miter:
1273 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1274 \cs_new_protected:Npn \__draw_backend_join_round:
1275 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1276 \cs_new_protected:Npn \__draw_backend_join_bevel:
1277 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

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

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

```

1278 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1279 {
1280   \__draw_backend_literal:n
1281   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1282 }

```

(End definition for `__draw_backend_cm:nnnn`.)

`__draw_backend_box_use:Nnnnn` Inside a picture `@beginspecial/@endspecial` are active, which is normally a good thing but means that the position and scaling would be off if the box was inserted directly. To deal with that, there are a number of possible approaches. The implementation here was suggested by Tom Rokici (author of `dvips`). We end the current special placement, then set the current point with a literal `[begin]`. As for general literals, we then use the stack to store the current point and move to it. To insert the required transformation, we have to flip the y -axis, once before and once after it. Then we get back to the `TeX` reference point to insert our content. The clean up has to happen in the right places, hence the `[begin]/[end]` pair around `restore`. Finally, we can return to “normal” drawing mode. Notice that the set up here is very similar to that in `__draw_align_currentpoint_...`, but the ordering of saving and restoring is different (intermixed).

```

1283 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1284 {
1285   \__draw_backend_literal:n { @endspecial }
1286   \__draw_backend_literal:n { [end] }
1287   \__draw_backend_literal:n { [begin] }
1288   \__draw_backend_literal:n { save }
1289   \__draw_backend_literal:n { currentpoint }
1290   \__draw_backend_literal:n { currentpoint~translate }
1291   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1292   \__draw_backend_cm:nnnn { #2 } { #3 } { #4 } { #5 }
1293   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1294   \__draw_backend_literal:n { neg~exch~neg~exch~translate }

```

```

1295     \__draw_backend_literal:n { [end] }
1296     \hbox_overlap_right:n { \box_use:N #1 }
1297     \__draw_backend_literal:n { [begin] }
1298     \__draw_backend_literal:n { restore }
1299     \__draw_backend_literal:n { [end] }
1300     \__draw_backend_literal:n { [begin] }
1301     \__draw_backend_literal:n { @beginspecial }
1302 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1303 </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.

```

1304 < *dvipdfmx | luatex | pdftex | xetex >

```

4.2.1 Drawing

__draw_backend_literal:n Pass data through using a dedicated interface.

```

\__draw_backend_literal:x
1305 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1306 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for __draw_backend_literal:n.)

__draw_backend_begin: No special requirements here, so simply set up a drawing scope.

```

\__draw_backend_end:
1307 \cs_new_protected:Npn \__draw_backend_begin:
1308 { \__draw_backend_scope_begin: }
1309 \cs_new_protected:Npn \__draw_backend_end:
1310 { \__draw_backend_scope_end: }

```

(End definition for __draw_backend_begin: and __draw_backend_end:.)

__draw_backend_scope_begin: Use the backend-level scope mechanisms.

```

\__draw_backend_scope_end:
1311 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1312 \cs_new_eq:NN \__draw_backend_scope_end: \__kernel_backend_scope_end:

```

(End definition for __draw_backend_scope_begin: and __draw_backend_scope_end:.)

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

```

\__draw_backend_lineto:nn
\__draw_backend_moveto:nnnnnn
\__draw_backend_rectangle:nnnn
1313 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1314 {
1315     \__draw_backend_literal:x
1316     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1317 }
1318 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1319 {
1320     \__draw_backend_literal:x
1321     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1322 }
1323 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1324 {

```

```

1325 \__draw_backend_literal:x
1326 {
1327   \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1328   \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1329   \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1330   c
1331 }
1332 }
1333 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1334 {
1335   \__draw_backend_literal:x
1336   {
1337     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1338     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1339     re
1340   }
1341 }

```

(End definition for __draw_backend_moveto:nn and others.)

__draw_backend_evenodd_rule: The even-odd rule here can be implemented as a simply switch.

```

1342 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1343 { \bool_gset_true:N \g__draw_draw_eor_bool }
1344 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1345 { \bool_gset_false:N \g__draw_draw_eor_bool }
1346 \bool_new:N \g__draw_draw_eor_bool

```

(End 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.

```

1347 \cs_new_protected:Npn \__draw_backend_closepath:
1348 { \__draw_backend_literal:n { h } }
1349 \cs_new_protected:Npn \__draw_backend_stroke:
1350 { \__draw_backend_literal:n { S } }
1351 \cs_new_protected:Npn \__draw_backend_closestroke:
1352 { \__draw_backend_literal:n { s } }
1353 \cs_new_protected:Npn \__draw_backend_fill:
1354 {
1355   \__draw_backend_literal:x
1356   { f \bool_if:NT \g__draw_draw_eor_bool * }
1357 }
1358 \cs_new_protected:Npn \__draw_backend_fillstroke:
1359 {
1360   \__draw_backend_literal:x
1361   { B \bool_if:NT \g__draw_draw_eor_bool * }
1362 }
1363 \cs_new_protected:Npn \__draw_backend_clip:
1364 {
1365   \__draw_backend_literal:x
1366   { W \bool_if:NT \g__draw_draw_eor_bool * }
1367 }
1368 \cs_new_protected:Npn \__draw_backend_discardpath:
1369 { \__draw_backend_literal:n { n } }

```

(End 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_but:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1370 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1371 {
1372   \__draw_backend_literal:x
1373   {
1374     [
1375       \exp_args:Nf \use:n
1376       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1377     ] ~
1378     \dim_to_decimal_in_bp:n {#2} ~ d
1379   }
1380 }
1381 \cs_new:Npn \__draw_backend_dash:n #1
1382 { ~ \dim_to_decimal_in_bp:n {#1} }
1383 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1384 {
1385   \__draw_backend_literal:x
1386   { \dim_to_decimal_in_bp:n {#1} ~ w }
1387 }
1388 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1389 { \__draw_backend_literal:x { #1 ~ M } }
1390 \cs_new_protected:Npn \__draw_backend_cap_but:
1391 { \__draw_backend_literal:n { 0 ~ J } }
1392 \cs_new_protected:Npn \__draw_backend_cap_round:
1393 { \__draw_backend_literal:n { 1 ~ J } }
1394 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1395 { \__draw_backend_literal:n { 2 ~ J } }
1396 \cs_new_protected:Npn \__draw_backend_join_miter:
1397 { \__draw_backend_literal:n { 0 ~ j } }
1398 \cs_new_protected:Npn \__draw_backend_join_round:
1399 { \__draw_backend_literal:n { 1 ~ j } }
1400 \cs_new_protected:Npn \__draw_backend_join_bevel:
1401 { \__draw_backend_literal:n { 2 ~ j } }

```

(End definition for `_draw_backend_dash_pattern:nn` and others.)

```

\__draw_backend_cm:nnnn
\__draw_backend_cm_aux:nnnn

```

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!

```

1402 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1403 {
1404   <*luatex | pdftex>
1405   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1406   </luatex | pdftex>
1407   <*dvipdfmx | xetex>
1408   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1409   \__draw_backend_cm_aux:nnnn
1410   </dvipdfmx | xetex>

```

```

1411 }
1412 <*dvipdfmx | xetex>
1413 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1414 {
1415   \__kernel_backend_literal:x
1416   {
1417     x:rotate~
1418     \fp_compare:nNnTF {#1} = \c_zero_fp
1419     { 0 }
1420     { \fp_eval:n { round ( -#1 , 5 ) } }
1421   }
1422   \__kernel_backend_literal:x
1423   {
1424     x:scale~
1425     \fp_eval:n { round ( #2 , 5 ) } ~
1426     \fp_eval:n { round ( #3 , 5 ) }
1427   }
1428   \__kernel_backend_literal:x
1429   {
1430     x:rotate~
1431     \fp_compare:nNnTF {#4} = \c_zero_fp
1432     { 0 }
1433     { \fp_eval:n { round ( -#4 , 5 ) } }
1434   }
1435 }
1436 </dvipdfmx | xetex>

```

(End definition for __draw_backend_cm:nnnn and __draw_backend_cm_aux:nnnn.)

```

\__draw_backend_cm_decompose:nnnnN
\__draw_backend_cm_decompose_auxi:nnnnN
\__draw_backend_cm_decompose_auxii:nnnnN
\__draw_backend_cm_decompose_auxiii:nnnnN

```

Internally, transformations for drawing are tracked as a matrix. Not all engines provide a way of dealing with this: if we use a raw matrix, the engine loses track of positions (for example for hyperlinks), and this is not desirable. They do, however, allow us to track rotations and scalings. Luckily, we can decompose any (two-dimensional) matrix into two rotations and a single scaling:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} \cos \beta & \sin \beta \\ -\sin \beta & \cos \beta \end{bmatrix} \begin{bmatrix} w_1 & 0 \\ 0 & w_2 \end{bmatrix} \begin{bmatrix} \cos \gamma & \sin \gamma \\ -\sin \gamma & \cos \gamma \end{bmatrix}$$

The parent matrix can be converted to

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} E & H \\ -H & E \end{bmatrix} + \begin{bmatrix} F & G \\ G & -F \end{bmatrix}$$

From these, we can find that

$$\begin{aligned} \frac{w_1 + w_2}{2} &= \sqrt{E^2 + H^2} \\ \frac{w_1 - w_2}{2} &= \sqrt{F^2 + G^2} \\ \gamma - \beta &= \tan^{-1}(G/F) \\ \gamma + \beta &= \tan^{-1}(H/E) \end{aligned}$$

at which point we just have to do various pieces of re-arrangement to get all of the values. (See J. Blinn, *IEEE Comput. Graph. Appl.*, 1996, **16**, 82–88.) There is one wrinkle: the

PostScript (and PDF) way of specifying a transformation matrix exchanges where one would normally expect B and C to be.

```

1437 <*dvipdfmx|xetex>
1438 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1439 {
1440   \use:x
1441   {
1442     \__draw_backend_cm_decompose_auxi:nnnnN
1443     { \fp_eval:n { (#1 + #4) / 2 } }
1444     { \fp_eval:n { (#1 - #4) / 2 } }
1445     { \fp_eval:n { (#3 + #2) / 2 } }
1446     { \fp_eval:n { (#3 - #2) / 2 } }
1447   }
1448   #5
1449 }
1450 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1451 {
1452   \use:x
1453   {
1454     \__draw_backend_cm_decompose_auxii:nnnnN
1455     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1456     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1457     { \fp_eval:n { atand ( #3 , #2 ) } }
1458     { \fp_eval:n { atand ( #4 , #1 ) } }
1459   }
1460   #5
1461 }
1462 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1463 {
1464   \use:x
1465   {
1466     \__draw_backend_cm_decompose_auxiii:nnnnN
1467     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1468     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1469     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1470     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1471   }
1472   #5
1473 }
1474 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1475 {
1476   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1477   { #5 {#1} {#2} {#3} {#4} }
1478   { #5 {#1} {#3} {#2} {#4} }
1479 }
1480 </dvipdfmx|xetex>

```

(End definition for `__draw_backend_cm_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.

```

1481 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1482 {
1483   \__kernel_backend_scope_begin:
1484   <*luatex | pdftex>
1485   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1486   </luatex | pdftex>
1487   <*dvi.pdfmx | xetex>
1488   \__kernel_backend_literal:n
1489   { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1490   </dvi.pdfmx | xetex>
1491   \hbox_overlap_right:n { \box_use:N #1 }
1492   <*dvi.pdfmx | xetex>
1493   \__kernel_backend_literal:n { pdf:etrans }
1494   </dvi.pdfmx | xetex>
1495   \__kernel_backend_scope_end:
1496 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1497 </dvi.pdfmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1498 <*dvisvgm>

```

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

```

\__draw_backend_literal:x 1499 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1500 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for __draw_backend_literal:n.)

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

```

1501 \cs_new_protected:Npn \__draw_backend_begin:
1502 {
1503   \__kernel_backend_scope_begin:
1504   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1505 }
1506 \cs_new_eq:NN \__draw_backend_end: \__kernel_backend_scope_end:

```

(End definition for __draw_backend_begin: and __draw_backend_end:.)

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

```

\__draw_backend_lineto:nn 1507 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
\__draw_backend_rectangle:nnnnn 1508 {
\__draw_backend_curveto:nnnnnn 1509   \__draw_backend_add_to_path:n
\__draw_backend_add_to_path:n 1510   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1511 }
1512 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1513 {
1514   \__draw_backend_add_to_path:n

```

```

1515     { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1516   }
1517   \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1518   {
1519     \__draw_backend_add_to_path:n
1520     {
1521       M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1522       h ~ \dim_to_decimal:n {#3} ~
1523       v ~ \dim_to_decimal:n {#4} ~
1524       h ~ \dim_to_decimal:n { -#3 } ~
1525       Z
1526     }
1527   }
1528   \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1529   {
1530     \__draw_backend_add_to_path:n
1531     {
1532       C ~
1533       \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1534       \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1535       \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1536     }
1537   }
1538   \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1539   {
1540     \tl_gset:Nx \g__draw_draw_path_tl
1541     {
1542       \g__draw_draw_path_tl
1543       \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1544       #1
1545     }
1546   }
1547   \tl_new:N \g__draw_draw_path_tl

```

(End 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: 1548 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1549   { \__draw_backend_scope:n { fill-rule="evenodd" } }
1550 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1551   { \__draw_backend_scope:n { fill-rule="nonzero" } }

```

(End definition for __draw_backend_evenodd_rule: and __draw_backend_nonzero_rule:.)

```

\__draw_backend_path:n Setting fill and stroke effects and doing clipping all has to be done using scopes. This
\__draw_backend_closepath: means setting up the various requirements in a shared auxiliary which deals with the
\__draw_backend_stroke: bits and pieces. Clipping paths are reused for path drawing; not essential but avoids
\__draw_backend_closestroke: constructing them twice. Discarding a path needs a separate function as it's not quite
\__draw_backend_fill: the same.
\__draw_backend_fillstroke: 1552 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_clip: 1553   { \__draw_backend_add_to_path:n { Z } }
\__draw_backend_discardpath: 1554 \cs_new_protected:Npn \__draw_backend_path:n #1
\g__draw_draw_clip_bool 1555   {
\g__draw_draw_path_int 1556   \bool_if:NTF \g__draw_draw_clip_bool

```

```

1557 {
1558     \int_gincr:N \g__draw_clip_path_int
1559     \__draw_backend_literal:x
1560     {
1561         < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1562         { ?nl }
1563         <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1564         < /clipPath > { ? nl }
1565         <
1566             use~xlink:href =
1567                 "\c_hash_str l3path \int_use:N \g__draw_path_int " ~
1568                 #1
1569         />
1570     }
1571     \__draw_backend_scope:x
1572     {
1573         clip-path =
1574         "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1575     }
1576 }
1577 {
1578     \__draw_backend_literal:x
1579     { <path ~ d=" \g__draw_draw_path_tl " ~ #1 /> }
1580 }
1581 \tl_gclear:N \g__draw_draw_path_tl
1582 \bool_gset_false:N \g__draw_draw_clip_bool
1583 }
1584 \int_new:N \g__draw_path_int
1585 \cs_new_protected:Npn \__draw_backend_stroke:
1586 { \__draw_backend_path:n { style="fill:none" } }
1587 \cs_new_protected:Npn \__draw_backend_closestroke:
1588 {
1589     \__draw_backend_closepath:
1590     \__draw_backend_stroke:
1591 }
1592 \cs_new_protected:Npn \__draw_backend_fill:
1593 { \__draw_backend_path:n { style="stroke:none" } }
1594 \cs_new_protected:Npn \__draw_backend_fillstroke:
1595 { \__draw_backend_path:n { } }
1596 \cs_new_protected:Npn \__draw_backend_clip:
1597 { \bool_gset_true:N \g__draw_draw_clip_bool }
1598 \bool_new:N \g__draw_draw_clip_bool
1599 \cs_new_protected:Npn \__draw_backend_discardpath:
1600 {
1601     \bool_if:NT \g__draw_draw_clip_bool
1602     {
1603         \int_gincr:N \g__draw_clip_path_int
1604         \__draw_backend_literal:x
1605         {
1606             < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1607             { ?nl }
1608             <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1609             < /clipPath >
1610         }

```

```

1611     \__draw_backend_scope:x
1612     {
1613         clip-path =
1614         "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1615     }
1616 }
1617 \tl_gclear:N \g__draw_draw_path_tl
1618 \bool_gset_false:N \g__draw_draw_clip_bool
1619 }

```

(End definition for __draw_backend_path:n and others.)

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_dash_aux:nn
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_but:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1620 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1621 {
1622     \use:x
1623     {
1624         \__draw_backend_dash_aux:nn
1625         { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1626         { \dim_to_decimal:n {#2} }
1627     }
1628 }
1629 \cs_new:Npn \__draw_backend_dash:n #1
1630 { , \dim_to_decimal_in_bp:n {#1} }
1631 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1632 {
1633     \__draw_backend_scope:x
1634     {
1635         stroke-dasharray =
1636         "
1637         \tl_if_empty:oTF { \use_none:n #1 }
1638         { none }
1639         { \use_none:n #1 }
1640         " ~
1641         stroke-offset=" #2 "
1642     }
1643 }
1644 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1645 { \__draw_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1646 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1647 { \__draw_backend_scope:x { stroke-miterlimit=" #1 " } }
1648 \cs_new_protected:Npn \__draw_backend_cap_but:
1649 { \__draw_backend_scope:n { stroke-linecap="butt" } }
1650 \cs_new_protected:Npn \__draw_backend_cap_round:
1651 { \__draw_backend_scope:n { stroke-linecap="round" } }
1652 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1653 { \__draw_backend_scope:n { stroke-linecap="square" } }
1654 \cs_new_protected:Npn \__draw_backend_join_miter:
1655 { \__draw_backend_scope:n { stroke-linejoin="miter" } }
1656 \cs_new_protected:Npn \__draw_backend_join_round:
1657 { \__draw_backend_scope:n { stroke-linejoin="round" } }
1658 \cs_new_protected:Npn \__draw_backend_join_bevel:
1659 { \__draw_backend_scope:n { stroke-linejoin="bevel" } }

```

(End definition for `_draw_backend_dash_pattern:nn` and others.)

