

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}{2021-02-18}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2021-02-18}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2021-02-18}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2021-02-18}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2021-02-18}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2021-02-18}{ }
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 TeX magnification. Thus for any operation involving absolute lengths there is a correction to make. See **normalscale** from **special.pro** for the variables, noting that here everything is saved on the stack rather than as a separate variable. Once all of that is done, the actual clipping is trivial.

```

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
`__box_backend_rotate_aux:Nn` 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_T_EX and pdf_T_EX

```

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
`_color_backend_pickup:w`

```

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` pdfTeX, LuaTeX 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         \cs_if_exist:NTF \AtBeginDvi
517         { \exp_not:N \AtBeginDvi }
518         { \exp_not:N \use:n }

```

```

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

```

(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.

```

542 \int_compare:nNnF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
543 {
544     \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
545     {
546         \__kernel_backend_literal:x
547         {
548             pdfcolorstack ~
549             \int_eval:n {#1} ~
550             push ~ (#2)
551         }
552     }
553     \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
554     \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
555     {
556         \__kernel_backend_literal:x
557         {
558             pdfcolorstack ~
559             \int_eval:n {#1} ~
560             pop
561         }
562     }
563 }

```

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

564 </dvipdfmx | xetex>

3.2.3 LuaTeX and pdfTeX

565 $\langle *luatex \mid pdftex \rangle$

$__kernel_color_backend_stack_init:Nnn$

```

566 \cs_new_protected:Npn \_\_kernel_color_backend_stack_init:Nnn #1#2#3
567 {
568   \int_const:Nn #1
569   {
570      $\langle *luatex \rangle$ 
571     \tex_pdffeedback:D colorstackinit ~
572    $\langle /luatex \rangle$ 
573      $\langle *pdftex \rangle$ 
574     \tex_pdfcolorstackinit:D
575    $\langle /pdftex \rangle$ 
576     \tl_if_blank:nF {#2} { #2 ~ }
577     {#3}
578   }
579 }
```

(End definition for $__kernel_color_backend_stack_init:Nnn$.)

$__kernel_color_backend_stack_push:nn$

$__kernel_color_backend_stack_push:nx$

$__kernel_color_backend_stack_pop:n$

```

580 \cs_new_protected:Npn \_\_kernel_color_backend_stack_push:nn #1#2
581 {
582    $\langle *luatex \rangle$ 
583   \tex_pdfextension:D colorstack ~
584  $\langle /luatex \rangle$ 
585    $\langle *pdftex \rangle$ 
586   \tex_pdfcolorstack:D
587  $\langle /pdftex \rangle$ 
588   \int_eval:n {#1} ~ push ~ {#2}
589 }
590 \cs_generate_variant:Nn \_\_kernel_color_backend_stack_push:nn { nx }
591 \cs_new_protected:Npn \_\_kernel_color_backend_stack_pop:n #1
592 {
593    $\langle *luatex \rangle$ 
594   \tex_pdfextension:D colorstack ~
595  $\langle /luatex \rangle$ 
596    $\langle *pdftex \rangle$ 
597   \tex_pdfcolorstack:D
598  $\langle /pdftex \rangle$ 
599   \int_eval:n {#1} ~ pop \scan_stop:
600 }
```

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

601 $\langle /luatex \mid pdftex \rangle$

3.3 General color

3.3.1 dvips-style

602 $\langle *dvips \mid dvisvgm \rangle$

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

```

\_color_backend_select_gray:n
\_color_backend_select_rgb:n
\_color_backend_select:n
\_color_backend_reset:
    color.sc
603 \cs_new_protected:Npn \_color_backend_select_cmyk:n #1
604 { \_color_backend_select:n { cmyk ~ #1 } }
605 \cs_new_protected:Npn \_color_backend_select_gray:n #1
606 { \_color_backend_select:n { gray ~ #1 } }
607 \cs_new_protected:Npn \_color_backend_select_rgb:n #1
608 { \_color_backend_select:n { rgb ~ #1 } }
609 \cs_new_protected:Npn \_color_backend_select:n #1
610 {
611     \_kernel_backend_literal:n { color~push~ #1 }
612     <dvips>
613     \_kernel_backend_postscript:n { /color.sc ~ { } ~ def }
614     </dvips>
615     \group_insert_after:N \_color_backend_reset:
616 }
617 \cs_new_protected:Npn \_color_backend_reset:
618 { \_kernel_backend_literal:n { color~pop } }
```

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

```
619 </dvips | dvisvgm>
```

3.3.2 LuaTeX and pdfTeX

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

```

\_l_color_backend_fill_tl
\_l_color_backend_stroke_tl
621 \tl_new:N \l_color_backend_fill_tl
622 \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` Store the values then pass to the stack.

```

\_color_backend_select_gray:n
\_color_backend_select_rgb:n
\_color_backend_select:nn
\_color_backend_reset:
623 \cs_new_protected:Npn \_color_backend_select_cmyk:n #1
624 { \_color_backend_select:nn { #1 ~ k } { #1 ~ K } }
625 \cs_new_protected:Npn \_color_backend_select_gray:n #1
626 { \_color_backend_select:nn { #1 ~ g } { #1 ~ G } }
627 \cs_new_protected:Npn \_color_backend_select_rgb:n #1
628 { \_color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
629 \cs_new_protected:Npn \_color_backend_select:nn #1#2
630 {
631     \tl_set:Nn \l_color_backend_fill_tl {#1}
632     \tl_set:Nn \l_color_backend_stroke_tl {#2}
633     \_kernel_color_backend_stack_push:nn \l_color_backend_stack_int { #1 ~ #2 }
634     \group_insert_after:N \_color_backend_reset:
635 }
636 \cs_new_protected:Npn \_color_backend_reset:
637 { \_kernel_color_backend_stack_pop:n \l_color_backend_stack_int }
```

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

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

3.3.3 dvipdfmx/X_YTeX

639 \langle *dvipdfmx | xetex \rangle

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.

```

640 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
641 {
642   \cs_gset_protected:Npn \__color_backend_select_cmyk:n #1
643   {
644     \__kernel_backend_literal:n { pdf: bc ~ [#1] }
645     \group_insert_after:N \__color_backend_reset:
646   }
647   \cs_gset_eq:NN \__color_backend_select_gray:n \__color_backend_select_cmyk:n
648   \cs_gset_eq:NN \__color_backend_select_rgb:n \__color_backend_select_cmyk:n
649   \cs_gset_protected:Npn \__color_backend_reset:
650   { \__kernel_backend_literal:n { pdf: ec } }
651 }

```

(End definition for __color_backend_select_cmyk:n and others.)

652 \langle /dvipdfmx | xetex \rangle

3.4 Separations

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

653 \langle *dvips \rangle

```

\__color_backend_select_separation:nn
\__color_backend_select_devicen:nn
654 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
655 { \__color_backend_select:n { separation ~ #1 ~ #2 } }
656 \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.

```

657 \cs_new_protected:Npx \__color_backend_separation_init:nnnnn #1#2#3#4#5
658 {
659   \bool_if:NT \g__kernel_backend_header_bool
660   {
661     \cs_if_exist:NTF \AtBeginDvi
662     { \exp_not:N \AtBeginDvi }
663     { \use:n }
664     {
665       \exp_not:N \__color_backend_separation_init_aux:nnnnn
666       {#1} {#2} {#3} {#4} {#5}

```

```

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

```

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.

```

705 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3
706 {
707   \exp_args:Ne \__color_backend_separation_init:nnnnn
708   { \__color_backend_separation_init_count:n {#2} }
709   {#1} {#2} {#3}
710 }
711 \cs_new:Npn \__color_backend_separation_init_count:n #1
712 { \int_eval:n { 0 \__color_backend_separation_init_count:w #1 ~ \s__color_stop } }
713 \cs_new:Npn \__color_backend_separation_init_count:w #1 ~ #2 \s__color_stop
714 {
715   +1

```

```

716     \tl_if_blank:nF {#2}
717     { \__color_backend_separation_init_count:w #2 \s__color_stop }
718 }

```

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.

```

719 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
720 {
721     \__color_backend_separation_init:w #3 ~ \s__color_stop #4 ~ \s__color_stop
722     \prg_replicate:nn {#1}
723     {
724         pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
725         \int_eval:n { 3 * #1 } ~ index ~ mul ~
726         2 ~ index ~ add ~
727         \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
728     }
729     \int_step_function:nnnN {#1} { -1 } { 1 }
730     \__color_backend_separation_init:n
731     \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
732     \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
733     \tl_if_blank:nF {#2}
734     { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
735 }
736 \cs_new:Npn \__color_backend_separation_init:w
737 #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
738 {
739     #1 ~ #3 ~ 0 ~
740     \tl_if_blank:nF {#2}
741     { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }
742 }
743 \cs_new:Npn \__color_backend_separation_init:n #1
744 { \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.

```

745 \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s__color_stop
746 {
747     #2 ~ #3 ~
748     2 ~ index ~ 2 ~ index ~ lt ~
749     { ~ pop ~ exch ~ pop ~ } ~
750     { ~
751         2 ~ index ~ 1 ~ index ~ gt ~
752         { ~ exch ~ pop ~ exch ~ pop ~ } ~
753         { ~ pop ~ pop ~ } ~

```

```

754         ifelse ~
755     }
756     ifelse ~
757     #1 ~ 1 ~ roll ~
758     \tl_if_blank:nF {#4}
759     { \_color_backend_separation_init:nw {#1} #4 \s_color_stop }
760 }

```

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.

```

761 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
762 {
763     \_color_backend_separation_init:nxxxnn
764     {#2}
765     {
766         /CIEBasedABC ~
767         << ~
768         /RangeABC ~ [ ~ \c_color_model_range_CIELAB_tl \c_space_tl ] ~
769         /DecodeABC ~
770         [ ~
771         { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
772         { ~ 500 ~ div ~ } ~ bind ~
773         { ~ 200 ~ div ~ } ~ bind ~
774         ] ~
775         /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
776         /DecodeLMN ~
777         [ ~
778         { ~
779             dup ~ 6 ~ 29 ~ div ~ ge ~
780             { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
781             { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
782             ifelse ~
783             0.9505 ~ mul ~
784         } ~ bind ~
785         { ~
786             dup ~ 6 ~ 29 ~ div ~ ge ~
787             { ~ dup ~ dup ~ mul ~ mul ~ } ~
788             { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
789             ifelse ~
790             } ~ bind ~
791             { ~
792                 dup ~ 6 ~ 29 ~ div ~ ge ~
793                 { ~ dup ~ dup ~ mul ~ mul ~ } ~
794                 { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
795                 ifelse ~
796                 1.0890 ~ mul ~
797             } ~ bind
798         ] ~
799         /WhitePoint ~
800         [ ~ \tl_use:c { c_color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
801         >>
802     }
803     { \c_color_model_range_CIELAB_tl }
804     { 100 ~ 0 ~ 0 }

```

```

805     {#3}
806   }

```

(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.

```

807 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
808 {
809   \_kernel_backend_literal:e
810   {
811     !
812     TeXDict ~ begin ~
813     /color \int_use:N \g\_color_model_int
814     {
815       [ ~
816         /DeviceN ~
817         [ ~ #1 ~ ] ~
818         #2 ~
819         { ~ #3 ~ } ~
820       ] ~ setcolorspace
821     } ~ def ~
822   end
823 }
824 }

```

(End definition for `_color_backend_devicen_init:nnn`.)

```

825 </dvips>
826 <*dvisvgm>

```

`_color_backend_select_separation:nn` No support at present.

```

827 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2 { }
828 \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.

```

829 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }
830 \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:nnnnn`.)

```

831 </dvisvgm>
832 <*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.

```

833 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
834 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
835 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn

```

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

```

__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.

```

836 \cs_new_protected:Npn __color_backend_separation_init:nnnn #1#2#3#4#5
837 {
838   \pdf_object_unnamed_write:nx { dict }
839   {
840     /FunctionType ~ 2
841     /Domain ~ [0 ~ 1]
842     \tl_if_blank:nF {#3} { /Range ~ [#3] }
843     /C0 ~ [#4] ~
844     /C1 ~ [#5] /N ~ 1
845   }
846   __color_backend_separation_init:n
847   {
848     /Separation ~
849     / \str_convert_pdfname:n {#1} ~ #2 ~
850     \pdf_object_ref_last:
851   }
852   \cs_if_exist:NT \pdfmanagement_add:nnn
853   {
854     \use:x
855     {
856       \pdfmanagement_add:nnn
857       { Page / Resources / ColorSpace }
858       { color \int_use:N \g__color_model_int }
859       { \pdf_object_ref_last: }
860     }
861   }
862 }
863 \cs_new_protected:Npn __color_backend_separation_init:n #1
864 {
865   \pdf_object_unnamed_write:nx { array } {#1}
866 }

```

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

```

867 \cs_new_protected:Npn __color_backend_separation_init_CIELAB:nnn #1#2#3
868 {
869   \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
870   {
871     \pdf_object_new:nn { __color_illuminant_CIELAB_ #1 } { array }
872     \pdf_object_write:nx { __color_illuminant_CIELAB_ #1 }
873     {
874       /Lab ~
875       <<
876       /WhitePoint ~
877       [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
878       /Range ~ [ \c__color_model_range_CIELAB_tl ]
879       >>
880     }
881   }
882   __color_backend_separation_init:nnnn
883   {#2}

```



```

884     { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
885     { \c__color_model_range_CIELAB_t1 }
886     { 100 ~ 0 ~ 0 }
887     {#3}
888   }
889   \cs_if_exist:NF \pdf_object_unnamed_write:nn
890   {
891     \cs_gset_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
892     { }
893   }

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

```

894   \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
895   {
896     \pdf_object_unnamed_write:nx { stream }
897     {
898       {
899         /FunctionType ~ 4 ~
900         /Domain ~
901         [ ~
902           \prg_replicate:nn
903           { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
904           { 0 ~ 1 ~ } ~
905         ] ~
906         /Range ~
907         [ ~
908           \str_case:nn {#2}
909           {
910             { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
911             { /DeviceGray } { 0 ~ 1 }
912             { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
913           } ~
914         ]
915       }
916       {#3}
917     }
918     \__color_backend_separation_init:n
919     {
920       /DeviceN ~
921       [ ~ #1 ~ ] ~
922       #2 ~
923       \pdf_object_ref_last:
924     }
925     \cs_if_exist:NT \pdfmanagement_add:nnn
926     {
927       \use:x
928       {
929         \pdfmanagement_add:nnn
930         { Page / Resources / ColorSpace }
931         { color \int_use:N \g__color_model_int }

```

```

932         { \pdf_object_ref_last: }
933     }
934 }
935 }
936 \cs_new:Npn \__color_backend_devicen_init:w #1 ~ #2 \s__color_stop
937 {
938     + 1
939     \tl_if_blank:nF {#2}
940     { \__color_backend_devicen_init:w #2 \s__color_stop }
941 }
942 \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.)
943 </dvipdfmx | luatex | pdftex | xetex>
944 <*dvipdfmx | xetex>

```

__color_backend_select_separation: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.

```

945 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
946 {
947     \cs_gset_protected:Npn \__color_backend_select_separation:nn #1#2 { }
948     \cs_gset_eq:NN \__color_backend_select_devicen:nn
949         \__color_backend_select_separation:nn
950 }
(End definition for \__color_backend_select_separation:nn and \__color_backend_select_devicen:nn.)
951 </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.

```

952 <*dvipdfmx | luatex | pdftex | xetex>

```

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.

```

953 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
954 { \__color_backend_fill:n { #1 ~ k } }
955 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
956 { \__color_backend_fill:n { #1 ~ g } }
957 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
958 { \__color_backend_fill:n { #1 ~ rg } }
959 \cs_new_protected:Npn \__color_backend_fill:n #1
960 {
961     \tl_set:Nn \l__color_backend_fill_tl {#1}
962     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
963     { #1 ~ \l__color_backend_stroke_tl }

```

```

964 \group_insert_after:N \__color_backend_reset:
965 }
966 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
967 { \__color_backend_stroke:n { #1 ~ K } }
968 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
969 { \__color_backend_stroke:n { #1 ~ G } }
970 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
971 { \__color_backend_stroke:n { #1 ~ RG } }
972 \cs_new_protected:Npn \__color_backend_stroke:n #1
973 {
974   \tl_set:Nn \l__color_backend_stroke_tl {#1}
975   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
976   { \l__color_backend_fill_tl \c_space_tl #1 }
977   \group_insert_after:N \__color_backend_reset:
978 }

```

(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
979 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
980 { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
981 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
982 { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
983 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
984 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

985 </dvipdfmx | luatex | pdftex | xetex>
986 <*dvipdfmx | xetex>

```

Deal with older (x)dvipdfmx.

```

987 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
988 {
989   \cs_gset_protected:Npn \__color_backend_fill_cmyk:n #1
990   {
991     \__kernel_backend_literal:n { pdf: bc ~ [#1] }
992     \group_insert_after:N \__color_backend_reset:
993   }
994   \cs_gset_eq:NN \__color_backend_fill_gray:n \__color_backend_fill_cmyk:n
995   \cs_gset_eq:NN \__color_backend_fill_rgb:n \__color_backend_fill_cmyk:n
996   \cs_gset_protected:Npn \__color_backend_reset:
997   { \__kernel_backend_literal:n { pdf: ec } }
998   \cs_gset_protected:Npn \__color_backend_stroke:n #1
999   { \__kernel_backend_literal:n {#1} }
1000   \cs_gset_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1001   \cs_gset_eq:NN \__color_backend_fill_devicen:nn
1002   \__color_backend_fill_separation:nn
1003   \cs_gset_eq:NN \__color_backend_stroke_separation:nn
1004   \__color_backend_fill_separation:nn
1005   \cs_gset_eq:NN \__color_backend_stroke_devicen:nn
1006   \__color_backend_stroke_separation:nn
1007 }

```

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

1008 `</dviptdpmx | xetex>`

1009 `<*dvips>`

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

```

1010 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1011 { \_color_backend_fill:n { cmyk ~ #1 } }
1012 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1013 { \_color_backend_fill:n { gray ~ #1 } }
1014 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1015 { \_color_backend_fill:n { rgb ~ #1 } }
1016 \cs_new_protected:Npn \_color_backend_fill:n #1
1017 {
1018   \_kernel_backend_literal:n { color~push~ #1 }
1019   \group_insert_after:N \_color_backend_reset:
1020 }
1021 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1022 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1023 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1024 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1025 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1026 { \_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
1027 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1028 { \_color_backend_fill:n { separation ~ #1 ~ #2 } }
1029 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1030 { \_kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1031 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1032 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

1033 `</dvips>`

1034 `<*dvisvgm>`

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

```

1035 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1036 { \_color_backend_fill:n { cmyk ~ #1 } }
1037 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1038 { \_color_backend_fill:n { gray ~ #1 } }
1039 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1040 { \_color_backend_fill:n { rgb ~ #1 } }
1041 \cs_new_protected:Npn \_color_backend_fill:n #1
1042 {
1043   \_kernel_backend_literal:n { color~push~ #1 }
1044   \group_insert_after:N \_color_backend_reset:
1045 }

```

(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).

```

1046 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1047 { \__color_backend_cmyk:w #1 \s__color_stop }
1048 \cs_new_protected:Npn \__color_backend_stroke_cmyk:w
1049 #1 ~ #2 ~ #3 ~ #4 \s__color_stop
1050 {
1051   \use:x
1052   {
1053     \__color_backend:nnn
1054     { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1055     { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1056     { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }
1057   }
1058 }
1059 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1060 {
1061   \use:x
1062   {
1063     \__color_backend_stroke_gray_aux:n
1064     { \fp_eval:n { 100 * (#1) } }
1065   }
1066 }
1067 \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1068 { \__color_backend:nnn {#1} {#1} {#1} }
1069 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1070 { \__color_backend_rgb:w #1 \s__color_stop }
1071 \cs_new_protected:Npn \__color_backend_stroke_rgb:w
1072 #1 ~ #2 ~ #3 \s__color_stop
1073 {
1074   \use:x
1075   {
1076     \__color_backend:nnn
1077     { \fp_eval:n { 100 * (#1) } }
1078     { \fp_eval:n { 100 * (#2) } }
1079     { \fp_eval:n { 100 * (#3) } }
1080   }
1081 }
1082 \cs_new_protected:Npx \__color_backend:nnn #1#2#3
1083 {
1084   \__kernel_backend_scope:n
1085   {
1086     stroke =
1087     "
1088     rgb
1089     (
1090       #1 \c_percent_str ,
1091       #2 \c_percent_str ,
1092       #3 \c_percent_str
1093     )
1094     "
1095   }
1096 }

```

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

At present, these are no-ops.

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
1097 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1098 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2 { }
1099 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1100 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn
(End definition for \_color_backend_fill_separation:nn and others.)
1101 </dvisvgm>
1102 </package>

```

4 l3backend-draw Implementation

```

1103 <*package>
1104 <@@=draw>

```

4.1 dvips backend

```

1105 <*dvips>

```

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

```

\_draw_backend_literal:n
\_draw_backend_literal:x
1106 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
1107 \cs_generate_variant:Nn \_draw_backend_literal:n { x }

```

(End definition for `_draw_backend_literal:n`.)

```

\_draw_backend_begin:
  \_draw_backend_end:

```

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.

```

1108 \cs_new_protected:Npn \_draw_backend_begin:
1109 {
1110   \_kernel_backend_literal:n { ps::[begin] }
1111   \_draw_backend_literal:n { @beginspecial }
1112 }
1113 \cs_new_protected:Npn \_draw_backend_end:
1114 {
1115   \_draw_backend_literal:n { @endspecial }
1116   \_kernel_backend_literal:n { ps::[end] }
1117 }

```

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

```

\_draw_backend_scope_begin:
\_draw_backend_scope_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.