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

```

1660 \cs_new_protected:Npn \_draw_backend_cm:nnnn #1#2#3#4
1661 {
1662   \_draw_backend_scope:n
1663   {
1664     transform =
1665     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1666   }
1667 }

```

(End definition for `_draw_backend_cm:nnnn`.)

`_draw_backend_box_use:Nnnnn` No special savings can be made here: simply displace the box inside a scope. As there is nothing to re-box, just make the box passed of zero size.

```

1668 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1669 {
1670   \_kernel_backend_scope_begin:
1671   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1672   \_kernel_backend_literal_svg:n
1673   {
1674     < g~
1675     stroke="none"~
1676     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1677     >
1678   }
1679   \box_set_wd:Nn #1 { Opt }
1680   \box_set_ht:Nn #1 { Opt }
1681   \box_set_dp:Nn #1 { Opt }
1682   \box_use:N #1
1683   \_kernel_backend_literal_svg:n { </g> }
1684   \_kernel_backend_scope_end:
1685 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```

1686 </dvisvgm>
1687 </package>

```

5 l3backend-graphics Implementation

```

1688 <*package>
1689 <@=graphics>

```

5.1 dvips backend

```

1690 <*dvips>

```

`_graphics_backend_getbb_eps:n` Simply use the generic function.

```

1691 \cs_new_eq:NN \_graphics_backend_getbb_eps:n \graphics_read_bb:n

```

(End definition for `_graphics_backend_getbb_eps:n`.)

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

```

1692 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1693 {
1694   \__kernel_backend_literal:x
1695   {
1696     PSfile = #1 \c_space_tl
1697     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1698     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1699     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1700     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1701   }
1702 }
(End definition for \_graphics_backend_include_eps:n.)
1703 </dvips>

```

5.2 LuaTeX and pdfTeX backends

```

1704 <*luatex | pdftex>

```

`\l_graphics_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.

```

1705 \tl_new:N \l_graphics_graphics_attr_tl
(End definition for \l_graphics_graphics_attr_tl.)

```

`_graphics_backend_getbb_jpg:n`
`_graphics_backend_getbb_pdf:n`
`_graphics_backend_getbb_png:n`
`_graphics_backend_getbb_auxi:n`
`_graphics_backend_getbb_auxii:n` 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.

```

1706 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1707 {
1708   \int_zero:N \l_graphics_page_int
1709   \tl_clear:N \l_graphics_pagebox_tl
1710   \tl_set:Nx \l_graphics_graphics_attr_tl
1711   {
1712     \tl_if_empty:NF \l_graphics_decodearray_tl
1713     { :D \l_graphics_decodearray_tl }
1714     \bool_if:NT \l_graphics_interpolate_bool
1715     { :I }
1716   }
1717   \tl_clear:N \l_graphics_graphics_attr_tl
1718   \_graphics_backend_getbb_auxi:n {#1}
1719 }
1720 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1721 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1722 {
1723   \tl_clear:N \l_graphics_decodearray_tl
1724   \bool_set_false:N \l_graphics_interpolate_bool

```

```

1725 \tl_set:Nx \l__graphics_graphics_attr_tl
1726 {
1727   : \l_graphics_pagebox_tl
1728   \int_compare:nNnT \l_graphics_page_int > 1
1729   { :P \int_use:N \l_graphics_page_int }
1730 }
1731 \__graphics_backend_getbb_auxi:n {#1}
1732 }
1733 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1734 {
1735   \graphics_bb_restore:xF { #1 \l_graphics_graphics_attr_tl }
1736   { \__graphics_backend_getbb_auxii:n {#1} }
1737 }

```

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.

```

1738 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1739 {
1740   \tex_immediate:D \tex_pdfximage:D
1741   \bool_lazy_or:nnT
1742   { \l_graphics_interpolate_bool }
1743   { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1744   {
1745     attr ~
1746     {
1747       \tl_if_empty:NF \l_graphics_decodearray_tl
1748       { /Decode~[ \l_graphics_decodearray_tl ] }
1749       \bool_if:NT \l_graphics_interpolate_bool
1750       { /Interpolate~true }
1751     }
1752   }
1753   \int_compare:nNnT \l_graphics_page_int > 0
1754   { page ~ \int_use:N \l_graphics_page_int }
1755   \tl_if_empty:NF \l_graphics_pagebox_tl
1756   { \l_graphics_pagebox_tl }
1757   {#1}
1758   \hbox_set:Nn \l__graphics_internal_box
1759   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1760   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1761   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1762   \int_const:cn { c__graphics_graphics_ #1 \l_graphics_graphics_attr_tl _int }
1763   { \tex_the:D \tex_pdflastximage:D }
1764   \graphics_bb_save:x { #1 \l_graphics_graphics_attr_tl }
1765 }

```

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

```

\_graphics_backend_include_jpg: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.

```

1766 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1767 {
1768   \tex_pdfrefximage:D

```

```

1769     \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1770   }
1771   \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1772   \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

(End definition for \__graphics_backend_include_jpg:n, \__graphics_backend_include_pdf:n, and
\__graphics_backend_include_png:n.)

```

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

```

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

1773 \sys_if_shell:T
1774 {
1775   \str_new:N \l__graphics_backend_dir_str
1776   \str_new:N \l__graphics_backend_name_str
1777   \str_new:N \l__graphics_backend_ext_str
1778   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1779   {
1780     \file_parse_full_name:nNNN {#1}
1781     \l__graphics_backend_dir_str
1782     \l__graphics_backend_name_str
1783     \l__graphics_backend_ext_str
1784     \exp_args:Nx \__graphics_backend_getbb_eps:nn
1785     {
1786       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1787       -converted-to.pdf
1788     }
1789     {#1}
1790   }
1791   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1792   {
1793     \file_compare_timestamp:nNnT {#2} > {#1}
1794     {
1795       \sys_shell_now:n
1796       { repstopdf ~ #2 ~ #1 }
1797     }
1798     \tl_set:Nn \l__graphics_name_tl {#1}
1799     \__graphics_backend_getbb_pdf:n {#1}
1800   }
1801   \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1802   {
1803     \file_parse_full_name:nNNN {#1}
1804     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1805     \exp_args:Nx \__graphics_backend_include_pdf:n
1806     {
1807       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1808       -converted-to.pdf
1809     }
1810   }
1811 }

```

(End definition for __graphics_backend_getbb_eps:n and others.)

```

1812 </luatex | pdftex>

```

5.3 dvipdfmx backend

1813 $\langle *dvipdfmx | xetex \rangle$

$\backslash_graphics_backend_getbb_eps:n$
 $\backslash_graphics_backend_getbb_jpg:n$
 $\backslash_graphics_backend_getbb_pdf:n$
 $\backslash_graphics_backend_getbb_png:n$

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

```
1814 \cs_new_eq:NN \_graphics_backend_getbb_eps:n \graphics_read_bb:n
1815  $\langle *dvipdfmx \rangle$ 
1816 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1817 {
1818   \int_zero:N \l_graphics_page_int
1819   \tl_clear:N \l_graphics_pagebox_tl
1820   \graphics_extract_bb:n {#1}
1821 }
1822 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1823 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1824 {
1825   \tl_clear:N \l_graphics_decodearray_tl
1826   \bool_set_false:N \l_graphics_interpolate_bool
1827   \graphics_extract_bb:n {#1}
1828 }
1829  $\langle /dvipdfmx \rangle$ 
```

(End definition for $\backslash_graphics_backend_getbb_eps:n$ and others.)

$\backslash g_graphics_track_int$

Used to track the object number associated with each graphic.

```
1830 \int_new:N \g_graphics_track_int
```

(End definition for $\backslash g_graphics_track_int$.)

$\backslash_graphics_backend_include_eps:n$
 $\backslash_graphics_backend_include_jpg:n$
 $\backslash_graphics_backend_include_pdf:n$
 $\backslash_graphics_backend_include_png:n$
 $\backslash_graphics_backend_include_auxi:nn$
 $\backslash_graphics_backend_include_auxii:nnn$
 $\backslash_graphics_backend_include_auxiii:nnn$

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

```
1831 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1832 {
1833   \_kernel_backend_literal:x
1834   {
1835     PSfile = #1 \c_space_tl
1836     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1837     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1838     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1839     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1840   }
1841 }
1842 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1843 { \_graphics_backend_include_auxi:nn {#1} { image } }
1844 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1845  $\langle *dvipdfmx \rangle$ 
1846 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1847 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1848  $\langle /dvipdfmx \rangle$ 
```

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.

```

1849 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1850 {
1851   \__graphics_backend_include_auxii:xnn
1852   {
1853     \tl_if_empty:NF \l_graphics_pagebox_tl
1854     { : \l_graphics_pagebox_tl }
1855     \int_compare:nNnT \l_graphics_page_int > 1
1856     { :P \int_use:N \l_graphics_page_int }
1857     \tl_if_empty:NF \l_graphics_decodearray_tl
1858     { :D \l_graphics_decodearray_tl }
1859     \bool_if:NT \l_graphics_interpolate_bool
1860     { :I }
1861   }
1862   {#1} {#2}
1863 }
1864 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1865 {
1866   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1867   {
1868     \__kernel_backend_literal:x
1869     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1870   }
1871   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1872 }
1873 \cs_generate_variant:Nn \__graphics_backend_include_auxii:nnn { x }

```

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!

```

1874 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1875 {
1876   \int_gincr:N \g__graphics_track_int
1877   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1878   \__kernel_backend_literal:x
1879   {
1880     pdf:#3~
1881     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1882     \int_compare:nNnT \l_graphics_page_int > 1
1883     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1884     \tl_if_empty:NF \l_graphics_pagebox_tl
1885     {
1886       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1887       bbox ~
1888       \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1889       \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1890       \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1891       \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1892     }
1893     (#1)
1894     \bool_lazy_or:nnT
1895     { \l_graphics_interpolate_bool }
1896     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1897     {
1898       <<

```

```

1899         \tl_if_empty:NF \l_graphics_decodearray_tl
1900         { /Decode~[ \l_graphics_decodearray_tl ] }
1901         \bool_if:NT \l_graphics_interpolate_bool
1902         { /Interpolate~true> }
1903     >>
1904 }
1905 }
1906 }

```

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

```

1907 </dvipdfmx | xetex>

```

5.4 X_YTeX backend

```

1908 <*xetex>

```

5.4.1 Images

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

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
\__graphics_backend_getbb_auxi:nN
\__graphics_backend_getbb_auxii:nNn
\__graphics_backend_getbb_auxii:VnN
\__graphics_backend_getbb_auxiii:nNnn
\__graphics_backend_getbb_auxiv:nNnn
\__graphics_backend_getbb_auxiv:VnNnn
\__graphics_backend_getbb_auxv:nNnn
\__graphics_backend_getbb_auxv:VnNnn
\__graphics_backend_getbb_pagebox:w

```

```

1909 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1910 {
1911     \int_zero:N \l_graphics_page_int
1912     \tl_clear:N \l_graphics_pagebox_tl
1913     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1914 }
1915 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1916 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1917 {
1918     \tl_clear:N \l_graphics_decodearray_tl
1919     \bool_set_false:N \l_graphics_interpolate_bool
1920     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1921 }
1922 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
1923 {
1924     \int_compare:nNnTF \l_graphics_page_int > 1
1925     { \__graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
1926     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1927 }
1928 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nN #1#2#3
1929 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1930 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nN { V }
1931 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1932 {
1933     \tl_if_empty:NTF \l_graphics_pagebox_tl
1934     { \__graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1935     { \__graphics_backend_getbb_auxv:nNnn }
1936     {#1} #2 {#3} {#4}
1937 }
1938 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nNnn #1#2#3#4#5
1939 {
1940     \use:x

```

```

1941     {
1942         \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1943         { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1944     }
1945 }
1946 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1947 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1948 {
1949     \graphics_bb_restore:nF {#1#3}
1950     { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1951 }
1952 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1953 {
1954     \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1955     \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1956     \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1957     \graphics_bb_save:n {#1#3}
1958 }
1959 \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

__graphics_backend_include_pdf:n
__graphics_backend_include_bitmap_quote:w

For PDF graphics, properly supporting the pagebox concept in X_YTeX is best done using the `\tex_XeTeXpdf_file:D` primitive. The syntax here is the same as for the graphic measurement part, although we know at this stage that there must be some valid setting for `\l_graphics_pagebox_tl`.

```

1960 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1961 {
1962     \tex_XeTeXpdf_file:D
1963     \__graphics_backend_include_pdf_quote:w #1 "#1" \s__graphics_stop \c_space_tl
1964     \int_compare:nNnT \l_graphics_page_int > 0
1965         { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1966     \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1967 }
1968 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \s__graphics_stop
1969 { " #2 " }

```

(End definition for __graphics_backend_include_pdf:n and __graphics_backend_include_bitmap_quote:w.)

```

1970 </xetex>

```

5.5 dvisvgm backend

```

1971 <*dvisvgm>

```

__graphics_backend_getbb_eps:n Simply use the generic function.

```

1972 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n

```

(End definition for __graphics_backend_getbb_eps:n.)

__graphics_backend_getbb_png:n
__graphics_backend_getbb_jpg:n

These can be included by extracting the bounding box data.

```

1973 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1974 {
1975     \int_zero:N \l_graphics_page_int

```

```

1976 \tl_clear:N \l_graphics_pagebox_tl
1977 \graphics_extract_bb:n {#1}
1978 }
1979 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
(End definition for \__graphics_backend_getbb_png:n and \__graphics_backend_getbb_jpg:n.)

```

__graphics_backend_getbb_pdf:n Same as for dvipdfmx: use the generic function

```

1980 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1981 {
1982   \tl_clear:N \l_graphics_decodearray_tl
1983   \bool_set_false:N \l_graphics_interpolate_bool
1984   \graphics_extract_bb:n {#1}
1985 }
(End definition for \__graphics_backend_getbb_pdf:n.)

```

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

```

\__graphics_backend_include_pdf:n
\__graphics_backend_include:nn
1986 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1987 { __graphics_backend_include:nn { PSfile } {#1} }
1988 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1989 { __graphics_backend_include:nn { pdffile } {#1} }
1990 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
1991 {
1992   \__kernel_backend_literal:x
1993   {
1994     #1 = #2 \c_space_tl
1995     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1996     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1997     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1998     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1999   }
2000 }
(End definition for \__graphics_backend_include_eps:n, \__graphics_backend_include_pdf:n, and \__graphics_backend_include:nn.)

```

__graphics_backend_include_png:n The backend here has built-in support for basic graphic inclusion (see dvisvgm.def for a more complex approach, needed if clipping, *etc.*, is covered at the graphic backend level).

__graphics_backend_include_jpg:n The only issue is that #1 must be quote-corrected. The dvisvgm:img operation quotes the file name, but if it is already quoted (contains spaces) then we have an issue: we simply strip off any quotes as a result.

__graphics_backend_include_bitmap_quote:w

```

2001 \cs_new_protected:Npn \__graphics_backend_include_png:n #1
2002 {
2003   \__kernel_backend_literal:x
2004   {
2005     dvisvgm:img~
2006     \dim_to_decimal:n { \l_graphics_ury_dim } ~
2007     \dim_to_decimal:n { \l_graphics_ury_dim } ~
2008     \__graphics_backend_include_bitmap_quote:w #1 " #1 " \s__graphics_stop
2009   }
2010 }
2011 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_png:n
2012 \cs_new:Npn \__graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \s__graphics_stop
2013 { " #2 " }

```

(End definition for `_graphics_backend_include_png:n`, `_graphics_backend_include_jpg:n`, and `_graphics_backend_include_bitmap_quote:w`.)

2014 `</dvisvgm>`

2015 `</package>`

6 l3backend-pdf Implementation

2016 `<*package>`

2017 `<@=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 at various points.

6.1 Shared code

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

`\l__pdf_internal_box`

2018 `\box_new:N \l__pdf_internal_box`

(End definition for `\l__pdf_internal_box`.)

6.2 dvips backend

2019 `<*dvips>`

`__pdf_backend_pdfmark:n`

`__pdf_backend_pdfmark:x`

Used often enough it should be a separate function.

2020 `\cs_new_protected:Npn __pdf_backend_pdfmark:n #1`

2021 `{ __kernel_backend_postscript:n { mark #1 ~ pdfmark } }`

2022 `\cs_generate_variant:Nn __pdf_backend_pdfmark:n { x }`

(End definition for `__pdf_backend_pdfmark:n`.)

6.2.1 Catalogue entries

`__pdf_backend_catalog_gput:nn`

`__pdf_backend_info_gput:nn`

2023 `\cs_new_protected:Npn __pdf_backend_catalog_gput:nn #1#2`

2024 `{ __pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }`

2025 `\cs_new_protected:Npn __pdf_backend_info_gput:nn #1#2`

2026 `{ __pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }`

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

6.2.2 Objects

\g__pdf_backend_object_int
\g__pdf_backend_object_prop

For tracking objects to allow finalisation.