```

1118 \cs_new_protected:Npn \_draw_backend_scope_begin:
1119 { \_draw_backend_literal:n { save } }
1120 \cs_new_protected:Npn \_draw_backend_scope_end:
1121 { \_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.

```

1122 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1123 {
1124   \_draw_backend_literal:x
1125   {
1126     \dim_to_decimal_in_bp:n {#1} ~
1127     \dim_to_decimal_in_bp:n {#2} ~ moveto
1128   }
1129 }
1130 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1131 {
1132   \_draw_backend_literal:x
1133   {
1134     \dim_to_decimal_in_bp:n {#1} ~
1135     \dim_to_decimal_in_bp:n {#2} ~ lineto
1136   }
1137 }
1138 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1139 {
1140   \_draw_backend_literal:x
1141   {
1142     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1143     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1144     moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1145   }
1146 }
1147 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1148 {
1149   \_draw_backend_literal:x
1150   {
1151     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1152     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1153     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1154     curveto
1155   }
1156 }

```

(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`

```

1157 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1158 { \bool_gset_true:N \g__draw_draw_eor_bool }
1159 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1160 { \bool_gset_false:N \g__draw_draw_eor_bool }
1161 \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 T_EX 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`

```

1162 \cs_new_protected:Npn \__draw_backend_closepath:
1163 { \__draw_backend_literal:n { closepath } }
1164 \cs_new_protected:Npn \__draw_backend_stroke:
1165 {
1166   \__draw_backend_literal:n { gsave }
1167   \__draw_backend_literal:n { color.sc }
1168   \__draw_backend_literal:n { stroke }
1169   \__draw_backend_literal:n { grestore }
1170   \bool_if:NT \g__draw_draw_clip_bool
1171   {
1172     \__draw_backend_literal:x
1173     {
1174       \bool_if:NT \g__draw_draw_eor_bool { eo }
1175       clip
1176     }
1177   }
1178   \__draw_backend_literal:n { newpath }
1179   \bool_gset_false:N \g__draw_draw_clip_bool
1180 }
1181 \cs_new_protected:Npn \__draw_backend_closestroke:
1182 {
1183   \__draw_backend_closepath:
1184   \__draw_backend_stroke:
1185 }
1186 \cs_new_protected:Npn \__draw_backend_fill:
1187 {
1188   \__draw_backend_literal:x
1189   {
1190     \bool_if:NT \g__draw_draw_eor_bool { eo }
1191     fill
1192   }
1193   \bool_if:NT \g__draw_draw_clip_bool
1194   {
1195     \__draw_backend_literal:x
1196     {
1197       \bool_if:NT \g__draw_draw_eor_bool { eo }
1198       clip
1199     }
1200   }
1201   \__draw_backend_literal:n { newpath }
1202   \bool_gset_false:N \g__draw_draw_clip_bool
1203 }
1204 \cs_new_protected:Npn \__draw_backend_fillstroke:
1205 {
1206   \__draw_backend_literal:x
1207   {
1208     \bool_if:NT \g__draw_draw_eor_bool { eo }

```



```

1209         fill
1210     }
1211     \__draw_backend_literal:n { gsave }
1212     \__draw_backend_literal:n { color.sc }
1213     \__draw_backend_literal:n { stroke }
1214     \__draw_backend_literal:n { grestore }
1215     \bool_if:NT \g__draw_draw_clip_bool
1216     {
1217         \__draw_backend_literal:x
1218         {
1219             \bool_if:NT \g__draw_draw_eor_bool { eo }
1220             clip
1221         }
1222     }
1223     \__draw_backend_literal:n { newpath }
1224     \bool_gset_false:N \g__draw_draw_clip_bool
1225 }
1226 \cs_new_protected:Npn \__draw_backend_clip:
1227 { \bool_gset_true:N \g__draw_draw_clip_bool }
1228 \bool_new:N \g__draw_draw_clip_bool
1229 \cs_new_protected:Npn \__draw_backend_discardpath:
1230 {
1231     \bool_if:NT \g__draw_draw_clip_bool
1232     {
1233         \__draw_backend_literal:x
1234         {
1235             \bool_if:NT \g__draw_draw_eor_bool { eo }
1236             clip
1237         }
1238     }
1239     \__draw_backend_literal:n { newpath }
1240     \bool_gset_false:N \g__draw_draw_clip_bool
1241 }

```

(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:
1242 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1243 {
1244     \__draw_backend_literal:x
1245     {
1246         [
1247             \exp_args:Nf \use:n
1248             { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1249         ] ~
1250         \dim_to_decimal_in_bp:n {#2} ~ setdash
1251     }
1252 }
1253 \cs_new:Npn \__draw_backend_dash:n #1
1254 { ~ \dim_to_decimal_in_bp:n {#1} }
1255 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1256 {
1257     \__draw_backend_literal:x
1258     { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }

```

```

1259 }
1260 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1261 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1262 \cs_new_protected:Npn \__draw_backend_cap_butt:
1263 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1264 \cs_new_protected:Npn \__draw_backend_cap_round:
1265 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1266 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1267 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1268 \cs_new_protected:Npn \__draw_backend_join_miter:
1269 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1270 \cs_new_protected:Npn \__draw_backend_join_round:
1271 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1272 \cs_new_protected:Npn \__draw_backend_join_bevel:
1273 { \__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.

```

1274 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1275 {
1276   \__draw_backend_literal:n
1277   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1278 }

```

(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).

```

1279 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1280 {
1281   \__draw_backend_literal:n { @endspecial }
1282   \__draw_backend_literal:n { [end] }
1283   \__draw_backend_literal:n { [begin] }
1284   \__draw_backend_literal:n { save }
1285   \__draw_backend_literal:n { currentpoint }
1286   \__draw_backend_literal:n { currentpoint~translate }
1287   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1288   \__draw_backend_cm:nnnn { #2 } { #3 } { #4 } { #5 }
1289   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1290   \__draw_backend_literal:n { neg~exch~neg~exch~translate }

```

```

1291     \__draw_backend_literal:n { [end] }
1292     \hbox_overlap_right:n { \box_use:N #1 }
1293     \__draw_backend_literal:n { [begin] }
1294     \__draw_backend_literal:n { restore }
1295     \__draw_backend_literal:n { [end] }
1296     \__draw_backend_literal:n { [begin] }
1297     \__draw_backend_literal:n { @beginspecial }
1298 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1299 </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.

```

1300 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

__draw_backend_literal:n Pass data through using a dedicated interface.

```

\__draw_backend_literal:x
1301 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1302 \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:
1303 \cs_new_protected:Npn \__draw_backend_begin:
1304 { \__draw_backend_scope_begin: }
1305 \cs_new_protected:Npn \__draw_backend_end:
1306 { \__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:
1307 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1308 \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
1309 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1310 {
1311     \__draw_backend_literal:x
1312     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1313 }
1314 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1315 {
1316     \__draw_backend_literal:x
1317     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1318 }
1319 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1320 {

```

```

1321 \__draw_backend_literal:x
1322 {
1323   \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1324   \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1325   \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1326   c
1327 }
1328 }
1329 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1330 {
1331   \__draw_backend_literal:x
1332   {
1333     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1334     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1335     re
1336   }
1337 }

```

(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
1338 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1339 { \bool_gset_true:N \g__draw_draw_eor_bool }
1340 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1341 { \bool_gset_false:N \g__draw_draw_eor_bool }
1342 \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.

```

\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
1343 \cs_new_protected:Npn \__draw_backend_closepath:
1344 { \__draw_backend_literal:n { h } }
1345 \cs_new_protected:Npn \__draw_backend_stroke:
1346 { \__draw_backend_literal:n { S } }
1347 \cs_new_protected:Npn \__draw_backend_closestroke:
1348 { \__draw_backend_literal:n { s } }
1349 \cs_new_protected:Npn \__draw_backend_fill:
1350 {
1351   \__draw_backend_literal:x
1352   { f \bool_if:NT \g__draw_draw_eor_bool * }
1353 }
1354 \cs_new_protected:Npn \__draw_backend_fillstroke:
1355 {
1356   \__draw_backend_literal:x
1357   { B \bool_if:NT \g__draw_draw_eor_bool * }
1358 }
1359 \cs_new_protected:Npn \__draw_backend_clip:
1360 {
1361   \__draw_backend_literal:x
1362   { W \bool_if:NT \g__draw_draw_eor_bool * }
1363 }
1364 \cs_new_protected:Npn \__draw_backend_discardpath:
1365 { \__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:
1366 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1367 {
1368   \__draw_backend_literal:x
1369   {
1370     [
1371       \exp_args:Nf \use:n
1372       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1373     ] ~
1374     \dim_to_decimal_in_bp:n {#2} ~ d
1375   }
1376 }
1377 \cs_new:Npn \__draw_backend_dash:n #1
1378 { ~ \dim_to_decimal_in_bp:n {#1} }
1379 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1380 {
1381   \__draw_backend_literal:x
1382   { \dim_to_decimal_in_bp:n {#1} ~ w }
1383 }
1384 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1385 { \__draw_backend_literal:x { #1 ~ M } }
1386 \cs_new_protected:Npn \__draw_backend_cap_but:
1387 { \__draw_backend_literal:n { 0 ~ J } }
1388 \cs_new_protected:Npn \__draw_backend_cap_round:
1389 { \__draw_backend_literal:n { 1 ~ J } }
1390 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1391 { \__draw_backend_literal:n { 2 ~ J } }
1392 \cs_new_protected:Npn \__draw_backend_join_miter:
1393 { \__draw_backend_literal:n { 0 ~ j } }
1394 \cs_new_protected:Npn \__draw_backend_join_round:
1395 { \__draw_backend_literal:n { 1 ~ j } }
1396 \cs_new_protected:Npn \__draw_backend_join_bevel:
1397 { \__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!

```

1398 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1399 {
1400   <*luatex | pdftex>
1401   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1402   </luatex | pdftex>
1403   <*dvipdfmx | xetex>
1404   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1405   \__draw_backend_cm_aux:nnnn
1406   </dvipdfmx | xetex>

```

```

1407 }
1408 <*dvipdfmx | xetex>
1409 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1410 {
1411   \__kernel_backend_literal:x
1412   {
1413     x:rotate~
1414     \fp_compare:nNnTF {#1} = \c_zero_fp
1415       { 0 }
1416       { \fp_eval:n { round ( -#1 , 5 ) } }
1417   }
1418   \__kernel_backend_literal:x
1419   {
1420     x:scale~
1421     \fp_eval:n { round ( #2 , 5 ) } ~
1422     \fp_eval:n { round ( #3 , 5 ) }
1423   }
1424   \__kernel_backend_literal:x
1425   {
1426     x:rotate~
1427     \fp_compare:nNnTF {#4} = \c_zero_fp
1428       { 0 }
1429       { \fp_eval:n { round ( -#4 , 5 ) } }
1430   }
1431 }
1432 </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.

```

1433 <*dvipdfmx|xetex>
1434 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1435 {
1436   \use:x
1437   {
1438     \__draw_backend_cm_decompose_auxi:nnnnN
1439     { \fp_eval:n { (#1 + #4) / 2 } }
1440     { \fp_eval:n { (#1 - #4) / 2 } }
1441     { \fp_eval:n { (#3 + #2) / 2 } }
1442     { \fp_eval:n { (#3 - #2) / 2 } }
1443   }
1444   #5
1445 }
1446 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1447 {
1448   \use:x
1449   {
1450     \__draw_backend_cm_decompose_auxii:nnnnN
1451     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1452     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1453     { \fp_eval:n { atand ( #3 , #2 ) } }
1454     { \fp_eval:n { atand ( #4 , #1 ) } }
1455   }
1456   #5
1457 }
1458 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1459 {
1460   \use:x
1461   {
1462     \__draw_backend_cm_decompose_auxiii:nnnnN
1463     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1464     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1465     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1466     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1467   }
1468   #5
1469 }
1470 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1471 {
1472   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1473   { #5 {#1} {#2} {#3} {#4} }
1474   { #5 {#1} {#3} {#2} {#4} }
1475 }
1476 </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.

```

1477 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1478 {
1479   \__kernel_backend_scope_begin:
1480   <*luatex | pdftex>
1481   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1482   </luatex | pdftex>
1483   <*dviptfm | xetex>
1484   \__kernel_backend_literal:n
1485   { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1486   </dviptfm | xetex>
1487   \hbox_overlap_right:n { \box_use:N #1 }
1488   <*dviptfm | xetex>
1489   \__kernel_backend_literal:n { pdf:etrans }
1490   </dviptfm | xetex>
1491   \__kernel_backend_scope_end:
1492 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1493 </dviptfm | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1494 <*dvisvgm>

```

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

```

\__draw_backend_literal:x
1495 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1496 \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

```

1497 \cs_new_protected:Npn \__draw_backend_begin:
1498 {
1499   \__kernel_backend_scope_begin:
1500   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1501 }
1502 \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. __draw_backend_lineto:nn Since paths should be fully expanded there is no need to worry about the internal x-type expansion.

```

\__draw_backend_rectangle:nnnn
\__draw_backend_curveto:nnnnnn
\__draw_backend_add_to_path:n
\g__draw_draw_path_tl
1503 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1504 {
1505   \__draw_backend_add_to_path:n
1506   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1507 }
1508 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1509 {
1510   \__draw_backend_add_to_path:n

```



```

1511     { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1512   }
1513   \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1514   {
1515     \__draw_backend_add_to_path:n
1516     {
1517       M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1518       h ~ \dim_to_decimal:n {#3} ~
1519       v ~ \dim_to_decimal:n {#4} ~
1520       h ~ \dim_to_decimal:n { -#3 } ~
1521       Z
1522     }
1523   }
1524   \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1525   {
1526     \__draw_backend_add_to_path:n
1527     {
1528       C ~
1529       \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1530       \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1531       \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1532     }
1533   }
1534   \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1535   {
1536     \tl_gset:Nx \g__draw_draw_path_tl
1537     {
1538       \g__draw_draw_path_tl
1539       \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1540       #1
1541     }
1542   }
1543   \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: 1544 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1545   { \__draw_backend_scope:n { fill-rule="evenodd" } }
1546 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1547   { \__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: 1548 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_clip: 1549 { \__draw_backend_add_to_path:n { Z } }
\__draw_backend_discardpath: 1550 \cs_new_protected:Npn \__draw_backend_path:n #1
\g__draw_draw_clip_bool 1551 {
\g__draw_draw_path_int 1552   \bool_if:NTF \g__draw_draw_clip_bool

```

```

1553 {
1554   \int_gincr:N \g__draw_clip_path_int
1555   \__draw_backend_literal:x
1556   {
1557     < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1558     { ?nl }
1559     <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1560     < /clipPath > { ? nl }
1561     <
1562       use~xlink:href =
1563       "\c_hash_str l3path \int_use:N \g__draw_path_int " ~
1564       #1
1565     />
1566   }
1567   \__draw_backend_scope:x
1568   {
1569     clip-path =
1570     "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1571   }
1572 }
1573 {
1574   \__draw_backend_literal:x
1575   { <path ~ d=" \g__draw_draw_path_tl " ~ #1 /> }
1576 }
1577 \tl_gclear:N \g__draw_draw_path_tl
1578 \bool_gset_false:N \g__draw_draw_clip_bool
1579 }
1580 \int_new:N \g__draw_path_int
1581 \cs_new_protected:Npn \__draw_backend_stroke:
1582 { \__draw_backend_path:n { style="fill:none" } }
1583 \cs_new_protected:Npn \__draw_backend_closestroke:
1584 {
1585   \__draw_backend_closepath:
1586   \__draw_backend_stroke:
1587 }
1588 \cs_new_protected:Npn \__draw_backend_fill:
1589 { \__draw_backend_path:n { style="stroke:none" } }
1590 \cs_new_protected:Npn \__draw_backend_fillstroke:
1591 { \__draw_backend_path:n { } }
1592 \cs_new_protected:Npn \__draw_backend_clip:
1593 { \bool_gset_true:N \g__draw_draw_clip_bool }
1594 \bool_new:N \g__draw_draw_clip_bool
1595 \cs_new_protected:Npn \__draw_backend_discardpath:
1596 {
1597   \bool_if:NT \g__draw_draw_clip_bool
1598   {
1599     \int_gincr:N \g__draw_clip_path_int
1600     \__draw_backend_literal:x
1601     {
1602       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1603       { ?nl }
1604       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1605       < /clipPath >
1606     }

```

```

1607         \__draw_backend_scope:x
1608         {
1609             clip-path =
1610             "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1611         }
1612     }
1613     \tl_gclear:N \g__draw_draw_path_tl
1614     \bool_gset_false:N \g__draw_draw_clip_bool
1615 }

```

(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_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1616 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1617 {
1618     \use:x
1619     {
1620         \__draw_backend_dash_aux:nn
1621         { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1622         { \dim_to_decimal:n {#2} }
1623     }
1624 }
1625 \cs_new:Npn \__draw_backend_dash:n #1
1626 { , \dim_to_decimal_in_bp:n {#1} }
1627 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1628 {
1629     \__draw_backend_scope:x
1630     {
1631         stroke-dasharray =
1632         "
1633         \tl_if_empty:oTF { \use_none:n #1 }
1634         { none }
1635         { \use_none:n #1 }
1636         " ~
1637         stroke-offset=" #2 "
1638     }
1639 }
1640 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1641 { \__draw_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1642 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1643 { \__draw_backend_scope:x { stroke-miterlimit=" #1 " } }
1644 \cs_new_protected:Npn \__draw_backend_cap_butt:
1645 { \__draw_backend_scope:n { stroke-linecap="butt" } }
1646 \cs_new_protected:Npn \__draw_backend_cap_round:
1647 { \__draw_backend_scope:n { stroke-linecap="round" } }
1648 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1649 { \__draw_backend_scope:n { stroke-linecap="square" } }
1650 \cs_new_protected:Npn \__draw_backend_join_miter:
1651 { \__draw_backend_scope:n { stroke-linejoin="miter" } }
1652 \cs_new_protected:Npn \__draw_backend_join_round:
1653 { \__draw_backend_scope:n { stroke-linejoin="round" } }
1654 \cs_new_protected:Npn \__draw_backend_join_bevel:
1655 { \__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.

```

1656 \cs_new_protected:Npn \_draw_backend_cm:nnnn #1#2#3#4
1657 {
1658   \_draw_backend_scope:n
1659   {
1660     transform =
1661     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1662   }
1663 }

```

(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.

```

1664 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1665 {
1666   \_kernel_backend_scope_begin:
1667   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1668   \_kernel_backend_literal_svg:n
1669   {
1670     < g~
1671     stroke="none"~
1672     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1673     >
1674   }
1675   \box_set_wd:Nn #1 { Opt }
1676   \box_set_ht:Nn #1 { Opt }
1677   \box_set_dp:Nn #1 { Opt }
1678   \box_use:N #1
1679   \_kernel_backend_literal_svg:n { </g> }
1680   \_kernel_backend_scope_end:
1681 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```

1682 </dvisvgm>
1683 </package>

```

5 l3backend-graphics Implementation

```

1684 <*package>
1685 <@=graphics>

```

5.1 dvips backend

```

1686 <*dvips>

```

`_graphics_backend_getbb_eps:n` Simply use the generic function.

```

1687 \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.

```

1688 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1689 {
1690   \__kernel_backend_literal:x
1691   {
1692     PSfile = #1 \c_space_tl
1693     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1694     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1695     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1696     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1697   }
1698 }

```

(End definition for `_graphics_backend_include_eps:n`.)