```
2027 \int_new:N \g__pdf_backend_object_int
2028 \prop_new:N \g__pdf_backend_object_prop
```

(End definition for \g__pdf_backend_object_int and \g__pdf_backend_object_prop.)

__pdf_backend_object_new:nn
__pdf_backend_object_ref:n

Tracking objects is similar to dvipdfmx.

```
2029 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2030 {
2031   \int_gincr:N \g__pdf_backend_object_int
2032   \int_const:cn
2033     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2034     { \g__pdf_backend_object_int }
2035   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2036 }
2037 \cs_new:Npn \__pdf_backend_object_ref:n #1
2038 { { pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } } }
```

(End definition for __pdf_backend_object_new:nn and __pdf_backend_object_ref:n.)

__pdf_backend_object_write:nn
__pdf_backend_object_write:nx
__pdf_backend_object_write_array:nn
__pdf_backend_object_write_dict:nn
__pdf_backend_object_write_fstream:nn
__pdf_backend_object_write_stream:nn
__pdf_backend_object_write_stream:nnn

This is where we choose the actual type: some work to get things right.

```
2039 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2040 {
2041   \__pdf_backend_pdfmark:x
2042   {
2043     /_objdef ~ \__pdf_backend_object_ref:n {#1}
2044     /type
2045     \str_case_e:nn
2046       { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2047       {
2048         { array } { /array }
2049         { dict } { /dict }
2050         { fstream } { /stream }
2051         { stream } { /stream }
2052       }
2053     /OBJ
2054   }
2055   \use:c
2056     { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2057     { \__pdf_backend_object_ref:n {#1} } {#2}
2058 }
2059 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2060 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2061 {
2062   \__pdf_backend_pdfmark:x
2063     { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2064 }
2065 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2066 {
2067   \__pdf_backend_pdfmark:x
2068     { #1 << \exp_not:n {#2} >> /PUT }
2069 }
2070 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
```

```

2071 {
2072   \exp_args:Nx
2073   \__pdf_backend_object_write_fstream:nnn {#1} #2
2074 }
2075 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2076 {
2077   \__kernel_backend_postscript:n
2078   {
2079     SDict ~ begin ~
2080     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2081     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2082     end
2083   }
2084 }
2085 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2086 {
2087   \exp_args:Nx
2088   \__pdf_backend_object_write_stream:nnn {#1} #2
2089 }
2090 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2091 {
2092   \__kernel_backend_postscript:n
2093   {
2094     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2095     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2096   }
2097 }

```

(End definition for __pdf_backend_object_write:nn and others.)

__pdf_backend_object_now:nn
__pdf_backend_object_now:nx

No anonymous objects, so things are done manually.

```

2098 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2099 {
2100   \int_gincr:N \g__pdf_backend_object_int
2101   \__pdf_backend_pdfmark:x
2102   {
2103     /_objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2104     /type
2105     \str_case:nn
2106       {#1}
2107       {
2108         { array } { /array }
2109         { dict } { /dict }
2110         { fstream } { /stream }
2111         { stream } { /stream }
2112       }
2113     /OBJ
2114   }
2115   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2116   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2117 }
2118 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for __pdf_backend_object_now:nn.)

`__pdf_backend_object_last:` Much like the annotation version.

```

2119 \cs_new:Npn \__pdf_backend_object_last:
2120 { { pdf.obj \int_use:N \g__pdf_backend_object_int } }

(End definition for \__pdf_backend_object_last:.)

```

`__pdf_backend_pageobject_ref:n` Page references are easy in dvips.

```

2121 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2122 { { Page #1 } }

(End definition for \__pdf_backend_pageobject_ref:n.)

```

6.2.3 Annotations

In dvips, annotations have to be constructed manually. As such, we need the object code above for some definitions.

`\l__pdf_backend_content_box` The content of an annotation.

```

2123 \box_new:N \l__pdf_backend_content_box

(End definition for \l__pdf_backend_content_box.)

```

`\l__pdf_backend_model_box` For creating model sizing for links.

```

2124 \box_new:N \l__pdf_backend_model_box

(End definition for \l__pdf_backend_model_box.)

```

`\g__pdf_backend_annotation_int` Needed as objects which are not annotations could be created.

```

2125 \int_new:N \g__pdf_backend_annotation_int

(End definition for \g__pdf_backend_annotation_int.)

```

`__pdf_backend_annotation:nnnn` Annotations are objects, but we track them separately. Notably, they are not in the object data lists. Here, to get the co-ordinates 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.

```

2126 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2127 {
2128   \exp_args:Nf \__pdf_backend_annotation_aux:nnnn
2129     { \dim_eval:n {#1} } {#2} {#3} {#4}
2130 }
2131 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2132 {
2133   \box_move_down:nn {#3}
2134   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2135   \box_move_up:nn {#2}
2136   {
2137     \hbox:n
2138     {
2139       \__kernel_kern:n {#1}
2140       \__kernel_backend_postscript:n { pdf.save.ur }
2141       \__kernel_kern:n { -#1 }
2142     }

```

```

2143     }
2144     \int_gincr:N \g__pdf_backend_object_int
2145     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2146     \__pdf_backend_pdfmark:x
2147     {
2148         /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2149         pdf.rect
2150         #4 ~
2151         /ANN
2152     }
2153 }

```

(End definition for __pdf_backend_annotation:nnnn.)

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

```

2154 \cs_new:Npn \__pdf_backend_annotation_last:
2155 { { pdf.obj \int_use:N \g__pdf_backend_annotation_int } }

```

(End definition for __pdf_backend_annotation_last:.)

\g__pdf_backend_link_int To track annotations which are links.

```

2156 \int_new:N \g__pdf_backend_link_int

```

(End definition for \g__pdf_backend_link_int.)

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

```

2157 \tl_new:N \g__pdf_backend_link_dict_tl

```

(End definition for \g__pdf_backend_link_dict_tl.)

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

```

2158 \int_new:N \g__pdf_backend_link_sf_int

```

(End definition for \g__pdf_backend_link_sf_int.)

\g__pdf_backend_link_math_bool Needed to save/restore math mode.

```

2159 \bool_new:N \g__pdf_backend_link_math_bool

```

(End definition for \g__pdf_backend_link_math_bool.)

\g__pdf_backend_link_bool Track link formation: we cannot nest at all.

```

2160 \bool_new:N \g__pdf_backend_link_bool

```

(End definition for \g__pdf_backend_link_bool.)

\l__pdf_breaklink_pdfmark_tl Swappable content for link breaking.

```

2161 \tl_new:N \l__pdf_breaklink_pdfmark_tl
2162 \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdfmark }

```

(End definition for \l__pdf_breaklink_pdfmark_tl.)

__pdf_breaklink_postscript:n To allow dropping material unless link breaking is active.

```

2163 \cs_new_protected:Npn \__pdf_breaklink_postscript:n #1 { }

```

(End definition for __pdf_breaklink_postscript:n.)

`_pdf_breaklink_usebox:N` Swappable box unpacking or use.

2164 `\cs_new_eq:NN _pdf_breaklink_usebox:N \box_use:N`

(End definition for `_pdf_breaklink_usebox:N`.)

Links are crated 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`.

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 format mode are still to re-examine.

```
pdf.linkdp.pad
pdf.linkht.pad
pdf.llx
pdf.lly
pdf.ury
pdf.link.dict
pdf.outerbox
pdf.baselineskip

2165 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2166 { \_pdf_backend_link_begin:nw { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2167 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2168 { \_pdf_backend_link_begin:nw {#1#2} }
2169 \cs_new_protected:Npn \_pdf_backend_link_begin:nw #1
2170 {
2171   \bool_if:NF \g__pdf_backend_link_bool
2172   { \_pdf_backend_link_begin_aux:nw {#1} }
2173 }
2174 \cs_new_protected:Npn \_pdf_backend_link_begin_aux:nw #1
2175 {
2176   \bool_gset_true:N \g__pdf_backend_link_bool
2177   \_kernel_backend_postscript:n
2178   { /pdf.link.dict ( #1 ) def }
2179   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2180   \_pdf_backend_link_sf_save:
2181   \mode_if_math:TF
2182   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2183   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2184   \hbox_set:Nw \l__pdf_backend_content_box
2185   \_pdf_backend_link_sf_restore:
2186   \bool_if:NT \g__pdf_backend_link_math_bool
2187   { \c_math_toggle_token }
2188 }
2189 \cs_new_protected:Npn \_pdf_backend_link_end:
2190 {
2191   \bool_if:NT \g__pdf_backend_link_bool
2192   { \_pdf_backend_link_end_aux: }
2193 }
2194 \cs_new_protected:Npn \_pdf_backend_link_end_aux:
2195 {
2196   \bool_if:NT \g__pdf_backend_link_math_bool
2197   { \c_math_toggle_token }
2198   \_pdf_backend_link_sf_save:
```

```

2199 \hbox_set_end:
2200 \__pdf_backend_link_minima:
2201 \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2202 \exp_args:Nx \__pdf_backend_link_outerbox:n
2203 {
2204   \int_if_odd:nTF { \value { page } }
2205     { \oddsidemargin }
2206     { \evensidemargin }
2207 }
2208 \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2209 { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2210 \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2211 \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2212 \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2213 \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2214 {
2215   \hbox:n
2216     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2217 }
2218 \int_gincr:N \g__pdf_backend_object_int
2219 \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2220 \__kernel_backend_postscript:x
2221 {
2222   mark
2223   /objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2224   \g__pdf_backend_link_dict_tl \c_space_tl
2225   pdf.rect
2226   /ANN ~ \l__pdf_breaklink_pdfmark_tl
2227 }
2228 \__pdf_backend_link_sf_restore:
2229 \bool_gset_false:N \g__pdf_backend_link_bool
2230 }
2231 \cs_new_protected:Npn \__pdf_backend_link_minima:
2232 {
2233   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2234   \__kernel_backend_postscript:x
2235   {
2236     /pdf.linkdp.pad ~
2237     \dim_to_decimal:n
2238     {
2239       \dim_max:nn
2240       {
2241         \box_dp:N \l__pdf_backend_model_box
2242         - \box_dp:N \l__pdf_backend_content_box
2243       }
2244       { Opt }
2245     } ~
2246     pdf.pt.dvi ~ def
2247     /pdf.linkht.pad ~
2248     \dim_to_decimal:n
2249     {
2250       \dim_max:nn
2251       {
2252         \box_ht:N \l__pdf_backend_model_box

```

```

2253         - \box_ht:N \l__pdf_backend_content_box
2254     }
2255     { Opt }
2256 } ~
2257 pdf.pt.dvi ~ def
2258 }
2259 }
2260 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2261 {
2262     \__kernel_backend_postscript:x
2263     {
2264         /pdf.outerbox
2265         [
2266             \dim_to_decimal:n {#1} ~
2267             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2268             \dim_to_decimal:n { #1 + \textwidth } ~
2269             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2270         ]
2271         [ exch { pdf.pt.dvi } forall ] def
2272         /pdf.baselineskip ~
2273         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2274         { pdf.pt.dvi ~ def }
2275         { pop ~ pop }
2276     ifelse
2277 }
2278 }
2279 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2280 {
2281     \int_gset:Nn \g__pdf_backend_link_sf_int
2282     {
2283         \mode_if_horizontal:TF
2284         { \tex_spacefactor:D }
2285         { 0 }
2286     }
2287 }
2288 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2289 {
2290     \mode_if_horizontal:T
2291     {
2292         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2293         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2294     }
2295 }

```

(End definition for `__pdf_backend_link_begin_goto:nw` and others. These functions are documented on page ??.)

`\@makecol@hook` Hooks to allow link breaking: something will be needed in format mode at some stage. At present this code is disabled as there is an open question about the name of the hook: to be resolved at the L^AT_EX 2_ε end.

```

2296 \use_none:n
2297 {
2298     \cs_if_exist:NT \@makecol@hook
2299     {

```

```

2300     \tl_put_right:Nn \@makecol@hook
2301     {
2302         \box_if_empty:NF \@cclv
2303         {
2304             \vbox_set:Nn \@cclv
2305             {
2306                 \__kernel_backend_postscript:n
2307                 {
2308                     pdf.globaldict /pdf.brokenlink.rect ~ known
2309                     { pdf.bordertracking.continue }
2310                     if
2311                 }
2312                 \vbox_unpack_drop:N \@cclv
2313                 \__kernel_backend_postscript:n
2314                 { pdf.bordertracking.endpage }
2315             }
2316         }
2317     }
2318     \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2319     \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2320     \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2321 }
2322 }

```

(End definition for \@makecol@hook. This function is documented on page ??.)

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

```

2323 \cs_new:Npn \__pdf_backend_link_last:
2324 { { pdf.obj \int_use:N \g__pdf_backend_link_int } }

```

(End definition for __pdf_backend_link_last:.)

__pdf_backend_link_margin:n Convert to big points and pass to PostScript.

```

2325 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2326 {
2327     \__kernel_backend_postscript:x
2328     {
2329         /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2330     }
2331 }

```

(End definition for __pdf_backend_link_margin:n.)

_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.

```

2332 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2333 {
2334     \__kernel_backend_postscript:n { pdf.dest.anchor }
2335     \__pdf_backend_pdfmark:x
2336     {
2337         /View
2338         [

```

```

2339     \str_case:nnF {#2}
2340     {
2341         { xyz }    { /XYZ ~ pdf.dest.point ~ null }
2342         { fit }    { /Fit }
2343         { fitb }   { /FitB }
2344         { fitbh }  { /FitBH ~ pdf.dest.y }
2345         { fitbv }  { /FitBV ~ pdf.dest.x }
2346         { fith }   { /FitH ~ pdf.dest.y }
2347         { fitv }   { /FitV ~ pdf.dest.x }
2348         { fitr }   { /Fit }
2349     }
2350     {
2351         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2352     }
2353 ]
2354 /Dest ( \exp_not:n {#1} ) cvn
2355 /DEST
2356 }
2357 }
2358 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2359 {
2360     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2361     { \dim_eval:n {#2} } {#1} {#3} {#4}
2362 }
2363 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2364 {
2365     \vbox_to_zero:n
2366     {
2367         \__kernel_kern:n {#4}
2368         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2369         \tex_vss:D
2370     }
2371     \__kernel_kern:n {#1}
2372     \vbox_to_zero:n
2373     {
2374         \__kernel_kern:n { -#3 }
2375         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2376         \tex_vss:D
2377     }
2378     \__kernel_kern:n { -#1 }
2379     \__pdf_backend_pdfmark:n
2380     {
2381         /View
2382         [
2383             /FitR ~
2384             pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2385             pdf.urx ~ pdf.ury ~ pdf.dest2device
2386         ]
2387         /Dest ( #2 ) cvn
2388         /DEST
2389     }
2390 }

```

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

6.2.4 Structure

_pdf_backend_compresslevel:n
_pdf_backend_compress_objects:n

Doable for the usual ps2pdf method.

```

2391 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2392 {
2393   \int_compare:nNnT {#1} = 0
2394   {
2395     \__kernel_backend_literal_postscript:n
2396     {
2397       /setdistillerparams ~ where
2398       { pop << /CompressPages ~ false >> setdistillerparams }
2399       if
2400     }
2401   }
2402 }
2403 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2404 {
2405   \bool_if:nF {#1}
2406   {
2407     \__kernel_backend_literal_postscript:n
2408     {
2409       /setdistillerparams ~ where
2410       { pop << /CompressStreams ~ false >> setdistillerparams }
2411       if
2412     }
2413   }
2414 }
```

(End definition for _pdf_backend_compresslevel:n and _pdf_backend_compress_objects:n.)

_pdf_backend_version_major_gset:n
_pdf_backend_version_minor_gset:n

Data not available!

```

2415 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
2416 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1 { }
```

(End definition for _pdf_backend_version_major_gset:n and _pdf_backend_version_minor_gset:n.)

_pdf_backend_version_major:
_pdf_backend_version_minor:

Data not available!

```

2417 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2418 \cs_new:Npn \_pdf_backend_version_minor: { -1 }
```

(End definition for _pdf_backend_version_major: and _pdf_backend_version_minor:.)