```

1699 </dvips>

```

5.2 LuaTeX and pdfTeX backends

```

1700 <*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.

```

1701 \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.

```

1702 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1703 {
1704   \int_zero:N \l_graphics_page_int
1705   \tl_clear:N \l_graphics_pagebox_tl
1706   \tl_set:Nx \l_graphics_graphics_attr_tl
1707   {
1708     \tl_if_empty:NF \l_graphics_decodearray_tl
1709     { :D \l_graphics_decodearray_tl }
1710     \bool_if:NT \l_graphics_interpolate_bool
1711     { :I }
1712   }
1713   \tl_clear:N \l_graphics_graphics_attr_tl
1714   \_graphics_backend_getbb_auxi:n {#1}
1715 }
1716 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1717 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1718 {
1719   \tl_clear:N \l_graphics_decodearray_tl
1720   \bool_set_false:N \l_graphics_interpolate_bool

```

```

1721 \tl_set:Nx \l__graphics_graphics_attr_tl
1722 {
1723   : \l_graphics_pagebox_tl
1724   \int_compare:nNnT \l_graphics_page_int > 1
1725   { :P \int_use:N \l_graphics_page_int }
1726 }
1727 \__graphics_backend_getbb_auxi:n {#1}
1728 }
1729 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1730 {
1731   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1732   { \__graphics_backend_getbb_auxii:n {#1} }
1733 }

```

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.

```

1734 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1735 {
1736   \tex_immediate:D \tex_pdfximage:D
1737   \bool_lazy_or:nnT
1738   { \l_graphics_interpolate_bool }
1739   { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1740   {
1741     attr ~
1742     {
1743       \tl_if_empty:NF \l_graphics_decodearray_tl
1744       { /Decode~[ \l_graphics_decodearray_tl ] }
1745       \bool_if:NT \l_graphics_interpolate_bool
1746       { /Interpolate~true }
1747     }
1748   }
1749   \int_compare:nNnT \l_graphics_page_int > 0
1750   { page ~ \int_use:N \l_graphics_page_int }
1751   \tl_if_empty:NF \l_graphics_pagebox_tl
1752   { \l_graphics_pagebox_tl }
1753   {#1}
1754   \hbox_set:Nn \l__graphics_internal_box
1755   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1756   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1757   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1758   \int_const:cn { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1759   { \tex_the:D \tex_pdflastximage:D }
1760   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1761 }

```

(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.

```

1762 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1763 {
1764   \tex_pdfrefximage:D

```

```

1765     \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1766   }
1767   \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1768   \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 LuaTeX/pdfTeX 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

1769 \sys_if_shell:T
1770 {
1771   \str_new:N \l__graphics_backend_dir_str
1772   \str_new:N \l__graphics_backend_name_str
1773   \str_new:N \l__graphics_backend_ext_str
1774   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1775   {
1776     \file_parse_full_name:nNNN {#1}
1777     \l__graphics_backend_dir_str
1778     \l__graphics_backend_name_str
1779     \l__graphics_backend_ext_str
1780     \exp_args:Nx \__graphics_backend_getbb_eps:nn
1781     {
1782       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1783       -converted-to.pdf
1784     }
1785     {#1}
1786   }
1787   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1788   {
1789     \file_compare_timestamp:nNnT {#2} > {#1}
1790     {
1791       \sys_shell_now:n
1792       { repstopdf ~ #2 ~ #1 }
1793     }
1794     \tl_set:Nn \l__graphics_name_tl {#1}
1795     \__graphics_backend_getbb_pdf:n {#1}
1796   }
1797   \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1798   {
1799     \file_parse_full_name:nNNN {#1}
1800     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1801     \exp_args:Nx \__graphics_backend_include_pdf:n
1802     {
1803       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1804       -converted-to.pdf
1805     }
1806   }
1807 }

```

(End definition for __graphics_backend_getbb_eps:n and others.)

```

1808 </luatex | pdftex>

```

5.3 dvipdfmx backend

1809 $\langle *dvipdfmx | xetex \rangle$

_graphics_backend_getbb_eps:n
_graphics_backend_getbb_jpg:n
_graphics_backend_getbb_pdf:n
_graphics_backend_getbb_png:n

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

```
1810 \cs_new_eq:NN \_graphics_backend_getbb_eps:n \graphics_read_bb:n
1811  $\langle *dvipdfmx \rangle$ 
1812 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1813 {
1814   \int_zero:N \l_graphics_page_int
1815   \tl_clear:N \l_graphics_pagebox_tl
1816   \graphics_extract_bb:n {#1}
1817 }
1818 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1819 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1820 {
1821   \tl_clear:N \l_graphics_decodearray_tl
1822   \bool_set_false:N \l_graphics_interpolate_bool
1823   \graphics_extract_bb:n {#1}
1824 }
1825  $\langle /dvipdfmx \rangle$ 
```

(End definition for _graphics_backend_getbb_eps:n and others.)

\g__graphics_track_int

Used to track the object number associated with each graphic.

```
1826 \int_new:N \g__graphics_track_int
```

(End definition for \g__graphics_track_int.)

_graphics_backend_include_eps:n
_graphics_backend_include_jpg:n
_graphics_backend_include_pdf:n
_graphics_backend_include_png:n
_graphics_backend_include_auxi:nn
_graphics_backend_include_auxii:nnn
_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 XeTeX: for the latter it is better to use the primitive route. The relevant code for that is included later in this file.

```
1827 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1828 {
1829   \__kernel_backend_literal:x
1830   {
1831     PSfile = #1 \c_space_tl
1832     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1833     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1834     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1835     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1836   }
1837 }
1838 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1839 { \_graphics_backend_include_auxi:nn {#1} { image } }
1840 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1841  $\langle *dvipdfmx \rangle$ 
1842 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1843 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1844  $\langle /dvipdfmx \rangle$ 
```

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.


```

1845 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1846 {
1847   \__graphics_backend_include_auxii:xnn
1848   {
1849     \tl_if_empty:NF \l_graphics_pagebox_tl
1850     { : \l_graphics_pagebox_tl }
1851     \int_compare:nNnT \l_graphics_page_int > 1
1852     { :P \int_use:N \l_graphics_page_int }
1853     \tl_if_empty:NF \l_graphics_decodearray_tl
1854     { :D \l_graphics_decodearray_tl }
1855     \bool_if:NT \l_graphics_interpolate_bool
1856     { :I }
1857   }
1858   {#1} {#2}
1859 }
1860 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1861 {
1862   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1863   {
1864     \__kernel_backend_literal:x
1865     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1866   }
1867   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1868 }
1869 \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!

```

1870 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1871 {
1872   \int_gincr:N \g__graphics_track_int
1873   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1874   \__kernel_backend_literal:x
1875   {
1876     pdf:#3~
1877     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1878     \int_compare:nNnT \l_graphics_page_int > 1
1879     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1880     \tl_if_empty:NF \l_graphics_pagebox_tl
1881     {
1882       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1883       bbox ~
1884       \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1885       \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1886       \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1887       \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1888     }
1889     (#1)
1890     \bool_lazy_or:nnT
1891     { \l_graphics_interpolate_bool }
1892     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1893     {
1894       <<

```

```

1895         \tl_if_empty:NF \l_graphics_decodearray_tl
1896         { /Decode~[ \l_graphics_decodearray_tl ] }
1897         \bool_if:NT \l_graphics_interpolate_bool
1898         { /Interpolate~true> }
1899     >>
1900 }
1901 }
1902 }

```

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

```

1903 </dvipdfmx | xetex>

```

5.4 X_YTeX backend

```

1904 <*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

```

```

1905 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1906 {
1907     \int_zero:N \l_graphics_page_int
1908     \tl_clear:N \l_graphics_pagebox_tl
1909     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1910 }
1911 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1912 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1913 {
1914     \tl_clear:N \l_graphics_decodearray_tl
1915     \bool_set_false:N \l_graphics_interpolate_bool
1916     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1917 }
1918 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
1919 {
1920     \int_compare:nNnTF \l_graphics_page_int > 1
1921     { \__graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
1922     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1923 }
1924 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nN #1#2#3
1925 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1926 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nN { V }
1927 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1928 {
1929     \tl_if_empty:NTF \l_graphics_pagebox_tl
1930     { \__graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1931     { \__graphics_backend_getbb_auxv:nNnn }
1932     {#1} #2 {#3} {#4}
1933 }
1934 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nNnn #1#2#3#4#5
1935 {
1936     \use:x

```

```

1937     {
1938         \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1939         { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1940     }
1941 }
1942 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1943 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1944 {
1945     \graphics_bb_restore:nF {#1#3}
1946     { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1947 }
1948 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1949 {
1950     \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1951     \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1952     \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1953     \graphics_bb_save:n {#1#3}
1954 }
1955 \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_XeTeXpdfxfile: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.

```

1956 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1957 {
1958     \tex_XeTeXpdfxfile:D
1959     \__graphics_backend_include_pdf_quote:w #1 "#1" \s__graphics_stop \c_space_tl
1960     \int_compare:nNnT \l_graphics_page_int > 0
1961         { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1962     \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1963 }
1964 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \s__graphics_stop
1965 { " #2 " }

```

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

1966 </xetex>

5.5 dvisvgm backend

1967 <*dvisvgm>

__graphics_backend_getbb_eps:n Simply use the generic function.

```

1968 \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 These can be included by extracting the bounding box data.

```

\__graphics_backend_getbb_jpg:n
1969 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1970 {
1971     \int_zero:N \l_graphics_page_int

```

```

1972 \tl_clear:N \l_graphics_pagebox_tl
1973 \graphics_extract_bb:n {#1}
1974 }
1975 \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

```

1976 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1977 {
1978 \tl_clear:N \l_graphics_decodearray_tl
1979 \bool_set_false:N \l_graphics_interpolate_bool
1980 \graphics_extract_bb:n {#1}
1981 }
(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
1982 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1983 { __graphics_backend_include:nn { PSfile } {#1} }
1984 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1985 { __graphics_backend_include:nn { pdffile } {#1} }
1986 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
1987 {
1988 \__kernel_backend_literal:x
1989 {
1990 #1 = #2 \c_space_tl
1991 llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1992 lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1993 urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1994 ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1995 }
1996 }
(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

```

1997 \cs_new_protected:Npn \__graphics_backend_include_png:n #1
1998 {
1999 \__kernel_backend_literal:x
2000 {
2001 dvisvgm:img~
2002 \dim_to_decimal:n { \l_graphics_ury_dim } ~
2003 \dim_to_decimal:n { \l_graphics_ury_dim } ~
2004 \__graphics_backend_include_bitmap_quote:w #1 " #1 " \s__graphics_stop
2005 }
2006 }
2007 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_png:n
2008 \cs_new:Npn \__graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \s__graphics_stop
2009 { " #2 " }

```

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

2010 `\</dvisvgm>`

2011 `\</package>`

6 l3backend-pdf Implementation

2012 `*package>`

2013 `\@@=pdf>`

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

6.1 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`

2014 `\box_new:N \l__pdf_internal_box`

(End definition for `\l__pdf_internal_box`.)