6.2.5 Marked content

_pdf_backend_bdc:nn
_pdf_backend_emc:

Simple wrappers.

```

2419 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2420 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2421 \cs_new_protected:Npn \_pdf_backend_emc:
2422 { \_pdf_backend_pdfmark:n { /EMC } }
```

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

```

2423 </dvips>
```

6.3 LuaTeX and pdfTeX backend

2424 `<*luatex | pdftex>`

6.3.1 Annotations

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

```
2425 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2426 {
2427   <*luatex>
2428     \tex_pdfextension:D annot ~
2429   </luatex>
2430   <*pdftex>
2431     \tex_pdfannot:D
2432   </pdftex>
2433     width ~ \dim_eval:n {#1} ~
2434     height ~ \dim_eval:n {#2} ~
2435     depth ~ \dim_eval:n {#3} ~
2436     {#4}
2437 }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

`_pdf_backend_annotation_last:` A tiny amount of extra data gets added here; we use x-type expansion to get the space in the right place and form. The “extra” space in the LuaTeX version is *required* as it is consumed in finding the end of the keyword.

```
2438 \cs_new:Npx \_pdf_backend_annotation_last:
2439 {
2440   \exp_not:N \int_value:w
2441   <*luatex>
2442     \exp_not:N \tex_pdffeedback:D lastannot ~
2443   </luatex>
2444   <*pdftex>
2445     \exp_not:N \tex_pdflastannot:D
2446   </pdftex>
2447   \c_space_tl 0 ~ R
2448 }
```

(End definition for `_pdf_backend_annotation_last:`.)

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

```
\_pdf_backend_link_begin_user:nnw 2449 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
\_pdf_backend_link_begin:nnnw 2450 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
\_pdf_backend_link_end: 2451 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2452 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2453 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2454 {
2455   <*luatex>
2456     \tex_pdfextension:D startlink ~
2457   </luatex>
2458   <*pdftex>
2459     \tex_pdfstartlink:D
2460   </pdftex>
2461     attr {#1}
2462     #2 {#3}
```

```

2463 }
2464 \cs_new_protected:Npn \__pdf_backend_link_end:
2465 {
2466   \*luatex
2467     \tex_pdfextension:D endlink \scan_stop:
2468   \*luatex
2469   \*pdftex
2470     \tex_pdfendlink:D
2471   \*pdftex
2472 }

```

(End definition for __pdf_backend_link_begin_goto:nnw and others.)

__pdf_backend_link_last: Formatted for direct use.

```

2473 \cs_new:Npx \__pdf_backend_link_last:
2474 {
2475   \exp_not:N \int_value:w
2476   \*luatex
2477     \exp_not:N \tex_pdffeedback:D lastlink ~
2478   \*luatex
2479   \*pdftex
2480     \exp_not:N \tex_pdflastlink:D
2481   \*pdftex
2482     \c_space_tl 0 ~ R
2483 }

```

(End definition for __pdf_backend_link_last:.)

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

```

2484 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2485 {
2486   \*luatex
2487     \tex_pdfvariable:D linkmargin
2488   \*luatex
2489   \*pdftex
2490     \tex_pdflinkmargin:D
2491   \*pdftex
2492     \dim_eval:n {#1} \scan_stop:
2493 }

```

(End definition for __pdf_backend_link_margin:n.)

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

__pdf_backend_destination:nnnn

```

2494 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2495 {
2496   \*luatex
2497     \tex_pdfextension:D dest ~
2498   \*luatex
2499   \*pdftex
2500     \tex_pdfdest:D
2501   \*pdftex
2502     name {#1}
2503     \str_case:nnF {#2}

```

```

2504         {
2505             { xyz } { xyz }
2506             { fit } { fit }
2507             { fitb } { fitb }
2508             { fitbh } { fitbh }
2509             { fitbv } { fitbv }
2510             { fith } { fith }
2511             { fitv } { fitv }
2512             { fitr } { fitr }
2513         }
2514         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2515         \scan_stop:
2516     }
2517     \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2518     {
2519         \*luatex
2520         \tex_pdfextension:D dest ~
2521         \*pdftex
2522         \tex_pdfdest:D
2523         \*pdftex
2524         name {#1}
2525         fitr ~
2526         width \dim_eval:n {#2} ~
2527         height \dim_eval:n {#3} ~
2528         depth \dim_eval:n {#4} \scan_stop:
2529     }
2530 }

```

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

6.3.2 Catalogue entries

```

\__pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
2531 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2532 {
2533     \*luatex
2534     \tex_pdfextension:D catalog
2535     \*pdftex
2536     \tex_pdfcatalog:D
2537     \*pdftex
2538     { / #1 ~ #2 }
2539 }
2540
2541 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2542 {
2543     \*luatex
2544     \tex_pdfextension:D info
2545     \*pdftex
2546     \tex_pdfinfo:D
2547     \*pdftex
2548     { / #1 ~ #2 }
2549 }
2550 }

```

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

6.3.3 Objects

`\g__pdf_backend_object_prop` For tracking objects to allow finalisation.

```
2551 \prop_new:N \g__pdf_backend_object_prop
```

(End definition for `\g__pdf_backend_object_prop`.)

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

```
\__pdf_backend_object_ref:n
2552 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2553 {
2554   \*luatex
2555     \tex_pdfextension:D obj ~
2556   \*pdfTeX
2557     \tex_pdfobj:D
2558   \*pdfTeX
2559     \tex_pdfobj:D
2560     reserveobjnum ~
2561     \int_const:cn
2562     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2563   \*luatex
2564     { \tex_pdffeedback:D lastobj }
2565   \*pdfTeX
2566     { \tex_pdflastobj:D }
2567   \*pdfTeX
2568     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2569   }
2570   \cs_new:Npn \__pdf_backend_object_ref:n #1
2571     { \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }
```

(End definition for `__pdf_backend_object_new:nn` and `__pdf_backend_object_ref:n`.)

`__pdf_backend_object_write:nn` Writing the data needs a little information about the structure of the object.

```
\__pdf_backend_object_write:nx
2573 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2574 {
2575   \*luatex
2576     \tex_immediate:D \tex_pdfextension:D obj ~
2577   \*pdfTeX
2578     \tex_immediate:D \tex_pdfobj:D
2579   \*pdfTeX
2580     \tex_immediate:D \tex_pdfobj:D
2581     useobjnum ~
2582     \int_use:c
2583     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2584   \str_case_e:nn
2585     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2586     {
2587       { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2588       { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2589       { fstream }
2590       {
2591         stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2592         file ~ { \__pdf_exp_not_ii:nn #2 }
2593       }
2594     }
2595   { stream }
```

```

2595         {
2596             stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2597                 { \_pdf_exp_not_ii:nn #2 }
2598         }
2599     }
2600 }
2601 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
2602 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2603 \cs_new:Npn \_pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

(End definition for \_pdf_backend_object_write:nn, \_pdf_exp_not_i:nn, and \_pdf_exp_not_ii:nn.)

```

_pdf_backend_object_now:nn Much like writing, but direct creation.

```

\_pdf_backend_object_now:nx
2604 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2605 {
2606   \*luatex
2607   \tex_immediate:D \tex_pdfextension:D obj ~
2608   \*pdfTeX
2609   \tex_immediate:D \tex_pdfobj:D
2610   \*pdfTeX
2611   \str_case:nn
2612     {#1}
2613     {
2614       { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2615       { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2616       { fstream }
2617       {
2618         {
2619             stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2620                 file ~ { \_pdf_exp_not_ii:nn #2 }
2621         }
2622         { stream }
2623         {
2624             stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2625                 { \_pdf_exp_not_ii:nn #2 }
2626         }
2627       }
2628     }
2629 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }

(End definition for \_pdf_backend_object_now:nn.)

```

_pdf_backend_object_last: Much like annotation.

```

2630 \cs_new:Npx \_pdf_backend_object_last:
2631 {
2632   \exp_not:N \int_value:w
2633   \*luatex
2634   \exp_not:N \tex_pdffeedback:D lastobj ~
2635   \*pdfTeX
2636   \exp_not:N \tex_pdflastobj:D
2637   \*pdfTeX
2638   \c_space_tl 0 ~ R
2639 }
2640

```

(End definition for `_pdf_backend_object_last:.`)

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

```

2641 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2642 {
2643   \exp_not:N \int_value:w
2644   <*luatex>
2645   \exp_not:N \tex_pdffeedback:D pageref
2646   </luatex>
2647   <*pdftex>
2648   \exp_not:N \tex_pdfpageref:D
2649   </pdftex>
2650   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2651 }

```

(End definition for `_pdf_backend_pageobject_ref:n`.)

6.3.4 Structure

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

```

\_pdf_backend_compresslevel:n
\_pdf_backend_compress_objects:n
\_pdf_backend_objcompresslevel:n
2652 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2653 {
2654   \tex_global:D
2655   <*luatex>
2656   \tex_pdfvariable:D compresslevel
2657   </luatex>
2658   <*pdftex>
2659   \tex_pdfcompresslevel:D
2660   </pdftex>
2661   \int_value:w \int_eval:n {#1} \scan_stop:
2662 }
2663 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2664 {
2665   \bool_if:nTF {#1}
2666   { \_pdf_backend_objcompresslevel:n { 2 } }
2667   { \_pdf_backend_objcompresslevel:n { 0 } }
2668 }
2669 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2670 {
2671   \tex_global:D
2672   <*luatex>
2673   \tex_pdfvariable:D objcompresslevel
2674   </luatex>
2675   <*pdftex>
2676   \tex_pdfobjcompresslevel:D
2677   </pdftex>
2678   #1 \scan_stop:
2679 }

```

(End definition for `_pdf_backend_compresslevel:n`, `_pdf_backend_compress_objects:n`, and `_pdf_backend_objcompresslevel:n`.)

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

```

\_pdf_backend_version_minor_gset:n
2680 \cs_new_protected:Npx \_pdf_backend_version_major_gset:n #1

```

```

2681 {
2682 <*luatex>
2683   \int_compare:nNnT \tex luatexversion:D > { 106 }
2684   {
2685     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2686     \exp_not:N \int_eval:n {#1} \scan_stop:
2687   }
2688 </luatex>
2689 <*pdftex>
2690   \cs_if_exist:NT \tex_pdfmajorversion:D
2691   {
2692     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2693     \exp_not:N \int_eval:n {#1} \scan_stop:
2694   }
2695 </pdftex>
2696 }
2697 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2698 {
2699   \tex_global:D
2700 <*luatex>
2701   \tex_pdfvariable:D minorversion
2702 </luatex>
2703 <*pdftex>
2704   \tex_pdfminorversion:D
2705 </pdftex>
2706   \int_eval:n {#1} \scan_stop:
2707 }

```

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

__pdf_backend_version_major:
 __pdf_backend_version_minor:

As above.

```

2708 \cs_new:Npx \__pdf_backend_version_major:
2709 {
2710 <*luatex>
2711   \int_compare:nNnTF \tex luatexversion:D > { 106 }
2712   { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2713   { 1 }
2714 </luatex>
2715 <*pdftex>
2716   \cs_if_exist:NTF \tex_pdfmajorversion:D
2717   { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2718   { 1 }
2719 </pdftex>
2720 }
2721 \cs_new:Npn \__pdf_backend_version_minor:
2722 {
2723   \tex_the:D
2724 <*luatex>
2725   \tex_pdfvariable:D minorversion
2726 </luatex>
2727 <*pdftex>
2728   \tex_pdfminorversion:D
2729 </pdftex>
2730 }

```

(End 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:`

```
2731 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2732   { \_kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2733 \cs_new_protected:Npn \_pdf_backend_emc:
2734   { \_kernel_backend_literal_page:n { EMC } }
```

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

```
2735 </luatex | pdftex>
```

6.4 dvipdfmx backend

```
2736 <*dvipdfmx | xetex>
```

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

```
\_pdf_backend:x
2737 \cs_new_protected:Npx \_pdf_backend:n #1
2738   { \_kernel_backend_literal:n { pdf: #1 } }
2739 \cs_generate_variant:Nn \_pdf_backend:n { x }
```

(End definition for `_pdf_backend:n`.)

6.4.1 Catalogue entries

```
\_pdf_backend_catalog_gput:nn
\_pdf_backend_info_gput:nn
2740 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2741   { \_pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2742 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2743   { \_pdf_backend:n { docinfo << /#1 ~ #2 >> } }
```

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

6.4.2 Objects

`\g__pdf_backend_object_int` For tracking objects to allow finalisation.

```
\g__pdf_backend_object_prop
2744 \int_new:N \g__pdf_backend_object_int
2745 \prop_new:N \g__pdf_backend_object_prop
```

(End definition for `\g__pdf_backend_object_int` and `\g__pdf_backend_object_prop`.)

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

```
\_pdf_backend_object_ref:n
2746 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2
2747   {
2748     \int_gincr:N \g__pdf_backend_object_int
2749     \int_const:cn
2750     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2751     { \g__pdf_backend_object_int }
2752     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2753   }
2754 \cs_new:Npn \_pdf_backend_object_ref:n #1
2755   { @pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } }
```

(End definition for `_pdf_backend_object_new:nn` and `_pdf_backend_object_ref:n`.)

This is where we choose the actual type.

```

\__pdf_backend_object_write:nn
\__pdf_backend_object_write:nx
\__pdf_backend_object_write: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:nnnn
2756 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2757 {
2758   \exp_args:Nx \__pdf_backend_object_write:nnn
2759   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2760 }
2761 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2762 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2763 {
2764   \use:c { __pdf_backend_object_write_ #1 :nn }
2765   { \__pdf_backend_object_ref:n {#2} } {#3}
2766 }
2767 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2768 {
2769   \__pdf_backend:x
2770   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2771 }
2772 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2773 {
2774   \__pdf_backend:x
2775   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2776 }
2777 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2778 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2779 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2780 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2781 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2782 {
2783   \__pdf_backend:x
2784   {
2785     #1 stream ~ #2 ~
2786     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2787   }
2788 }

```

(End definition for `_pdf_backend_object_write:nn` and others.)

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

```

\__pdf_backend_object_now:nn
\__pdf_backend_object_now:nx
2789 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2790 {
2791   \int_gincr:N \g__pdf_backend_object_int
2792   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2793   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2794   {#2}
2795 }
2796 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for `_pdf_backend_object_now:nn`.)

`__pdf_backend_object_last:`

```

2797 \cs_new:Npn \__pdf_backend_object_last:
2798 { @pdf.obj \int_use:N \g__pdf_backend_object_int }

```

(End definition for _pdf_backend_object_last:.)

_pdf_backend_pageobject_ref:n Page references are easy in dvipdfmx/XT_{EX}.

```
2799 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2800 { @page #1 }
```

(End definition for _pdf_backend_pageobject_ref:n.)

6.4.3 Annotations

\g_pdf_backend_annotation_int Needed as objects which are not annotations could be created.

```
2801 \int_new:N \g_pdf_backend_annotation_int
```

(End definition for \g_pdf_backend_annotation_int.)

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

```
2802 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2803 {
2804   \int_gincr:N \g_pdf_backend_object_int
2805   \int_gset_eq:NN \g_pdf_backend_annotation_int \g_pdf_backend_object_int
2806   \_pdf_backend:x
2807   {
2808     ann ~ @pdf.obj \int_use:N \g_pdf_backend_object_int \c_space_tl
2809     width ~ \dim_eval:n {#1} ~
2810     height ~ \dim_eval:n {#2} ~
2811     depth ~ \dim_eval:n {#3} ~
2812     <</Type/Annot #4 >>
2813   }
2814 }
```

(End definition for _pdf_backend_annotation:nnnn.)

_pdf_backend_annotation_last:

```
2815 \cs_new:Npn \_pdf_backend_annotation_last:
2816 { @pdf.obj \int_use:N \g_pdf_backend_annotation_int }
```

(End definition for _pdf_backend_annotation_last:.)

\g_pdf_backend_link_int To track annotations which are links.

```
2817 \int_new:N \g_pdf_backend_link_int
```

(End definition for \g_pdf_backend_link_int.)

_pdf_backend_link_begin_goto:nnw All created using the same internals.

_pdf_backend_link_begin_user:nnw

_pdf_backend_link_begin:n

_pdf_backend_link_end:

```
2818 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2819 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2820 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2821 { \_pdf_backend_link_begin:n {#1#2} }
2822 \cs_new_protected:Npx \_pdf_backend_link_begin:n #1
2823 {
2824   \int_compare:nNf \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2825   {
2826     \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int
2827   }
2828   \_pdf_backend:x
```

```

2829 {
2830   bann ~
2831   \int_compare:nNnF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
2832   {
2833     @pdf.lnk
2834     \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
2835     \c_space_tl
2836   }
2837   <<
2838   /Type /Annot
2839   #1
2840   >>
2841 }
2842 }
2843 \cs_new_protected:Npn \__pdf_backend_link_end:
2844 { \__pdf_backend:n { eann } }

```

(End definition for __pdf_backend_link_begin_goto:nw and others.)