6.2 dvips backend

2015 `*dvips>`

`__pdf_backend_pdfmark:n`

`__pdf_backend_pdfmark:x`

Used often enough it should be a separate function.

2016 `\cs_new_protected:Npn __pdf_backend_pdfmark:n #1`

2017 `{ __kernel_backend_postscript:n { mark #1 ~ pdfmark } }`

2018 `\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`

2019 `\cs_new_protected:Npn __pdf_backend_catalog_gput:nn #1#2`

2020 `{ __pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }`

2021 `\cs_new_protected:Npn __pdf_backend_info_gput:nn #1#2`

2022 `{ __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.

```
2023 \int_new:N \g__pdf_backend_object_int
2024 \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.

```
2025 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2026 {
2027   \int_gincr:N \g__pdf_backend_object_int
2028   \int_const:cn
2029     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2030     { \g__pdf_backend_object_int }
2031   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2032 }
2033 \cs_new:Npn \__pdf_backend_object_ref:n #1
2034 { { 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.

```
2035 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2036 {
2037   \__pdf_backend_pdfmark:x
2038   {
2039     /_objdef ~ \__pdf_backend_object_ref:n {#1}
2040     /type
2041     \str_case:e:nn
2042       { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2043       {
2044         { array } { /array }
2045         { dict } { /dict }
2046         { fstream } { /stream }
2047         { stream } { /stream }
2048       }
2049     /OBJ
2050   }
2051   \use:c
2052     { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2053     { \__pdf_backend_object_ref:n {#1} } {#2}
2054 }
2055 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2056 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2057 {
2058   \__pdf_backend_pdfmark:x
2059   { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2060 }
2061 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2062 {
2063   \__pdf_backend_pdfmark:x
2064   { #1 << \exp_not:n {#2} >> /PUT }
2065 }
2066 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
```

```

2067 {
2068   \exp_args:Nx
2069   \__pdf_backend_object_write_fstream:nnn {#1} #2
2070 }
2071 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2072 {
2073   \__kernel_backend_postscript:n
2074   {
2075     SDict ~ begin ~
2076     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2077     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2078     end
2079   }
2080 }
2081 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2082 {
2083   \exp_args:Nx
2084   \__pdf_backend_object_write_stream:nnn {#1} #2
2085 }
2086 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2087 {
2088   \__kernel_backend_postscript:n
2089   {
2090     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2091     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2092   }
2093 }

```

(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.

```

2094 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2095 {
2096   \int_gincr:N \g__pdf_backend_object_int
2097   \__pdf_backend_pdfmark:x
2098   {
2099     /_objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2100     /type
2101     \str_case:nn
2102       {#1}
2103       {
2104         { array } { /array }
2105         { dict } { /dict }
2106         { fstream } { /stream }
2107         { stream } { /stream }
2108       }
2109     /OBJ
2110   }
2111   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2112   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2113 }
2114 \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.

```

2115 \cs_new:Npn \__pdf_backend_object_last:
2116 { { 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.

```

2117 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2118 { { 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.

```

2119 \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.

```

2120 \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.

```

2121 \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.

```

2122 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2123 {
2124   \exp_args:Nf \__pdf_backend_annotation_aux:nnnn
2125     { \dim_eval:n {#1} } {#2} {#3} {#4}
2126 }
2127 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2128 {
2129   \box_move_down:nn {#3}
2130   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2131   \box_move_up:nn {#2}
2132   {
2133     \hbox:n
2134     {
2135       \__kernel_kern:n {#1}
2136       \__kernel_backend_postscript:n { pdf.save.ur }
2137       \__kernel_kern:n { -#1 }
2138     }

```



```

2139     }
2140     \int_gincr:N \g__pdf_backend_object_int
2141     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2142     \__pdf_backend_pdfmark:x
2143     {
2144         /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2145         pdf.rect
2146         #4 ~
2147         /ANN
2148     }
2149 }

```

(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.

```

2150 \cs_new:Npn \__pdf_backend_annotation_last:
2151 { { 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.

```

2152 \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.

```

2153 \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.

```

2154 \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.

```

2155 \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.

```

2156 \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.

```

2157 \tl_new:N \l__pdf_breaklink_pdfmark_tl
2158 \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.

```

2159 \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.

2160 `\cs_new_eq:NN _pdf_breaklink_usebox:N \box_use:N`

(End definition for `_pdf_breaklink_usebox:N`.)

`_pdf_backend_link_begin_goto:nnw`

`_pdf_backend_link_begin_user:nnw`

`_pdf_backend_link:nw`

`_pdf_backend_link_aux:nw`

`_pdf_backend_link_end:`

`_pdf_backend_link_end_aux:`

`_pdf_backend_link_minima:`

`_pdf_backend_link_outerbox:n`

`_pdf_backend_link_sf_save:`

`_pdf_backend_link_sf_restore:`

`pdf.linkdp.pad`

`pdf.linkht.pad`

`pdf.llx`

`pdf.lly`

`pdf.ury`

`pdf.link.dict`

`pdf.outerbox`

`pdf.baselineskip`

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.

```

2161 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2162 { \_pdf_backend_link_begin:nw { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2163 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2164 { \_pdf_backend_link_begin:nw {#1#2} }
2165 \cs_new_protected:Npn \_pdf_backend_link_begin:nw #1
2166 {
2167   \bool_if:NF \g__pdf_backend_link_bool
2168   { \_pdf_backend_link_begin_aux:nw {#1} }
2169 }
2170 \cs_new_protected:Npn \_pdf_backend_link_begin_aux:nw #1
2171 {
2172   \bool_gset_true:N \g__pdf_backend_link_bool
2173   \_kernel_backend_postscript:n
2174   { /pdf.link.dict ( #1 ) def }
2175   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2176   \_pdf_backend_link_sf_save:
2177   \mode_if_math:TF
2178   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2179   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2180   \hbox_set:Nw \l__pdf_backend_content_box
2181   \_pdf_backend_link_sf_restore:
2182   \bool_if:NT \g__pdf_backend_link_math_bool
2183   { \c_math_toggle_token }
2184 }
2185 \cs_new_protected:Npn \_pdf_backend_link_end:
2186 {
2187   \bool_if:NT \g__pdf_backend_link_bool
2188   { \_pdf_backend_link_end_aux: }
2189 }
2190 \cs_new_protected:Npn \_pdf_backend_link_end_aux:
2191 {
2192   \bool_if:NT \g__pdf_backend_link_math_bool
2193   { \c_math_toggle_token }
2194   \_pdf_backend_link_sf_save:

```

```

2195 \hbox_set_end:
2196 \__pdf_backend_link_minima:
2197 \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2198 \exp_args:Nx \__pdf_backend_link_outerbox:n
2199 {
2200   \int_if_odd:nTF { \value { page } }
2201     { \oddsidemargin }
2202     { \evensidemargin }
2203 }
2204 \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2205 { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2206 \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2207 \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2208 \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2209 \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2210 {
2211   \hbox:n
2212     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2213 }
2214 \int_gincr:N \g__pdf_backend_object_int
2215 \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2216 \__kernel_backend_postscript:x
2217 {
2218   mark
2219   /objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2220   \g__pdf_backend_link_dict_tl \c_space_tl
2221   pdf.rect
2222   /ANN ~ \l__pdf_breaklink_pdfmark_tl
2223 }
2224 \__pdf_backend_link_sf_restore:
2225 \bool_gset_false:N \g__pdf_backend_link_bool
2226 }
2227 \cs_new_protected:Npn \__pdf_backend_link_minima:
2228 {
2229   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2230   \__kernel_backend_postscript:x
2231   {
2232     /pdf.linkdp.pad ~
2233     \dim_to_decimal:n
2234     {
2235       \dim_max:nn
2236       {
2237         \box_dp:N \l__pdf_backend_model_box
2238         - \box_dp:N \l__pdf_backend_content_box
2239       }
2240       { Opt }
2241     } ~
2242     pdf.pt.dvi ~ def
2243     /pdf.linkht.pad ~
2244     \dim_to_decimal:n
2245     {
2246       \dim_max:nn
2247       {
2248         \box_ht:N \l__pdf_backend_model_box

```

```

2249         - \box_ht:N \l__pdf_backend_content_box
2250     }
2251     { Opt }
2252 } ~
2253 pdf.pt.dvi ~ def
2254 }
2255 }
2256 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2257 {
2258     \__kernel_backend_postscript:x
2259     {
2260         /pdf.outerbox
2261         [
2262             \dim_to_decimal:n {#1} ~
2263             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2264             \dim_to_decimal:n { #1 + \textwidth } ~
2265             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2266         ]
2267         [ exch { pdf.pt.dvi } forall ] def
2268         /pdf.baselineskip ~
2269         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2270         { pdf.pt.dvi ~ def }
2271         { pop ~ pop }
2272     ifelse
2273 }
2274 }
2275 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2276 {
2277     \int_gset:Nn \g__pdf_backend_link_sf_int
2278     {
2279         \mode_if_horizontal:TF
2280         { \tex_spacefactor:D }
2281         { 0 }
2282     }
2283 }
2284 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2285 {
2286     \mode_if_horizontal:T
2287     {
2288         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2289         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2290     }
2291 }

```

(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.

```

2292 \use_none:n
2293 {
2294     \cs_if_exist:NT \@makecol@hook
2295     {

```

```

2296 \tl_put_right:Nn \@makecol@hook
2297 {
2298   \box_if_empty:NF \@cclv
2299   {
2300     \vbox_set:Nn \@cclv
2301     {
2302       \__kernel_backend_postscript:n
2303       {
2304         pdf.globaldict /pdf.brokenlink.rect ~ known
2305         { pdf.bordertracking.continue }
2306         if
2307         }
2308         \vbox_unpack_drop:N \@cclv
2309         \__kernel_backend_postscript:n
2310         { pdf.bordertracking.endpage }
2311       }
2312     }
2313   }
2314   \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2315   \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2316   \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2317 }
2318 }

```

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

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

```

2319 \cs_new:Npn \__pdf_backend_link_last:
2320 { { 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.