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

```

2845 \cs_new:Npx \__pdf_backend_link_last:
2846 {
2847   \int_compare:nNnF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
2848   {
2849     @pdf.lnk
2850     \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
2851   }
2852 }

```

(End definition for __pdf_backend_link_last:.)

__pdf_backend_link_margin:n Pass to dvipdfmx.

```

2853 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2854 { \__kernel_backend_literal:x { dvipdfmx:config-g~ \dim_eval:n {#1} } }

```

(End definition for __pdf_backend_link_margin:n.)

_pdf_backend_destination:nn
_pdf_backend_destination:nmmn
_pdf_backend_destination_aux:nmmn

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

```

2855 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2856 {
2857   \__pdf_backend:x
2858   {
2859     dest ~ ( \exp_not:n {#1} )
2860     [
2861       @thispage
2862       \str_case:nnF {#2}
2863       {
2864         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2865         { fit } { /Fit }
2866         { fitb } { /FitB }
2867         { fitbh } { /FitBH }

```

```

2868         { fitbv } { /FitBV ~ @xpos }
2869         { fith } { /FitH ~ @ypos }
2870         { fitv } { /FitV ~ @xpos }
2871         { fitr } { /Fit }
2872     }
2873     { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2874 ]
2875 }
2876 }
2877 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2878 {
2879     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2880     { \dim_eval:n {#2} } {#1} {#3} {#4}
2881 }
2882 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2883 {
2884     \vbox_to_zero:n
2885     {
2886         \__kernel_kern:n {#4}
2887         \hbox:n
2888         {
2889             \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2890             \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2891         }
2892         \tex_vss:D
2893     }
2894     \__kernel_kern:n {#1}
2895     \vbox_to_zero:n
2896     {
2897         \__kernel_kern:n { -#3 }
2898         \hbox:n
2899         {
2900             \__pdf_backend:n
2901             {
2902                 dest ~ (#2)
2903                 [
2904                     @thispage
2905                     /FitR ~
2906                     @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2907                     @xpos ~ @ypos
2908                 ]
2909             }
2910         }
2911         \tex_vss:D
2912     }
2913     \__kernel_kern:n { -#1 }
2914 }

```

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

6.4.4 Structure

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

```

2915 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2916 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2917 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2918 {
2919   \bool_if:nF {#1}
2920   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2921 }

```

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

__pdf_backend_version_major_gset:n
__pdf_backend_version_minor_gset:n

We start with the assumption that the default is active.

```

2922 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2923 {
2924   \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
2925   \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
2926 }
2927 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2928 {
2929   \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
2930   \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
2931 }

```

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

__pdf_backend_version_major:
__pdf_backend_version_minor:

We start with the assumption that the default is active.

```

2932 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2933 \cs_new:Npn \__pdf_backend_version_minor: { 5 }

```

(End definition for __pdf_backend_version_major: and __pdf_backend_version_minor:.)

6.4.5 Marked content

__pdf_backend_bdc:nn
__pdf_backend_emc:

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

```

2934 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2935 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2936 \cs_new_protected:Npn \__pdf_backend_emc:
2937 { \__kernel_backend_literal_page:n { EMC } }

```

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

2938 \langle /dvipdfmx | xetex \rangle

6.5 dvisvgm backend

2939 \langle *dvisvgm \rangle

6.5.1 Catalogue entries

__pdf_backend_catalog_gput:nn
__pdf_backend_info_gput:nn

No-op.

```

2940 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2 { }
2941 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2 { }

```

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

6.5.2 Objects

All no-ops here.

```

\__pdf_backend_object_new:nn
\__pdf_backend_object_ref:n      2942 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2 { }
    \_pdf_backend_object_write:nn 2943 \cs_new:Npn \__pdf_backend_object_ref:n #1 { }
    \_pdf_backend_object_write:nx 2944 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2 { }
\__pdf_backend_object_now:nn     2945 \cs_new_protected:Npn \__pdf_backend_object_write:nx #1#2 { }
\__pdf_backend_object_now:nx     2946 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2 { }
\__pdf_backend_object_last:      2947 \cs_new_protected:Npn \__pdf_backend_object_now:nx #1#2 { }
    \_pdf_backend_pageobject_ref:n 2948 \cs_new:Npn \__pdf_backend_object_last: { }
    2949 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1 { }

```

(End definition for __pdf_backend_object_new:nn and others.)

6.5.3 Structure

These are all no-ops.

```

\_pdf_backend_compresslevel:n    2950 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
\_pdf_backend_compress_objects:n 2951 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1 { }

```

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

_pdf_backend_version_major_gset:n Data not available!

```

\_pdf_backend_version_minor_gset:n 2952 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
                                     2953 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1 { }

```

(End 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:      2954 \cs_new:Npn \__pdf_backend_version_major: { -1 }
                                     2955 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

```

(End definition for __pdf_backend_version_major: and __pdf_backend_version_minor:.)

__pdf_backend_bdc:nn More no-ops.

```

    \__pdf_backend_emc:      2956 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
                             2957 \cs_new_protected:Npn \__pdf_backend_emc: { }

```

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

2958 </divisvgn>

2959 </package>

7 l3backend-opacity Implementation

```

2960 <*package>
2961 <@@=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.

2962 <*dvips>

`_opacity_backend_select:n` No stack so set values directly.

```

\opacity_backend_select_aux:n
2963 \cs_new_protected:Npn \_opacity_backend_select:n #1
2964 {
2965   \exp_args:Nx \_opacity_backend_select_aux:n
2966   { \fp_eval:n { min(max(0,#1),1) } }
2967 }
2968 \cs_new_protected:Npn \_opacity_backend_select_aux:n #1
2969 {
2970   \_kernel_backend_postscript:n
2971   { #1 ~ .setfillconstantalpha ~ #1 ~ .setstrokeconstantalpha }
2972 }

```

(End definition for `_opacity_backend_select:n` and `_opacity_backend_select_aux:n`.)

`_opacity_backend_fill:n` Similar to the above but with no stack and only adding to one or other of the entries.

`_opacity_backend_stroke:n`

`_opacity_backend:nn`

`_opacity_backend:xn`

```

2973 \cs_new_protected:Npn \_opacity_backend_fill:n #1
2974 { \_opacity_backend:xn { \fp_eval:n { min(max(0,#1),1) } } { fill } }
2975 \cs_new_protected:Npn \_opacity_backend_stroke:n #1
2976 { \_opacity_backend:xn { \fp_eval:n { min(max(0,#1),1) } } { stroke } }
2977 \cs_new_protected:Npn \_opacity_backend:nn #1#2
2978 {
2979   \_kernel_backend_postscript:n { #1 ~ .set #2 constantalpha }
2980 }
2981 \cs_generate_variant:Nn \_opacity_backend:nn { x }

```

(End definition for `_opacity_backend_fill:n`, `_opacity_backend_stroke:n`, and `_opacity_backend:nn`.)

```

2982 </dvips>
2983 <*dvipdfmx | luatex | pdftex | xetex>

```

`\c_opacity_backend_stack_int` Set up a stack.

```

2984 \cs_if_exist:NT \pdfmanagement_add:nnn
2985 {
2986   \_kernel_color_backend_stack_init:Nnn \c_opacity_backend_stack_int
2987   { page ~ direct } { /opacity 1 ~ gs }
2988   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
2989   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
2990 }

```

(End 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.

`\l_opacity_backend_stroke_tl`

```

2991 \tl_new:N \l_opacity_backend_fill_tl
2992 \tl_new:N \l_opacity_backend_stroke_tl

```

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

`_opacity_backend_select:n` Other than the need to evaluate the opacity as an `fp`, much the same as color.

`_opacity_backend_select_aux:n`

`_opacity_backend_reset:`

```

2993 \cs_new_protected:Npn \_opacity_backend_select:n #1
2994 {
2995   \exp_args:Nx \_opacity_backend_select_aux:n
2996   { \fp_eval:n { min(max(0,#1),1) } }
2997 }
2998 \cs_new_protected:Npn \_opacity_backend_select_aux:n #1

```

```

2999 {
3000   \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3001   \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3002   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3003     { opacity #1 }
3004     { << /ca ~ #1 /CA ~ #1 >> }
3005   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3006     { /opacity #1 ~ gs }
3007   \group_insert_after:N \__opacity_backend_reset:
3008 }
3009 \cs_if_exist:NF \pdfmanagement_add:nnn
3010 {
3011   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1 { }
3012 }
3013 \cs_new_protected:Npn \__opacity_backend_reset:
3014 { \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int }

(End definition for \__opacity_backend_select:n, \__opacity_backend_select_aux:n, and \__opacity_backend_reset:.)

```

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

```

\__opacity_backend_fillstroke:nn
\__opacity_backend_fillstroke:xx
3015 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3016 {
3017   \__opacity_backend_fill_stroke:xx
3018   { \fp_eval:n { min(max(0,#1),1) } }
3019   \l__opacity_backend_stroke_tl
3020 }
3021 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3022 {
3023   \__opacity_backend_fill_stroke:xx
3024   \l__opacity_backend_fill_tl
3025   { \fp_eval:n { min(max(0,#1),1) } }
3026 }
3027 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3028 {
3029   \str_if_eq:nnTF {#1} {#2}
3030   { \__opacity_backend_select_aux:n {#1} }
3031   {
3032     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3033     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3034     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3035       { opacity.fill #1 }
3036       { << /ca ~ #1 >> }
3037     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3038       { opacity.stroke #1 }
3039       { << /CA ~ #2 >> }
3040     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3041       { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3042     \group_insert_after:N \__opacity_backend_reset:
3043   }
3044 }
3045 \cs_generate_variant:Nn \__opacity_backend_fill_stroke:nn { xx }

```

(End definition for `_opacity_backend_fill:n`, `_opacity_backend_stroke:n`, and `_opacity_backend_fillstroke:nn`.)

3046 `</dvipdfmx | luatex | pdftex | xetex>`

3047 `<*dvipdfmx | xdvipdfmx>`

`_opacity_backend_select:n` Older backends have no stack support, so everything is done directly.

3048 `\int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }`

3049 `{`

3050 `\cs_gset_protected:Npn _opacity_backend_select_aux:n #1`

3051 `{`

3052 `\tl_set:Nn \l__opacity_backend_fill_tl {#1}`

3053 `\tl_set:Nn \l__opacity_backend_stroke_tl {#1}`

3054 `\pdfmanagement_add:nnn { Page / Resources / ExtGState }`

3055 `{ opacity #1 }`

3056 `{ << /ca ~ #1 /CA ~ #1 >> }`

3057 `__kernel_backend_literal_pdf:n { /opacity #1 ~ gs }`

3058 `}`

3059 `\cs_gset_protected:Npn _opacity_backend_fill_stroke:nn #1#2`

3060 `{`

3061 `\str_if_eq:nnTF {#1} {#2}`

3062 `{ _opacity_backend_select_aux:n {#1} }`

3063 `{`

3064 `\tl_set:Nn \l__opacity_backend_fill_tl {#1}`

3065 `\tl_set:Nn \l__opacity_backend_stroke_tl {#2}`

3066 `\pdfmanagement_add:nnn { Page / Resources / ExtGState }`

3067 `{ opacity.fill #1 }`

3068 `{ << /ca ~ #1 >> }`

3069 `\pdfmanagement_add:nnn { Page / Resources / ExtGState }`

3070 `{ opacity.stroke #1 }`

3071 `{ << /CA ~ #2 >> }`

3072 `__kernel_backend_literal_pdf:n`

3073 `{ /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }`

3074 `}`

3075 `}`

3076 `}`

(End definition for `_opacity_backend_select:n`.)

3077 `</dvipdfmx | xdvipdfmx>`

3078 `<*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`

3079 `\cs_new_protected:Npn _opacity_backend_select:n #1`

3080 `{ _opacity_backend:nn {#1} { } }`

3081 `\cs_new_protected:Npn _opacity_backend_fill:n #1`

3082 `{ _opacity_backend:nn {#1} { fill- } }`

3083 `\cs_new_protected:Npn _opacity_backend_stroke:n #1`

3084 `{ _opacity_backend:nn { {#1} } { stroke- } }`

3085 `\cs_new_protected:Npn _opacity_backend:nn #1#2`

3086 `{ __kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }`

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

3087 `</dvisvgm>`

3088 `</package>`

8 l3backend-header Implementation

	3089 <code><*dvips & header></code>	
<code>color.sc</code>	Empty definition for color at the top level.	
	3090 <code>/color.sc { } def</code>	
	<i>(End definition for color.sc. This function is documented on page ??.)</i>	
<code>TeXcolorseparation</code> <code>separation</code>	Support for separation/spot colors: this strange naming is so things work with the color stack.	
	3091 <code>TeXDict begin</code>	
	3092 <code>/TeXcolorseparation { setcolor } def</code>	
	3093 <code>end</code>	
	<i>(End definition for TeXcolorseparation and separation. These functions are documented on page ??.)</i>	
<code>pdf.globaldict</code>	A small global dictionary for backend use.	
	3094 <code>true setglobal</code>	
	3095 <code>/pdf.globaldict 4 dict def</code>	
	3096 <code>false setglobal</code>	
	<i>(End definition for pdf.globaldict. This function is documented on page ??.)</i>	
<code>pdf.cvs</code> <code>pdf.dvi.pt</code> <code>pdf.pt.dvi</code> <code>pdf.rect.ht</code>	Small utilities for PostScript manipulations. Conversion to DVI dimensions is done here to allow for Resolution. The total height of a rectangle (an array) needs a little maths, in contrast to simply extracting a value.	
	3097 <code>/pdf.cvs { 65534 string cvs } def</code>	
	3098 <code>/pdf.dvi.pt { 72.27 mul Resolution div } def</code>	
	3099 <code>/pdf.pt.dvi { 72.27 div Resolution mul } def</code>	
	3100 <code>/pdf.rect.ht { dup 1 get neg exch 3 get add } def</code>	
	<i>(End definition for pdf.cvs and others. These functions are documented on page ??.)</i>	
<code>pdf.linkmargin</code> <code>pdf.linkdp.pad</code> <code>pdf.linkht.pad</code>	Settings which are defined up-front in SDict.	
	3101 <code>/pdf.linkmargin { 1 pdf.pt.dvi } def</code>	
	3102 <code>/pdf.linkdp.pad { 0 } def</code>	
	3103 <code>/pdf.linkht.pad { 0 } def</code>	
	<i>(End definition for pdf.linkmargin, pdf.linkdp.pad, and pdf.linkht.pad. These functions are documented on page ??.)</i>	
<code>pdf.rect</code> <code>pdf.save.ll</code> <code>pdf.save.ur</code> <code>pdf.save.linkll</code> <code>pdf.save.linkur</code> <code>pdf.llx</code> <code>pdf.lly</code> <code>pdf.urx</code> <code>pdf.ury</code>	Functions for marking the limits of an annotation/link, plus drawing the border. We separate links for generic annotations to support adding a margin and setting a minimal size.	
	3104 <code>/pdf.rect</code>	
	3105 <code>{ /Rect [pdf.llx pdf.lly pdf.urx pdf.ury] } def</code>	
	3106 <code>/pdf.save.ll</code>	
	3107 <code>{</code>	
	3108 <code>currentpoint</code>	
	3109 <code>/pdf.lly exch def</code>	
	3110 <code>/pdf.llx exch def</code>	
	3111 <code>}</code>	
	3112 <code>def</code>	
	3113 <code>/pdf.save.ur</code>	

```

3114 {
3115     currentpoint
3116     /pdf.ury exch def
3117     /pdf.urx exch def
3118 }
3119 def
3120 /pdf.save.linkll
3121 {
3122     currentpoint
3123     pdf.linkmargin add
3124     pdf.linkdp.pad add
3125     /pdf.lly exch def
3126     pdf.linkmargin sub
3127     /pdf.llx exch def
3128 }
3129 def
3130 /pdf.save.linkur
3131 {
3132     currentpoint
3133     pdf.linkmargin sub
3134     pdf.linkht.pad sub
3135     /pdf.ury exch def
3136     pdf.linkmargin add
3137     /pdf.urx exch def
3138 }
3139 def

```

(End definition for pdf.rect and others. These functions are documented on page ??.)

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 co-ordinate 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      3140 /pdf.dest.anchor
pdf.dev.y      3141 {
pdf.tmpa       3142     currentpoint exch
pdf.tmpb       3143     pdf.dvi.pt 72 add
pdf.tmpc       3144     /pdf.dest.x exch def
pdf.tmpd       3145     pdf.dvi.pt
3146     vsize 72 sub exch sub
3147     /pdf.dest.y exch def
3148 }
3149 def
3150 /pdf.dest.point
3151 { pdf.dest.x pdf.dest.y } def
3152 /pdf.dest2device
3153 {
3154     /pdf.dest.y exch def
3155     /pdf.dest.x exch def
3156     matrix currentmatrix
3157     matrix defaultmatrix
3158     matrix invertmatrix
3159     matrix concatmatrix

```

```

3160     cvx exec
3161     /pdf.dev.y  exch def
3162     /pdf.dev.x  exch def
3163     /pdf.tmpd   exch def
3164     /pdf.tmpc   exch def
3165     /pdf.tmpb   exch def
3166     /pdf.tmpa   exch def
3167     pdf.dest.x pdf.tmpa mul
3168         pdf.dest.y pdf.tmpc mul add
3169         pdf.dev.x add
3170     pdf.dest.x pdf.tmpb mul
3171         pdf.dest.y pdf.tmpd mul add
3172         pdf.dev.y add
3173     }
3174     def

```

(End definition for pdf.dest.anchor and others. These functions are documented on page ??.)

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	3175 /pdf.bordertracking false def
pdf.brokenlink.skip	3176 /pdf.bordertracking.begin
pdf.brokenlink.dict	3177 {
pdf.bordertracking.endpage	3178 SDict /pdf.bordertracking true put
pdf.bordertracking.continue	3179 SDict /pdf.leftboundary undef
pdf.originx	3180 SDict /pdf.rightboundary undef
pdf.originy	3181 /a where
	3182 {
	3183 /a
	3184 {
	3185 currentpoint pop
	3186 SDict /pdf.rightboundary known dup
	3187 {
	3188 SDict /pdf.rightboundary get 2 index lt
	3189 { not }
	3190 if
	3191 }
	3192 if
	3193 { pop }
	3194 { SDict exch /pdf.rightboundary exch put }
	3195 ifelse
	3196 moveto
	3197 currentpoint pop
	3198 SDict /pdf.leftboundary known dup
	3199 {
	3200 SDict /pdf.leftboundary get 2 index gt
	3201 { not }
	3202 if
	3203 }
	3204 if
	3205 { pop }
	3206 { SDict exch /pdf.leftboundary exch put }

```

3207         ifelse
3208     }
3209     put
3210 }
3211 if
3212 }
3213 def
3214 /pdf.bordertracking.end
3215 {
3216     /a where { /a { moveto } put } if
3217     /x where { /x { 0 exch rmoveto } put } if
3218     SDict /pdf.leftboundary known
3219     { pdf.outerbox 0 pdf.leftboundary put }
3220     if
3221     SDict /pdf.rightboundary known
3222     { pdf.outerbox 2 pdf.rightboundary put }
3223     if
3224     SDict /pdf.bordertracking false put
3225 }
3226 def
3227 /pdf.bordertracking.endpage
3228 {
3229     pdf.bordertracking
3230     {
3231         pdf.bordertracking.end
3232         true setglobal
3233         pdf.globaldict
3234         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3235         pdf.globaldict
3236         /pdf.brokenlink.skip pdf.baselineskip put
3237         pdf.globaldict
3238         /pdf.brokenlink.dict
3239         pdf.link.dict pdf.cvs put
3240         false setglobal
3241         mark pdf.link.dict cvx exec /Rect
3242         [
3243             pdf.llx
3244             pdf.lly
3245             pdf.outerbox 2 get pdf.linkmargin add
3246             currentpoint exch pop
3247             pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3248         ]
3249         /ANN pdf.pdfmark
3250     }
3251     if
3252 }
3253 def
3254 /pdf.bordertracking.continue
3255 {
3256     /pdf.link.dict pdf.globaldict
3257     /pdf.brokenlink.dict get def
3258     /pdf.outerbox pdf.globaldict
3259     /pdf.brokenlink.rect get def
3260     /pdf.baselineskip pdf.globaldict

```

```

3261     /pdf.brokenlink.skip get def
3262 pdf.globaldict dup dup
3263 /pdf.brokenlink.dict undef
3264 /pdf.brokenlink.skip undef
3265 /pdf.brokenlink.rect undef
3266 currentpoint
3267 /pdf.originy exch def
3268 /pdf.originx exch def
3269 /a where
3270 {
3271     /a
3272     {
3273         moveto
3274         SDict
3275         begin
3276             currentpoint pdf.originy ne exch
3277             pdf.originx ne or
3278             {
3279                 pdf.save.linkll
3280                 /pdf.lly
3281                 pdf.lly pdf.outerbox 1 get sub def
3282                 pdf.bordertracking.begin
3283             }
3284             if
3285             end
3286         }
3287         put
3288     }
3289     if
3290     /x where
3291     {
3292         /x
3293         {
3294             0 exch rmoveto
3295             SDict
3296             begin
3297                 currentpoint
3298                 pdf.originy ne exch pdf.originx ne or
3299                 {
3300                     pdf.save.linkll
3301                     /pdf.lly
3302                     pdf.lly pdf.outerbox 1 get sub def
3303                     pdf.bordertracking.begin
3304                 }
3305                 if
3306                 end
3307             }
3308             put
3309         }
3310     }
3311 }
3312 def

```

(End definition for pdf.bordertracking and others. These functions are documented on page ??.)

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

```

3313 /pdf.breaklink
3314 {
3315     pop
3316     counttomark 2 mod 0 eq
3317     {
3318         counttomark /pdf.count exch def
3319         {
3320             pdf.count 0 eq { exit } if
3321             counttomark 2 roll
3322             1 index /Rect eq
3323             {
3324                 dup 4 array copy
3325                 dup dup
3326                 1 get
3327                 pdf.outerbox pdf.rect.ht
3328                 pdf.linkmargin 2 mul add sub
3329                 3 exch put
3330                 dup
3331                 pdf.outerbox 2 get
3332                 pdf.linkmargin add
3333                 2 exch put
3334                 dup dup
3335                 3 get
3336                 pdf.outerbox pdf.rect.ht
3337                 pdf.linkmargin 2 mul add add
3338                 1 exch put
3339                 /pdf.currentrect exch def
3340                 pdf.breaklink.write
3341                 {
3342                     pdf.currentrect
3343                     dup
3344                     pdf.outerbox 0 get
3345                     pdf.linkmargin sub
3346                     0 exch put
3347                     dup
3348                     pdf.outerbox 2 get
3349                     pdf.linkmargin add
3350                     2 exch put
3351                     dup dup
3352                     1 get
3353                     pdf.baselineskip add
3354                     1 exch put
3355                     dup dup
3356                     3 get
3357                     pdf.baselineskip add
3358                     3 exch put
3359                     /pdf.currentrect exch def
3360                     pdf.breaklink.write

```

```

3361         }
3362         1 index 3 get
3363         pdf.linkmargin 2 mul add
3364         pdf.outerbox pdf.rect.ht add
3365         2 index 1 get sub
3366         pdf.baselineskip div round cvi 1 sub
3367         exch
3368         repeat
3369         pdf.currentrect
3370         dup
3371         pdf.outerbox 0 get
3372         pdf.linkmargin sub
3373         0 exch put
3374         dup dup
3375         1 get
3376         pdf.baselineskip add
3377         1 exch put
3378         dup dup
3379         3 get
3380         pdf.baselineskip add
3381         3 exch put
3382         dup 2 index 2 get 2 exch put
3383         /pdf.currentrect exch def
3384         pdf.breaklink.write
3385         SDict /pdf.pdfmark.good false put
3386         exit
3387     }
3388     { pdf.count 2 sub /pdf.count exch def }
3389     ifelse
3390 }
3391 loop
3392 }
3393 if
3394 /ANN
3395 }
3396 def
3397 /pdf.breaklink.write
3398 {
3399     counttomark 1 sub
3400     index /_objdef eq
3401     {
3402         counttomark -2 roll
3403         dup wcheck
3404         {
3405             readonly
3406             counttomark 2 roll
3407         }
3408         { pop pop }
3409         ifelse
3410     }
3411     if
3412     counttomark 1 add copy
3413     pop pdf.currentrect
3414     /ANN pdfmark

```

```

3415 }
3416 def

```

(End definition for pdf.breaklink and others. These functions are documented on page ??.)

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

```

3417 /pdf.pdfmark
3418 {
3419   SDict /pdf.pdfmark.good true put
3420   dup /ANN eq
3421   {
3422     pdf.pdfmark.store
3423     pdf.pdfmark.dict
3424     begin
3425       Subtype /Link eq
3426       currentdict /Rect known and
3427       SDict /pdf.outerbox known and
3428       SDict /pdf.baselineskip known and
3429       {
3430         Rect 3 get
3431         pdf.linkmargin 2 mul add
3432         pdf.outerbox pdf.rect.ht add
3433         Rect 1 get sub
3434         pdf.baselineskip div round cvi 0 gt
3435         { pdf.breaklink }
3436         if
3437       }
3438       if
3439       end
3440       SDict /pdf.outerbox undef
3441       SDict /pdf.baselineskip undef
3442       currentdict /pdf.pdfmark.dict undef
3443     }
3444     if
3445     pdf.pdfmark.good
3446     { pdfmark }
3447     { cleartomark }
3448     ifelse
3449   }
3450   def
3451 /pdf.pdfmark.store
3452 {
3453   /pdf.pdfmark.dict 65534 dict def
3454   counttomark 1 add copy
3455   pop
3456   {
3457     dup mark eq
3458     {
3459       pop
3460       exit

```

```

3461         }
3462         {
3463             pdf.pdfmark.dict
3464             begin def end
3465         }
3466     ifelse
3467 }
3468 loop
3469 }
3470 def

```

(End definition for pdf.pdfmark and others. These functions are documented on page ??.)

```

3471 </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.

- _ 146
- A**
- \AtBeginDvi 59, 60, 518, 519, 665, 666
- B**
- bool commands:
 - \bool_gset_false:N 1164, 1183, 1206, 1228, 1244, 1345, 1582, 1618, 2183, 2229
 - \bool_gset_true:N 1162, 1231, 1343, 1597, 2176, 2182
 - \bool_if:NTF 57, 663, 1174, 1178, 1194, 1197, 1201, 1212, 1219, 1223, 1235, 1239, 1356, 1361, 1366, 1556, 1601, 1714, 1749, 1859, 1901, 2171, 2186, 2191, 2196
 - \bool_if:nTF 2405, 2665, 2919
 - \bool_lazy_or:nnTF 1741, 1894
 - \bool_new:N 1165, 1232, 1346, 1598, 2159, 2160
 - \bool_set_false:N 1724, 1826, 1919, 1983
- box commands:
 - \box_dp:N 216, 218, 266, 268, 323, 325, 372, 374, 376, 378, 2208, 2241, 2242, 2267
 - \box_ht:N 218, 268, 325, 376, 378, 1761, 1956, 2213, 2252, 2253, 2269
 - \box_if_empty:NTF 2302
 - \box_move_down:nn 2133, 2208
 - \box_move_up:nn 2135, 2213
 - \box_new:N 2018, 2123, 2124
 - \box_set_dp:Nn 1681
 - \box_set_ht:Nn 1680
 - \box_set_wd:Nn 280, 1679
 - \box_use:N 223, 241, 255, 271, 298, 312, 328, 344, 356, 407, 421, 440, 1296, 1491, 1682, 2164
 - \box_wd:N 217, 225, 267, 273, 324, 330, 373, 375, 1760, 1955
- box internal commands:
 - __box_backend_clip:N 205, 260, 317, 361
 - \l_box_backend_cos_fp 275
 - __box_backend_rotate:Nn 227, 275, 332, 411
 - __box_backend_rotate_aux:Nn 227, 275, 332
 - __box_backend_scale:Nnn 244, 303, 347, 424
 - \l_box_backend_sin_fp 275
 - \g_box_clip_path_int 361
- C**
- char commands:
 - \char_set_catcode_space:n 146
- clist commands:
 - \clist_map_function:nN ... 1252, 1376
 - \clist_map_function:nn 1625
- color internal commands:
 - __color_backend:nnn 1050
 - __color_backend_cmyk:w 1051
 - __color_backend_devicen_init:n 898
 - __color_backend_devicen_init:nnn 811, 898
 - __color_backend_devicen_init:w 898
 - __color_backend_fill:n 957, 984, 1014, 1032, 1039
 - __color_backend_fill_cmyk:n 957, 991, 1014, 1039
 - __color_backend_fill_devicen:nn 983, 1005, 1031, 1101
 - __color_backend_fill_gray:n 957, 991, 1014, 1039
 - __color_backend_fill_rgb:n 957, 991, 1014, 1039
 - __color_backend_fill_separation:nn 983, 991, 1031, 1101
 - \l_color_backend_fill_tl 625, 635, 965, 980
 - \c_color_backend_main_stack_int 508
 - __color_backend_pickup:N .. 448, 471
 - __color_backend_pickup:w 14, 448, 471
 - __color_backend_reset: 607, 627, 644, 968, 981, 991, 1023, 1048
 - __color_backend_rgb:w 1074
 - __color_backend_select:n .. 607, 659
 - __color_backend_select:nn .. 627, 838
 - __color_backend_select_cmyk:n .. 607, 627, 644
 - __color_backend_select_devicen:nn 658, 831, 837, 949
 - __color_backend_select_gray:n .. 607, 627, 644

_color_backend_select_rgb:n ...	\l_color_backend_stroke_tl
..... 607, 627, 644 625, 636, 967, 978
_color_backend_select_separation:nn	\g_color_model_int 681, 817, 862, 935
..... 658, 831, 837, 949	\c_color_model_range_CIELAB_tl .
_color_backend_separation- 772, 807, 882, 889
init:n 661, 840, 922, 946	color.sc 607, 3090
_color_backend_separation-	cs commands:
init:nnn 661	\cs_generate_variant:Nn .. 49, 53,
_color_backend_separation-	56, 91, 130, 135, 162, 193, 199, 557,
init:nnnn 661	594, 674, 1111, 1306, 1500, 1873,
_color_backend_separation-	1930, 1946, 2022, 2059, 2118, 2601,
init:nnnnn 661, 833, 840	2629, 2739, 2761, 2796, 2981, 3045
_color_backend_separation-	\cs_gset:Npx 2924, 2929
init:nw 661	\cs_gset_eq:NN 651,
_color_backend_separation-	652, 952, 998, 999, 1005, 1007, 1009
init:w 661	\cs_gset_protected:Npn 646, 653, 895, 951,
_color_backend_separation-	993, 1000, 1002, 1004, 3011, 3050, 3059
init_/DeviceCMYK:nnn 661	\cs_if_exist:NTF 27,
_color_backend_separation-	59, 449, 472, 518, 533, 665, 856,
init_/DeviceGray:nnn 661	893, 929, 2298, 2690, 2716, 2984, 3009
_color_backend_separation-	\cs_if_exist_use:NTF 38, 687
init_/DeviceRGB:nnn 661	\cs_new:Npn 696, 698, 700,
_color_backend_separation-	702, 709, 715, 717, 723, 740, 747,
init_aux:nnnnn 661	749, 940, 1257, 1381, 1629, 1959,
_color_backend_separation-	1968, 2012, 2037, 2119, 2121, 2154,
init_CIELAB:nnn 661, 833, 840	2323, 2417, 2418, 2571, 2602, 2603,
_color_backend_separation-	2721, 2754, 2797, 2799, 2815, 2932,
init_CIELAB:nnnnnn 834	2933, 2943, 2948, 2949, 2954, 2955
_color_backend_separation-	\cs_new:Npx 2438, 2473, 2630, 2641, 2708, 2845
init_count:n 661	\cs_new_eq:NN .. 46, 660, 839, 946,
_color_backend_separation-	987, 988, 1035, 1036, 1103, 1104,
init_count:w 661	1110, 1305, 1311, 1312, 1499, 1506,
_color_backend_separation-	1691, 1720, 1771, 1772, 1814, 1822,
init_Device:Nn 661	1844, 1915, 1972, 1979, 2011, 2164
\g_color_backend_stack_int ... 508	\cs_new_protected:Npn 47,
\l_color_backend_stack_int	51, 54, 64, 70, 75, 77, 81, 92, 102,
... 505, 535, 541, 637, 641, 966, 979	111, 120, 133, 136, 138, 140, 160,
_color_backend_stroke:n 957, 986, 991	165, 174, 184, 194, 205, 227, 229,
_color_backend_stroke_cmyk:n ..	244, 260, 275, 277, 303, 317, 332,
... 957, 1014, 1050	334, 347, 361, 411, 424, 448, 466,
_color_backend_stroke_cmyk:w 1050	471, 479, 509, 548, 558, 570, 584,
_color_backend_stroke_devicen:nn	595, 607, 609, 611, 613, 621, 627,
... 983, 1009, 1031, 1101	629, 631, 633, 640, 658, 675, 765,
_color_backend_stroke_gray:n ..	811, 831, 832, 833, 834, 837, 840,
... 957, 1014, 1050	867, 871, 898, 957, 959, 961, 963,
_color_backend_stroke_gray-	970, 972, 974, 976, 983, 985, 1014,
aux:n 1050	1016, 1018, 1020, 1025, 1027, 1029,
_color_backend_stroke_rgb:n ...	1031, 1033, 1039, 1041, 1043, 1045,
... 957, 1014, 1050	1050, 1052, 1063, 1071, 1073, 1075,
_color_backend_stroke_rgb:w . 1050	1101, 1102, 1112, 1117, 1122, 1124,
_color_backend_stroke_separation:nn	1126, 1134, 1142, 1151, 1161, 1163,
... 983, 991, 1031, 1101	1166, 1168, 1185, 1190, 1208, 1230,