```

2321 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2322 {
2323   \__kernel_backend_postscript:x
2324   {
2325     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2326   }
2327 }

```

(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.

```

2328 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2329 {
2330   \__kernel_backend_postscript:n { pdf.dest.anchor }
2331   \__pdf_backend_pdfmark:x
2332   {
2333     /View
2334     [

```

```

2335     \str_case:nnF {#2}
2336     {
2337         { xyz }    { /XYZ ~ pdf.dest.point ~ null }
2338         { fit }    { /Fit }
2339         { fitb }   { /FitB }
2340         { fitbh }  { /FitBH ~ pdf.dest.y }
2341         { fitbv }  { /FitBV ~ pdf.dest.x }
2342         { fith }   { /FitH ~ pdf.dest.y }
2343         { fitv }   { /FitV ~ pdf.dest.x }
2344         { fitr }   { /Fit }
2345     }
2346     {
2347         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2348     }
2349 ]
2350 /Dest ( \exp_not:n {#1} ) cvn
2351 /DEST
2352 }
2353 }
2354 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2355 {
2356     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2357     { \dim_eval:n {#2} } {#1} {#3} {#4}
2358 }
2359 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2360 {
2361     \vbox_to_zero:n
2362     {
2363         \__kernel_kern:n {#4}
2364         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2365         \tex_vss:D
2366     }
2367     \__kernel_kern:n {#1}
2368     \vbox_to_zero:n
2369     {
2370         \__kernel_kern:n { -#3 }
2371         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2372         \tex_vss:D
2373     }
2374     \__kernel_kern:n { -#1 }
2375     \__pdf_backend_pdfmark:n
2376     {
2377         /View
2378         [
2379             /FitR ~
2380             pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2381             pdf.urx ~ pdf.ury ~ pdf.dest2device
2382         ]
2383         /Dest ( #2 ) cvn
2384         /DEST
2385     }
2386 }

```

(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.

```

2387 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2388 {
2389   \int_compare:nNnT {#1} = 0
2390   {
2391     \__kernel_backend_literal_postscript:n
2392     {
2393       /setdistillerparams ~ where
2394       { pop << /CompressPages ~ false >> setdistillerparams }
2395       if
2396     }
2397   }
2398 }
2399 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2400 {
2401   \bool_if:nF {#1}
2402   {
2403     \__kernel_backend_literal_postscript:n
2404     {
2405       /setdistillerparams ~ where
2406       { pop << /CompressStreams ~ false >> setdistillerparams }
2407       if
2408     }
2409   }
2410 }
```

(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!

```

2411 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
2412 \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!

```

2413 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2414 \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.

```

2415 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2416 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2417 \cs_new_protected:Npn \_pdf_backend_emc:
2418 { \_pdf_backend_pdfmark:n { /EMC } }
```

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

```

2419 </dvips>
```

6.3 LuaTeX and pdfTeX backend

2420 `<*luatex | pdftex>`

6.3.1 Annotations

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

```

2421 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2422 {
2423   <*luatex>
2424     \tex_pdfextension:D annot ~
2425   </luatex>
2426   <*pdftex>
2427     \tex_pdfannot:D
2428   </pdftex>
2429     width ~ \dim_eval:n {#1} ~
2430     height ~ \dim_eval:n {#2} ~
2431     depth ~ \dim_eval:n {#3} ~
2432     {#4}
2433 }
```

(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.

```

2434 \cs_new:Npx \_pdf_backend_annotation_last:
2435 {
2436   \exp_not:N \int_value:w
2437   <*luatex>
2438     \exp_not:N \tex_pdffeedback:D lastannot ~
2439   </luatex>
2440   <*pdftex>
2441     \exp_not:N \tex_pdflastannot:D
2442   </pdftex>
2443   \c_space_tl 0 ~ R
2444 }
```

(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 2445 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
\_pdf_backend_link_begin:nnnw 2446 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
\_pdf_backend_link_end: 2447 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2448 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2449 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2450 {
2451   <*luatex>
2452     \tex_pdfextension:D startlink ~
2453   </luatex>
2454   <*pdftex>
2455     \tex_pdfstartlink:D
2456   </pdftex>
2457     attr {#1}
2458     #2 {#3}
```



```

2459 }
2460 \cs_new_protected:Npn \__pdf_backend_link_end:
2461 {
2462   \*luatex
2463     \tex_pdfextension:D endlink \scan_stop:
2464   \*luatex
2465   \*pdftex
2466     \tex_pdfendlink:D
2467   \*pdftex
2468 }

```

(End definition for __pdf_backend_link_begin_goto:nnw and others.)

__pdf_backend_link_last: Formatted for direct use.

```

2469 \cs_new:Npx \__pdf_backend_link_last:
2470 {
2471   \exp_not:N \int_value:w
2472   \*luatex
2473     \exp_not:N \tex_pdffeedback:D lastlink ~
2474   \*luatex
2475   \*pdftex
2476     \exp_not:N \tex_pdflastlink:D
2477   \*pdftex
2478     \c_space_tl 0 ~ R
2479 }

```

(End definition for __pdf_backend_link_last:.)

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

```

2480 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2481 {
2482   \*luatex
2483     \tex_pdfvariable:D linkmargin
2484   \*luatex
2485   \*pdftex
2486     \tex_pdflinkmargin:D
2487   \*pdftex
2488     \dim_eval:n {#1} \scan_stop:
2489 }

```

(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

```

2490 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2491 {
2492   \*luatex
2493     \tex_pdfextension:D dest ~
2494   \*luatex
2495   \*pdftex
2496     \tex_pdfdest:D
2497   \*pdftex
2498     name {#1}
2499     \str_case:nnF {#2}

```

```

2500         {
2501             { xyz } { xyz }
2502             { fit } { fit }
2503             { fitb } { fitb }
2504             { fitbh } { fitbh }
2505             { fitbv } { fitbv }
2506             { fith } { fith }
2507             { fitv } { fitv }
2508             { fitr } { fitr }
2509         }
2510         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2511         \scan_stop:
2512     }
2513     \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2514     {
2515         \*luatex
2516         \tex_pdfextension:D dest ~
2517         \*pdftex
2518         \tex_pdfdest:D
2519         \*pdftex
2520         name {#1}
2521         fitr ~
2522         width \dim_eval:n {#2} ~
2523         height \dim_eval:n {#3} ~
2524         depth \dim_eval:n {#4} \scan_stop:
2525     }
2526 }

```

(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
2527 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2528 {
2529     \*luatex
2530     \tex_pdfextension:D catalog
2531     \*pdftex
2532     \tex_pdfcatalog:D
2533     \*pdftex
2534     { / #1 ~ #2 }
2535 }
2536
2537 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2538 {
2539     \*luatex
2540     \tex_pdfextension:D info
2541     \*pdftex
2542     \tex_pdfinfo:D
2543     \*pdftex
2544     { / #1 ~ #2 }
2545 }
2546 }

```

(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.

```
2547 \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
2548 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2549 {
2550   \*luatex
2551     \tex_pdfextension:D obj ~
2552   \*pdfTeX
2553     \tex_pdfobj:D
2554   \*pdfTeX
2555     \tex_pdfobj:D
2556     reserveobjnum ~
2557     \int_const:cn
2558     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2559   \*luatex
2560     { \tex_pdffeedback:D lastobj }
2561   \*pdfTeX
2562     { \tex_pdflastobj:D }
2563   \*pdfTeX
2564     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2565   }
2566   \cs_new:Npn \__pdf_backend_object_ref:n #1
2567     { \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
2569 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2570 {
2571   \*luatex
2572     \tex_immediate:D \tex_pdfextension:D obj ~
2573   \*pdfTeX
2574     \tex_immediate:D \tex_pdfobj:D
2575   \*pdfTeX
2576     \tex_immediate:D \tex_pdfobj:D
2577     useobjnum ~
2578     \int_use:c
2579     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2580   \str_case_e:nn
2581     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2582     {
2583       { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2584       { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2585       { fstream }
2586       {
2587         stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2588         file ~ { \__pdf_exp_not_ii:nn #2 }
2589       }
2590     }
2591   { stream }
```

```

2591         {
2592             stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2593             { \_pdf_exp_not_ii:nn #2 }
2594         }
2595     }
2596 }
2597 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
2598 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2599 \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
2600 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2601 {
2602   \*luatex
2603   \tex_immediate:D \tex_pdfextension:D obj ~
2604   \*pdftex
2605   \tex_immediate:D \tex_pdfobj:D
2606   \*pdftex
2607   \str_case:nn
2608     {#1}
2609     {
2610       { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2611       { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2612       { fstream }
2613       {
2614         {
2615             stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2616             file ~ { \_pdf_exp_not_ii:nn #2 }
2617         }
2618         { stream }
2619         {
2620             stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2621             { \_pdf_exp_not_ii:nn #2 }
2622         }
2623       }
2624     }
2625 \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.

```

2626 \cs_new:Npx \_pdf_backend_object_last:
2627 {
2628   \exp_not:N \int_value:w
2629   \*luatex
2630   \exp_not:N \tex_pdffeedback:D lastobj ~
2631   \*pdftex
2632   \exp_not:N \tex_pdflastobj:D
2633   \c_space_tl 0 ~ R
2634 }
2635 }
2636

```

(End definition for `_pdf_backend_object_last:.`)

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

```

2637 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2638 {
2639   \exp_not:N \int_value:w
2640   <*luatex>
2641   \exp_not:N \tex_pdffeedback:D pageref
2642   </luatex>
2643   <*pdftex>
2644   \exp_not:N \tex_pdfpageref:D
2645   </pdftex>
2646   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2647 }

```

(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
2648 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2649 {
2650   \tex_global:D
2651   <*luatex>
2652   \tex_pdfvariable:D compresslevel
2653   </luatex>
2654   <*pdftex>
2655   \tex_pdfcompresslevel:D
2656   </pdftex>
2657   \int_value:w \int_eval:n {#1} \scan_stop:
2658 }
2659 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2660 {
2661   \bool_if:nTF {#1}
2662     { \_pdf_backend_objcompresslevel:n { 2 } }
2663     { \_pdf_backend_objcompresslevel:n { 0 } }
2664 }
2665 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2666 {
2667   \tex_global:D
2668   <*luatex>
2669   \tex_pdfvariable:D objcompresslevel
2670   </luatex>
2671   <*pdftex>
2672   \tex_pdfobjcompresslevel:D
2673   </pdftex>
2674   #1 \scan_stop:
2675 }

```

(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_major_gset:n
\_pdf_backend_version_minor_gset:n
2676 \cs_new_protected:Npx \_pdf_backend_version_major_gset:n #1