1233, 1246, 1259, 1264, 1266, 1268,
 1270, 1272, 1274, 1276, 1278, 1283,
 1307, 1309, 1313, 1318, 1323, 1333,
 1342, 1344, 1347, 1349, 1351, 1353,
 1358, 1363, 1368, 1370, 1383, 1388,
 1390, 1392, 1394, 1396, 1398, 1400,
 1402, 1413, 1438, 1450, 1462, 1474,
 1481, 1501, 1507, 1512, 1517, 1528,
 1538, 1548, 1550, 1552, 1554, 1585,
 1587, 1592, 1594, 1596, 1599, 1620,
 1631, 1644, 1646, 1648, 1650, 1652,
 1654, 1656, 1658, 1660, 1668, 1692,
 1706, 1721, 1733, 1738, 1766, 1778,
 1791, 1801, 1816, 1823, 1831, 1842,
 1846, 1849, 1864, 1874, 1909, 1916,
 1922, 1928, 1931, 1938, 1947, 1952,
 1960, 1973, 1980, 1986, 1988, 1990,
 2001, 2020, 2023, 2025, 2029, 2039,
 2060, 2065, 2070, 2075, 2085, 2090,
 2098, 2126, 2131, 2163, 2165, 2167,
 2169, 2174, 2189, 2194, 2231, 2260,
 2279, 2288, 2325, 2332, 2358, 2363,
 2391, 2403, 2415, 2416, 2419, 2421,
 2425, 2449, 2451, 2453, 2464, 2484,
 2494, 2517, 2531, 2541, 2552, 2573,
 2604, 2652, 2663, 2669, 2697, 2731,
 2733, 2740, 2742, 2746, 2756, 2762,
 2767, 2772, 2777, 2779, 2781, 2789,
 2802, 2818, 2820, 2843, 2853, 2855,
 2877, 2882, 2915, 2917, 2922, 2927,
 2934, 2936, 2940, 2941, 2942, 2944,
 2945, 2946, 2947, 2950, 2951, 2952,
 2953, 2956, 2957, 2963, 2968, 2973,
 2975, 2977, 2993, 2998, 3013, 3015,
 3021, 3027, 3079, 3081, 3083, 3085
 \cs_new_protected:Npx
 512, 661, 1086, 2680, 2737, 2822
 \cs_set:Npn 144
 \cs_set_eq:NN 2319, 2320
 \cs_set_protected:Npn 451, 474

D

dim commands:

\dim_eval:n 2129, 2361,
 2433, 2434, 2435, 2492, 2527, 2528,
 2529, 2809, 2810, 2811, 2854, 2880
 \dim_max:nn 2239, 2250
 \dim_set:Nn ... 1760, 1761, 1955, 1956
 \dim_to_decimal:n .. 372, 373, 374,
 375, 376, 378, 1510, 1515, 1521,
 1522, 1523, 1524, 1533, 1534, 1535,
 1626, 1645, 2006, 2007, 2237, 2248,
 2266, 2267, 2268, 2269, 2273, 2329
 \dim_to_decimal_in_bp:n

.... 216, 217, 218, 266, 267, 268,
 323, 324, 325, 1130, 1131, 1138,
 1139, 1146, 1147, 1155, 1156, 1157,
 1254, 1258, 1262, 1316, 1321, 1327,
 1328, 1329, 1337, 1338, 1378, 1382,
 1386, 1630, 1697, 1698, 1699, 1700,
 1836, 1837, 1838, 1839, 1888, 1889,
 1890, 1891, 1995, 1996, 1997, 1998

draw internal commands:

__draw_align_currentpoint... .. 34
 __draw_backend_add_to_path:n ...
 1507, 1553
 __draw_backend_begin:
 1112, 1307, 1501
 __draw_backend_box_use:Nnnnn ...
 30, 1283, 1481, 1668
 __draw_backend_cap_butt:
 1246, 1370, 1620
 __draw_backend_cap_rectangle: ..
 1246, 1370, 1620
 __draw_backend_cap_round:
 1246, 1370, 1620
 __draw_backend_clip: 1166, 1347, 1552
 __draw_backend_closepath:
 1166, 1347, 1552
 __draw_backend_closestroke: ...
 1166, 1347, 1552
 __draw_backend_cm:nnnn 1278, 1291,
 1292, 1293, 1402, 1485, 1660, 1671
 __draw_backend_cm_aux:nnnn .. 1402
 __draw_backend_cm_decompose:nnnnN
 1408, 1437
 __draw_backend_cm_decompose_-
 auxi:nnnnN 1437
 __draw_backend_cm_decompose_-
 auxii:nnnnN 1437
 __draw_backend_cm_decompose_-
 auxiii:nnnnN 1437
 __draw_backend_curveto:nnnnnn ..
 1126, 1313, 1507
 __draw_backend_dash:n
 1246, 1370, 1620
 __draw_backend_dash_aux:nn .. 1620
 __draw_backend_dash_pattern:nn ..
 1246, 1370, 1620
 __draw_backend_discardpath: ...
 1166, 1347, 1552
 __draw_backend_end: 1112, 1307, 1501
 __draw_backend_evenodd_rule: ...
 1161, 1342, 1548
 __draw_backend_fill: 1166, 1347, 1552
 __draw_backend_fillstroke:
 1166, 1347, 1552

<code>__draw_backend_join_bevel:</code>	1246 , 1370 , 1620
<code>__draw_backend_join_miter:</code>	1246 , 1370 , 1620
<code>__draw_backend_join_round:</code>	1246 , 1370 , 1620
<code>__draw_backend_lineto:nn</code>	1126 , 1313 , 1507
<code>__draw_backend_linewidth:n</code>	1246 , 1370 , 1620
<code>__draw_backend_literal:n</code>	1110 , 1115 , 1119 , 1123 , 1125 , 1128 , 1136 , 1144 , 1153 , 1167 , 1170 , 1171 , 1172 , 1173 , 1176 , 1182 , 1192 , 1199 , 1205 , 1210 , 1215 , 1216 , 1217 , 1218 , 1221 , 1227 , 1237 , 1243 , 1248 , 1261 , 1265 , 1267 , 1269 , 1271 , 1273 , 1275 , 1277 , 1280 , 1285 , 1286 , 1287 , 1288 , 1289 , 1290 , 1294 , 1295 , 1297 , 1298 , 1299 , 1300 , 1301 , 1305 , 1315 , 1320 , 1325 , 1335 , 1348 , 1350 , 1352 , 1355 , 1360 , 1365 , 1369 , 1372 , 1385 , 1389 , 1391 , 1393 , 1395 , 1397 , 1399 , 1401 , 1499 , 1559 , 1578 , 1604
<code>__draw_backend_miterlimit:n</code> . . .	1246 , 1370 , 1620
<code>__draw_backend_moveto:nn</code>	1126 , 1313 , 1507
<code>__draw_backend_nonzero_rule:</code> . . .	1161 , 1342 , 1548
<code>__draw_backend_path:n</code>	1552
<code>__draw_backend_rectangle:nnnn</code> . .	1126 , 1313 , 1507
<code>__draw_backend_scope:n</code> 1549 , 1551 , 1571 , 1611 , 1633 , 1645 , 1647 , 1649 , 1651 , 1653 , 1655 , 1657 , 1659 , 1662	
<code>__draw_backend_scope_begin:</code> . . .	1122 , 1308 , 1311
<code>__draw_backend_scope_end:</code>	1122 , 1310 , 1311
<code>__draw_backend_stroke:</code>	1166 , 1347 , 1552
<code>\g__draw_clip_path_int</code>	1558 , 1561 , 1574 , 1603 , 1606 , 1614
<code>\g__draw_draw_clip_bool</code> . . 1166 , 1552	
<code>\g__draw_draw_eor_bool</code>	1161 , 1178 , 1194 , 1201 , 1212 , 1223 , 1239 , 1342 , 1356 , 1361 , 1366
<code>\g__draw_draw_path_int</code>	1552
<code>\g__draw_draw_path_tl</code>	1507 , 1563 , 1579 , 1581 , 1608 , 1617
<code>\g__draw_path_int</code>	1567 , 1584
E	
<code>\errmessage</code>	38
<code>\evensidemargin</code>	2206
exp commands:	
<code>\exp_after:wN</code>	151 , 457 , 1966
<code>\exp_args:Ne</code>	711 , 2360 , 2879
<code>\exp_args:Nf</code>	1251 , 1375 , 2128
<code>\exp_args:NNf</code>	228 , 276 , 333
<code>\exp_args:Nnx</code>	2115 , 2792
<code>\exp_args:NV</code>	453
<code>\exp_args:Nx</code>	1784 , 1805 , 2072 , 2087 , 2202 , 2758 , 2965 , 2995
<code>\exp_last_unbraced:Nx</code>	462 , 476
<code>\exp_not:N</code> . 514 , 515 , 519 , 520 , 525 , 527 , 666 , 669 , 2440 , 2442 , 2445 , 2475 , 2477 , 2480 , 2632 , 2634 , 2637 , 2643 , 2645 , 2648 , 2685 , 2686 , 2692 , 2693 , 2712 , 2717 , 2826 , 2834 , 2850	
<code>\exp_not:n</code> . . 48 , 89 , 100 , 128 , 2063 , 2068 , 2354 , 2587 , 2588 , 2602 , 2603 , 2615 , 2616 , 2770 , 2775 , 2786 , 2859	
<code>\ExplBackendFileDate</code>	1
F	
file commands:	
<code>\file_compare_timestamp:nNnTF</code> .	1793
<code>\file_parse_full_name:nNNN</code>	1780 , 1803
fp commands:	
<code>\fp_compare:nNnTF</code>	235 , 282 , 288 , 340 , 1418 , 1431 , 1476
<code>\fp_eval:n</code> . 228 , 237 , 250 , 251 , 276 , 293 , 308 , 310 , 333 , 342 , 353 , 354 , 418 , 433 , 434 , 1058 , 1059 , 1060 , 1068 , 1081 , 1082 , 1083 , 1420 , 1425 , 1426 , 1433 , 1443 , 1444 , 1445 , 1446 , 1455 , 1456 , 1457 , 1458 , 1467 , 1468 , 1469 , 1470 , 2351 , 2514 , 2873 , 2966 , 2974 , 2976 , 2996 , 3018 , 3025 , 3086	
<code>\fp_new:N</code>	301 , 302
<code>\fp_set:Nn</code>	281 , 284
<code>\fp_use:N</code>	287 , 291 , 296
<code>\fp_zero:N</code>	283
<code>\c_zero_fp</code> 235 , 282 , 288 , 340 , 1418 , 1431	
G	
graphics commands:	
<code>\graphics_bb_restore:nTF</code> .	1735 , 1949
<code>\graphics_bb_save:n</code>	1764 , 1957
<code>\l_graphics_decodearray_tl</code>	1712 , 1713 , 1723 , 1743 , 1747 , 1748 , 1825 , 1857 , 1858 , 1896 , 1899 , 1900 , 1918 , 1982
<code>\graphics_extract_bb:n</code>	1820 , 1827 , 1977 , 1984

\l_graphics_interpolate_bool . . .	__graphics_backend_include:nn 1986
. 1714 , 1724 , 1742 , 1749 ,	__graphics_backend_include_-
1826 , 1859 , 1895 , 1901 , 1919 , 1983	auxi:nn 1831
\l_graphics_llx_dim	__graphics_backend_include_-
. 1697 , 1836 , 1888 , 1995	auxii:nnn 1831
\l_graphics_lly_dim	__graphics_backend_include_-
. 1698 , 1837 , 1889 , 1996	auxiii:nnn 1831
\l_graphics_name_tl 1798	__graphics_backend_include_-
\l_graphics_page_int	bitmap_quote:w 1960 , 2001
. 1708 , 1728 , 1729 , 1753 ,	__graphics_backend_include_-
1754 , 1818 , 1855 , 1856 , 1882 , 1883 ,	eps:n 1692 , 1773 , 1831 , 1986
1911 , 1924 , 1925 , 1964 , 1965 , 1975	__graphics_backend_include_-
\l_graphics_pagebox_tl	jpg:n 1766 , 1831 , 2001
. 51 , 1709 , 1727 ,	__graphics_backend_include_-
1755 , 1756 , 1819 , 1853 , 1854 , 1884 ,	pdf:n 1766 , 1805 , 1831 , 1960 , 1986
1886 , 1912 , 1933 , 1934 , 1966 , 1976	__graphics_backend_include_pdf_-
\graphics_read_bb:n 1691 , 1814 , 1972	quote:w 1963 , 1968
\l_graphics_urx_dim	__graphics_backend_include_-
. 1699 , 1760 , 1838 , 1890 , 1955 , 1997	png:n 1766 , 1831 , 2001
\l_graphics_ury_dim 1700 , 1761 ,	\l_graphics_backend_name_str . 1773
1839 , 1891 , 1956 , 1998 , 2006 , 2007	\l_graphics_graphics_attr_tl . . .
graphics internal commands: 1705 , 1710 ,
\l_graphics_backend_dir_str . 1773	1717 , 1725 , 1735 , 1762 , 1764 , 1769
\l_graphics_backend_ext_str . 1773	\l_graphics_internal_box
__graphics_backend_getbb_auxi:n 1758 , 1760 , 1761 , 1954 , 1955 , 1956
. 1706	\g_graphics_track_int
__graphics_backend_getbb_- 1830 , 1876 , 1877
auxi:nN 1909	group commands:
__graphics_backend_getbb_-	\group_begin: 143 , 171 , 190
auxii:n 1706	\group_end: 156 , 179
__graphics_backend_getbb_-	\group_insert_after:N 619 , 638 , 649 ,
auxii:nnN 1909	968 , 981 , 996 , 1023 , 1048 , 3007 , 3042
__graphics_backend_getbb_-	
auxiii:nNnn 1909	H
__graphics_backend_getbb_-	hbox commands:
auxiv:nnNnn 1909	\hbox:n 2134 , 2137 ,
__graphics_backend_getbb_-	2209 , 2215 , 2368 , 2375 , 2887 , 2898
auxv:nNnn 1909	\hbox_overlap_right:n 223 ,
__graphics_backend_getbb_-	255 , 271 , 312 , 328 , 356 , 440 , 1296 , 1491
auxvi:nNnn 1950 , 1952	\hbox_set:Nn 1758 , 1954 , 2201 , 2233
__graphics_backend_getbb_eps:n .	\hbox_set:Nw 2184
. 1691 , 1773 , 1814 , 1972	\hbox_set_end: 2199
__graphics_backend_getbb_eps:nm	\hbox_unpack:N 2320
. 1773	
__graphics_backend_getbb_eps:nn	I
. 1784 , 1791	int commands:
__graphics_backend_getbb_jpg:n .	\int_compare:nNnTF
. 1706 , 1814 , 1909 , 1973	508 , 546 , 644 , 949 , 991 , 1728 , 1753 ,
__graphics_backend_getbb_-	1855 , 1882 , 1924 , 1964 , 2292 , 2393 ,
pagebox:w 1909 , 1966	2683 , 2711 , 2824 , 2831 , 2847 , 3048
__graphics_backend_getbb_pdf:n .	\int_const:Nn 149 , 155 , 515 ,
. 1706 , 1799 , 1814 , 1909 , 1980	543 , 572 , 1762 , 1877 , 2032 , 2561 , 2749
__graphics_backend_getbb_png:n .	\int_eval:n 553 , 563 , 592 , 603 ,
. 1706 , 1814 , 1909 , 1973	707 , 716 , 729 , 731 , 735 , 748 , 2661 ,
	2686 , 2693 , 2706 , 2916 , 2924 , 2929

`\int_gincr:N` 197, 363,
 514, 1558, 1603, 1876, 2031, 2100,
 2144, 2218, 2748, 2791, 2804, 2826
`\int_gset:Nn` 172, 191, 2281
`\int_gset_eq:NN` 180, 2145, 2219, 2805
`\int_if_exist:NTF` 1866
`\int_if_odd:nTF` 2204
`\int_new:N` 163, 164,
 410, 505, 511, 1584, 1830, 2027,
 2125, 2156, 2158, 2744, 2801, 2817
`\int_set:Nn` 535
`\int_set_eq:NN` ... 168, 187, 541, 2293
`\int_step_function:nnnN` 733
`\int_use:N` 365,
 396, 525, 536, 681, 817, 862, 935,
 1561, 1567, 1574, 1606, 1614, 1729,
 1754, 1769, 1856, 1869, 1881, 1883,
 1965, 2038, 2103, 2116, 2120, 2148,
 2155, 2223, 2324, 2572, 2582, 2755,
 2793, 2798, 2808, 2816, 2834, 2850
`\int_value:w`
 2440, 2475, 2632, 2643, 2661
`\int_zero:N` ... 1708, 1818, 1911, 1975

K

kernel internal commands:

`__kernel_backend_align_begin:` ..
 64, 208, 232, 247
`__kernel_backend_align_end:` ...
 64, 222, 240, 254
`\g__kernel_backend_header_bool` ..
 57, 663
`__kernel_backend_literal:n` . 46,
 52, 55, 62, 66, 73, 76, 78, 134, 137,
 139, 141, 161, 337, 350, 522, 550,
 560, 615, 622, 648, 654, 677, 813,
 995, 1001, 1003, 1022, 1047, 1114,
 1120, 1415, 1422, 1428, 1488, 1493,
 1694, 1833, 1868, 1878, 1992, 2003,
 2738, 2854, 2916, 2920, 2925, 2930
`__kernel_backend_literal_page:n`
 92, 136, 2732, 2734, 2935, 2937
`__kernel_backend_literal_pdf:n` .
 .. 81, 133, 263, 320, 1305, 3057, 3072
`__kernel_backend_literal_-`
`postscript:n`
 51, 67, 68, 72, 209, 210, 212,
 213, 221, 233, 248, 1110, 2395, 2407
`__kernel_backend_literal_svg:n` .
 160, 167, 178, 186,
 196, 364, 366, 383, 1499, 1672, 1683
`__kernel_backend_matrix:n`
 120, 285, 306, 1405

`__kernel_backend_postscript:n` ..
 54, 617,
 1026, 1028, 1030, 1034, 2021, 2077,
 2092, 2134, 2140, 2177, 2209, 2216,
 2220, 2234, 2262, 2306, 2313, 2319,
 2327, 2334, 2368, 2375, 2970, 2979
`__kernel_backend_scope:n`
 165, 393, 398, 1088, 1504, 3086
`__kernel_backend_scope_begin:` ..
 75, 102, 138,
 165, 207, 231, 246, 262, 279, 305,
 319, 336, 349, 1311, 1483, 1503, 1670
`__kernel_backend_scope_begin:n` .
 165, 385, 413, 426
`__kernel_backend_scope_end:` ...
 75, 102, 138, 165, 224, 242,
 256, 272, 299, 313, 329, 345, 357,
 408, 422, 441, 1312, 1495, 1506, 1684
`\g__kernel_backend_scope_int` ...
 163, 170, 172, 177, 181, 189, 191, 197
`\l__kernel_backend_scope_int` ...
 163, 169, 182, 188
`__kernel_color_backend_stack_-`
`init:Nnn` 508, 570, 2986
`__kernel_color_backend_stack_-`
`pop:n` 546, 584, 641, 3014
`__kernel_color_backend_stack_-`
`push:nn`
 .. 546, 584, 637, 966, 979, 3005, 3040
`__kernel_dependency_version_-`
`check:Nn` 1
`__kernel_dependency_version_-`
`check:nn` 27, 29
`__kernel_kern:n`
 2139, 2141, 2367, 2371,
 2374, 2378, 2886, 2894, 2897, 2913
`\c__kernel_sys_dvipdfmx_version_-`
`int` 143, 508, 546,
 644, 949, 991, 2824, 2831, 2847, 3048

M

math commands:

`\c_math_toggle_token` 2187, 2197
`\MessageBreak` 40

mode commands:

`\mode_if_horizontal:TF` ... 2283, 2290
`\mode_if_math:TF` 2181

O

`\oddsidemargin` 2205

opacity internal commands:

`__opacity_backend:nn` ... 2973, 3079
`__opacity_backend_fill:n`
 2973, 3015, 3079

_opacity_backend_fill_stroke:nn	3017, 3023, 3027, 3045, 3059
\l_opacity_backend_fill_tl . . .	2991, 3000, 3024, 3032, 3052, 3064
_opacity_backend_fillstroke:nn	3015
_opacity_backend_reset: 2993, 3042	
_opacity_backend_select:n . . .	2963, 2993, 3048, 3079
_opacity_backend_select_aux:n .	2963, 2993, 3030, 3050, 3062
\c_opacity_backend_stack_int . .	2984, 3005, 3014, 3040
_opacity_backend_stroke:n . . .	2973, 3015, 3079
\l_opacity_backend_stroke_tl . .	2991, 3001, 3019, 3033, 3053, 3065
P	
pdf commands:	
\pdf_object_if_exist:nTF	873
\pdf_object_new:nn	875
\pdf_object_ref:n	888
\pdf_object_ref_last:	854, 863, 927, 936
\pdf_object_unnamed_write:nn . .	842, 869, 893, 900
\pdf_object_write:nn	876
pdf internal commands:	
_pdf_backend:n	2737, 2741, 2743, 2769, 2774, 2783, 2806, 2828, 2844, 2857, 2889, 2890, 2900
_pdf_backend_annotation:nnnn . .	2126, 2425, 2802
_pdf_backend_annotation_-aux:nnnn	2128, 2131
\g_pdf_backend_annotation_int . .	2125, 2145, 2155, 2801, 2805, 2816
_pdf_backend_annotation_last: .	2154, 2438, 2815
_pdf_backend_bdc:nn	2419, 2731, 2934, 2956
_pdf_backend_catalog_gput:nn . .	2023, 2531, 2740, 2940
_pdf_backend_compress_objects:n	2391, 2652, 2915, 2950
_pdf_backend_compresslevel:n . .	2391, 2652, 2915, 2950
\l_pdf_backend_content_box 2123,	2184, 2208, 2211, 2213, 2242, 2253
_pdf_backend_destination:nn . .	2332, 2494, 2855
_pdf_backend_destination:nnnn .	2332, 2494, 2855
_pdf_backend_destination_-aux:nnnn	2332, 2855
_pdf_backend_emc:	2419, 2731, 2934, 2956
_pdf_backend_info_gput:nn . . .	2023, 2531, 2740, 2940
_pdf_backend_link:nw	2165
_pdf_backend_link_aux:nw . . .	2165
_pdf_backend_link_begin:n . .	2818
_pdf_backend_link_begin:nnnw 2449	
_pdf_backend_link_begin:nw . .	2166, 2168, 2169
_pdf_backend_link_begin_aux:nw	2172, 2174
_pdf_backend_link_begin_-goto:nnw	2165, 2449, 2818
_pdf_backend_link_begin_-user:nnw	2165, 2449, 2818
\g_pdf_backend_link_bool	2160, 2171, 2176, 2191, 2229
\g_pdf_backend_link_dict_tl . .	2157, 2179, 2224
_pdf_backend_link_end:	2165, 2449, 2818
_pdf_backend_link_end_aux: .	2165
\g_pdf_backend_link_int	2156, 2219, 2223, 2324, 2817, 2826, 2834, 2850
_pdf_backend_link_last:	2323, 2473, 2845
_pdf_backend_link_margin:n . .	2325, 2484, 2853
\g_pdf_backend_link_math_bool .	2159, 2182, 2183, 2186, 2196
_pdf_backend_link_minima: . .	2165
_pdf_backend_link_outerbox:n	2165
\g_pdf_backend_link_sf_int . . .	2158, 2281, 2292, 2293
_pdf_backend_link_sf_restore: 2165	
_pdf_backend_link_sf_save: .	2165
\l_pdf_backend_model_box . 2124,	2201, 2233, 2241, 2252, 2267, 2269
_pdf_backend_objcompresslevel:n	2652
\g_pdf_backend_object_int	2027, 2031, 2034, 2100, 2103, 2116, 2120, 2144, 2145, 2148, 2218, 2219, 2744, 2748, 2751, 2791, 2793, 2798, 2804, 2805, 2808
_pdf_backend_object_last: . . .	2119, 2630, 2797, 2942
_pdf_backend_object_new:nn . .	2029, 2552, 2746, 2942

_pdf_backend_object_now:nn . . .	pdf.brokenlink.dict	3175
. 2098, 2604, 2789, 2942	pdf.brokenlink.rect	3175
_g_pdf_backend_object_prop . . .	pdf.brokenlink.skip	3175
. 2027, 2035, 2046, 2056,	pdf.count	3313
2551, 2569, 2585, 2744, 2752, 2759	pdf.currentrect	3313
_pdf_backend_object_ref:n 2029,	pdf.cvs	3097
2043, 2057, 2552, 2746, 2765, 2942	pdf.dest.anchor	3140
_pdf_backend_object_write:nn . .	pdf.dest.point	3140
. 2039, 2573, 2756, 2942	pdf.dest.x	3140
_pdf_backend_object_write:nnn 2756	pdf.dest.y	3140
_pdf_backend_object_write_-	pdf.dest2device	3140
array:nn 2039, 2756	pdf.dev.x	3140
_pdf_backend_object_write_-	pdf.dev.y	3140
dict:nn 2039, 2756	pdf.dvi.pt	3097
_pdf_backend_object_write_-	pdf.globaldict	3094
fstream:nn 2039, 2756	pdf.leftboundary	3175
_pdf_backend_object_write_-	pdf.link.dict	2165
fstream:nnn 2073, 2075	pdf.linkdp.pad	2165, 3101
_pdf_backend_object_write_-	pdf.linkht.pad	2165, 3101
stream:nn 2039, 2756	pdf.linkmargin	3101
_pdf_backend_object_write_-	pdf.llx	2165, 3104
stream:nnn 2039	pdf.lly	2165, 3104
_pdf_backend_object_write_-	pdf.originx	3175
stream:nnnn 2756	pdf.originy	3175
_pdf_backend_pageobject_ref:n .	pdf.outerbox	2165, 3417
. 2121, 2641, 2799, 2942	pdf.pdfmark	3417
_pdf_backend_pdfmark:n	pdf.pdfmark.dict	3417
2020, 2024, 2026, 2041, 2062, 2067,	pdf.pdfmark.good	3417
2101, 2146, 2335, 2379, 2420, 2422	pdf.pt.dvi	3097
_pdf_backend_version_major: . . .	pdf.rect	3104
. 2417, 2708, 2924, 2925, 2932, 2954	pdf.rect.ht	3097
_pdf_backend_version_major_-	pdf.rightboundary	3175
gset:n 2415, 2680, 2922, 2952	pdf.save.linkll	3104
_pdf_backend_version_minor: . . .	pdf.save.linkur	3104
. 2417, 2708, 2929, 2930, 2932, 2954	pdf.save.ll	3104
_pdf_backend_version_minor_-	pdf.save.ur	3104
gset:n 2415, 2680, 2922, 2952	pdf.tmpa	3140
\l_pdf_breaklink_pdfmark_tl . . .	pdf.tmpb	3140
. 2161, 2226, 2318	pdf.tmpc	3140
_pdf_breaklink_postscript:n . . .	pdf.tmpd	3140
. 2163, 2210, 2212, 2319	pdf.urx	3104
_pdf_breaklink_usebox:N	pdf.ury	2165, 3104
. 2164, 2211, 2320	pdfmanagement commands:	
_pdf_exp_not_i:nn . 2573, 2619, 2624	_pdfmanagement_add:nnn	856,
_pdf_exp_not_ii:nn 2573, 2620, 2625	860, 929, 933, 2984, 2988, 3002,	
\l_pdf_internal_box 2018	3009, 3034, 3037, 3054, 3066, 3069	
pdf.baselineskip 2165, 3417	prg commands:	
pdf.bordertracking 3175	_prg_replicate:nn	
pdf.bordertracking.begin 3175 176, 705, 726, 736, 906	
pdf.bordertracking.continue 3175	prop commands:	
pdf.bordertracking.end 3175	_prop_gput:Nnn 2035, 2569, 2752	
pdf.bordertracking.endpage 3175	_prop_item:Nn . 2046, 2056, 2585, 2759	
pdf.breaklink 3313	_prop_new:N 2028, 2551, 2745	
pdf.breaklink.write 3313	_ProvidesExplFile 2	

Q	
quark commands:	
\q_stop	144, 152
quark internal commands:	
\s__color_stop	
.	463, 466, 477, 480, 716, 717,
.	721, 725, 738, 741, 745, 749, 763,
.	907, 940, 944, 1051, 1053, 1074, 1076
\s__graphics_stop	
.	1963, 1968, 2008, 2012
S	
scan commands:	
\scan_stop:	
.	105, 114, 603, 2467, 2492, 2515,
.	2529, 2661, 2678, 2686, 2693, 2706
separation	<u>3091</u>
skip commands:	
\skip_horizontal:n	225, 273, 330
str commands:	
\c_hash_str	396, 1567, 1574, 1614
\c_percent_str	1094, 1095, 1096
\str_case:nn	912, 2105, 2612
\str_case:nnTF	2339, 2503, 2862
\str_case_e:nn	2045, 2584
\str_convert_pdfname:n	684, 853
\str_if_eq:nnTF	
.	482, 485, 488, 491, 3029, 3061
\str_new:N	1775, 1776, 1777
\str_tail:N	1786, 1807
sys commands:	
\sys_get_shell:nnNTF	145
\sys_if_shell:TF	1773
\sys_shell_now:n	1795
sys internal commands:	
\l_sys_internal_tl	147, 151
__sys_tmp:w	144, 151
T	
TeX and L ^A T _E X 2 _ε commands:	
\@cclv	2302, 2304, 2312
\@makecol@hook	2296
\current@color	14, 453, 457, 463, 477
\special	2
tex commands:	
\tex_baselineskip:D	2273
\tex_endinput:D	44
\tex_global:D	
.	2654, 2671, 2685, 2692, 2699
\tex_immediate:D	
.	1740, 2576, 2579, 2607, 2610
\tex luatexversion:D	2683, 2711
\tex_pdfannot:D	2431
\tex_pdfcatalog:D	2537
\tex_pdfcolorstack:D	590, 601
\tex_pdfcolorstackinit:D	578
\tex_pdfcompresslevel:D	2659
\tex_pdffdest:D	2500, 2523
\tex_pdfendlink:D	2470
\tex_pdfextension:D	
.	84, 95, 105, 114, 123,
.	587, 598, 2428, 2456, 2467, 2497,
.	2520, 2534, 2544, 2555, 2576, 2607
\tex_pdffeedback:D	
.	575, 2442, 2477, 2564, 2634, 2645
\tex_pdfinfo:D	2547
\tex_pdflastannot:D	2445
\tex_pdflastlink:D	2480
\tex_pdflastobj:D	2567, 2637
\tex_pdflastximage:D	1759, 1763
\tex_pdflinkmargin:D	2490
\tex_pdfliteral:D	87, 98
\tex_pdfmajorversion:D	
.	2690, 2692, 2716, 2717
\tex_pdfminorversion:D	2704, 2728
\tex_pdfobj:D	2558, 2579, 2610
\tex_pdfobjcompresslevel:D	2676
\tex_pdfpageref:D	2648
\tex_pdfrefximage:D	1759, 1768
\tex_pdfrestore:D	117
\tex_pdfsave:D	108
\tex_pdfsetmatrix:D	126
\tex_pdfstartlink:D	2459
\tex_pdfvariable:D	2487,
.	2656, 2673, 2685, 2701, 2712, 2725
\tex_pdfximage:D	1740
\tex_spacefactor:D	2284, 2293
\tex_special:D	46
\tex_the:D	1763, 2712, 2717, 2723
\tex_vss:D	2369, 2376, 2892, 2911
\tex_XeTeXpdffile:D	1920, 1962
\tex_XeTeXpicfile:D	1913
TeXcolorseparation	<u>3091</u>
\textwidth	2268
tl commands:	
\c_space_tl	287, 292, 295, 526,
.	772, 980, 1543, 1696, 1697, 1698,
.	1699, 1835, 1836, 1837, 1838, 1883,
.	1886, 1888, 1889, 1890, 1891, 1963,
.	1965, 1994, 1995, 1996, 1997, 2224,
.	2447, 2482, 2639, 2650, 2808, 2835
\tl_clear:N	1709, 1717, 1723,
.	1819, 1825, 1912, 1918, 1976, 1982
\tl_gclear:N	1581, 1617
\tl_gset:Nn	1540, 2179
\tl_if_blank:nTF	
.	527, 580, 720, 737, 744, 762, 846, 943

\tl_if_empty:NTF . 1543, 1712, 1747, 1755, 1853, 1857, 1884, 1899, 1933	
\tl_if_empty:nTF 1637	
\tl_if_empty_p:N 1743, 1896	
\tl_if_head_is_space:nTF 453	
\tl_new:N 625, 626, 1547, 1705, 2157, 2161, 2991, 2992	
\tl_put_right:Nn 2300	
\tl_set:Nn . 455, 467, 483, 486, 489, 493, 496, 635, 636, 965, 978, 1710, 1725, 1798, 2162, 2318, 3000, 3001, 3032, 3033, 3052, 3053, 3064, 3065	
\tl_to_str:n 2033, 2038, 2562, 2572, 2583, 2750, 2755	
\tl_use:N 804, 881	
	U
	use commands:
	\use:N 43, 2055, 2115, 2764, 2792
	\use:n . 61, 457, 493, 516, 520, 667, 858, 931, 1055, 1065, 1078, 1251, 1375, 1440, 1452, 1464, 1622, 1940
	\use_none:n 1637, 1639, 2296
	V
	\value 2204
	vbox commands:
	\vbox_set:Nn 2304
	\vbox_to_zero:n 2365, 2372, 2884, 2895
	\vbox_unpack_drop:N 2312