```

```

2677 {
2678 <*luatex>
2679   \int_compare:nNnT \tex luatexversion:D > { 106 }
2680   {
2681     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2682     \exp_not:N \int_eval:n {#1} \scan_stop:
2683   }
2684 </luatex>
2685 <*pdftex>
2686   \cs_if_exist:NT \tex_pdfmajorversion:D
2687   {
2688     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2689     \exp_not:N \int_eval:n {#1} \scan_stop:
2690   }
2691 </pdftex>
2692 }
2693 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2694 {
2695   \tex_global:D
2696   <*luatex>
2697     \tex_pdfvariable:D minorversion
2698   </luatex>
2699   <*pdftex>
2700     \tex_pdfminorversion:D
2701   </pdftex>
2702   \int_eval:n {#1} \scan_stop:
2703 }

```

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

__pdf_backend_version_major: As above.
 __pdf_backend_version_minor:

```

2704 \cs_new:Npx \__pdf_backend_version_major:
2705 {
2706 <*luatex>
2707   \int_compare:nNnTF \tex luatexversion:D > { 106 }
2708   { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2709   { 1 }
2710 </luatex>
2711 <*pdftex>
2712   \cs_if_exist:NTF \tex_pdfmajorversion:D
2713   { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2714   { 1 }
2715 </pdftex>
2716 }
2717 \cs_new:Npn \__pdf_backend_version_minor:
2718 {
2719   \tex_the:D
2720 <*luatex>
2721   \tex_pdfvariable:D minorversion
2722 </luatex>
2723 <*pdftex>
2724   \tex_pdfminorversion:D
2725 </pdftex>
2726 }

```

(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:`

```
2727 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2728   { \_kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2729 \cs_new_protected:Npn \_pdf_backend_emc:
2730   { \_kernel_backend_literal_page:n { EMC } }
```

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

```
2731 </luatex | pdftex>
```

6.4 dvipdfmx backend

```
2732 <*dvipdfmx | xetex>
```

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

```
\_pdf_backend:x
2733 \cs_new_protected:Npx \_pdf_backend:n #1
2734   { \_kernel_backend_literal:n { pdf: #1 } }
2735 \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`

```
2736 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2737   { \_pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2738 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2739   { \_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`
`\g__pdf_backend_object_prop`

For tracking objects to allow finalisation.

```
2740 \int_new:N \g__pdf_backend_object_int
2741 \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`

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

```
2742 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2
2743   {
2744     \int_gincr:N \g__pdf_backend_object_int
2745     \int_const:cn
2746     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2747     { \g__pdf_backend_object_int }
2748     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2749   }
2750 \cs_new:Npn \_pdf_backend_object_ref:n #1
2751   { @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
2752 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2753 {
2754   \exp_args:Nx \__pdf_backend_object_write:nnn
2755   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2756 }
2757 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2758 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2759 {
2760   \use:c { __pdf_backend_object_write_ #1 :nn }
2761   { \__pdf_backend_object_ref:n {#2} } {#3}
2762 }
2763 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2764 {
2765   \__pdf_backend:x
2766   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2767 }
2768 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2769 {
2770   \__pdf_backend:x
2771   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2772 }
2773 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2774 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2775 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2776 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2777 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2778 {
2779   \__pdf_backend:x
2780   {
2781     #1 stream ~ #2 ~
2782     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2783   }
2784 }

```

(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
2785 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2786 {
2787   \int_gincr:N \g__pdf_backend_object_int
2788   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2789   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2790   {#2}
2791 }
2792 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for `_pdf_backend_object_now:nn`.)

`__pdf_backend_object_last:`

```

2793 \cs_new:Npn \__pdf_backend_object_last:
2794 { @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/X_YTeX.

```
2795 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2796 { @page #1 }
```

(End definition for _pdf_backend_pageobject_ref:n.)

6.4.3 Annotations

\g__pdf_landscape_bool There is a bug in dvipdfmx/X_YTeX which means annotations do not rotate. As such, we need to know if landscape is active.

```
2797 \bool_new:N \g__pdf_landscape_bool
2798 \cs_if_exist:NT \landscape
2799 {
2800   \tl_put_right:Nn \landscape
2801     { \bool_gset_true:N \g__pdf_landscape_bool }
2802   \tl_put_left:Nn \endlandscape
2803     { \bool_gset_false:N \g__pdf_landscape_bool }
2804 }
```

(End definition for \g__pdf_landscape_bool.)

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

```
2805 \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. The only wrinkle is landscape: we have to adjust by hand.

```
2806 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2807 {
2808   \bool_if:NTF \g__pdf_landscape_bool
2809   {
2810     \box_move_up:nn {#2}
2811     {
2812       \vbox:n
2813       {
2814         \_pdf_backend_annotation_aux:nnnn
2815         { #2 + #3 } {#1} { Opt } {#4}
2816       }
2817     }
2818   }
2819   { \_pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4} }
2820 }
2821 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2822 {
2823   \int_gincr:N \g_pdf_backend_object_int
2824   \int_gset_eq:NN \g_pdf_backend_annotation_int \g_pdf_backend_object_int
2825   \_pdf_backend:x
2826   {
2827     ann ~ @pdf.obj \int_use:N \g_pdf_backend_object_int \c_space_tl
2828     width ~ \dim_eval:n {#1} ~
2829     height ~ \dim_eval:n {#2} ~
```

```

2830         depth ~ \dim_eval:n {#3} ~
2831         <</Type/Annot #4 >>
2832     }
2833 }

```

(End definition for _pdf_backend_annotation:nnnn and _pdf_backend_annotation_aux:nnnn.)

_pdf_backend_annotation_last:

```

2834 \cs_new:Npn \_pdf_backend_annotation_last:
2835 { @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.

```

2836 \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:
2837 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2838 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2839 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2840 { \_pdf_backend_link_begin:n {#1#2} }
2841 \cs_new_protected:Npx \_pdf_backend_link_begin:n #1
2842 {
2843     \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2844     {
2845         \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int
2846     }
2847     \_pdf_backend:x
2848     {
2849         bann ~
2850         \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2851         {
2852             @pdf.lnk
2853             \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int
2854             \c_space_tl
2855         }
2856         <<
2857         /Type /Annot
2858         #1
2859         >>
2860     }
2861 }
2862 \cs_new_protected:Npn \_pdf_backend_link_end:
2863 { \_pdf_backend:n { eann } }

```

(End definition for _pdf_backend_link_begin_goto:nnw and others.)

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

```

2864 \cs_new:Npx \_pdf_backend_link_last:
2865 {
2866     \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2867     {
2868         @pdf.lnk

```

```

2869         \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
2870     }
2871 }

```

(End definition for __pdf_backend_link_last:.)

__pdf_backend_link_margin:n Pass to dvipdfmx.

```

2872 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2873 { \__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:nnnn
\__pdf_backend_destination_aux:nnnn

```

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

```

2874 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2875 {
2876     \__pdf_backend:x
2877     {
2878         dest ~ ( \exp_not:n {#1} )
2879         [
2880             @thispage
2881             \str_case:nnF {#2}
2882             {
2883                 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2884                 { fit } { /Fit }
2885                 { fitb } { /FitB }
2886                 { fitbh } { /FitBH }
2887                 { fitbv } { /FitBV ~ @xpos }
2888                 { fith } { /FitH ~ @ypos }
2889                 { fitv } { /FitV ~ @xpos }
2890                 { fitr } { /Fit }
2891             }
2892             { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2893         ]
2894     }
2895 }
2896 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2897 {
2898     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2899     { \dim_eval:n {#2} } {#1} {#3} {#4}
2900 }
2901 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2902 {
2903     \vbox_to_zero:n
2904     {
2905         \__kernel_kern:n {#4}
2906         \hbox:n
2907         {
2908             \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2909             \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2910         }
2911         \tex_vss:D

```

```

2912     }
2913     \__kernel_kern:n {#1}
2914     \vbox_to_zero:n
2915     {
2916         \__kernel_kern:n { -#3 }
2917         \hbox:n
2918         {
2919             \__pdf_backend:n
2920             {
2921                 dest ~ (#2)
2922                 [
2923                     @thispage
2924                     /FitR ~
2925                     @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2926                     @xpos ~ @ypos
2927                 ]
2928             }
2929         }
2930         \tex_vss:D
2931     }
2932     \__kernel_kern:n { -#1 }
2933 }

```

(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
_pdf_backend_compress_objects:n

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

```

2934 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2935 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2936 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2937 {
2938     \bool_if:nF {#1}
2939     { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2940 }

```

(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.

```

2941 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2942 {
2943     \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
2944     \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
2945 }
2946 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2947 {
2948     \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
2949     \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
2950 }

```

(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.

```

2951 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2952 \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` Simple wrappers. May need refinement: see <https://chat.stackexchange.com/transcript/message/49970158#49970158>.
`_pdf_backend_emc:`

```
2953 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2954 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2955 \cs_new_protected:Npn \_pdf_backend_emc:
2956 { \__kernel_backend_literal_page:n { EMC } }
```

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

```
2957 </dvipdfmx | xetex>
```

6.5 dvisvgm backend

```
2958 <*dvisvgm>
```

6.5.1 Catalogue entries

`_pdf_backend_catalog_gput:nn`
`_pdf_backend_info_gput:nn`

No-op.

```
2959 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2 { }
2960 \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

`_pdf_backend_object_new:nn`
`_pdf_backend_object_ref:n`
`_pdf_backend_object_write:nn`
`_pdf_backend_object_write:nx`
`_pdf_backend_object_now:nn`
`_pdf_backend_object_now:nx`
`_pdf_backend_object_last:`
`_pdf_backend_pageobject_ref:n`

All no-ops here.

```
2961 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2 { }
2962 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
2963 \cs_new_protected:Npn \_pdf_backend_object_write:nn #1#2 { }
2964 \cs_new_protected:Npn \_pdf_backend_object_write:nx #1#2 { }
2965 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
2966 \cs_new_protected:Npn \_pdf_backend_object_now:nx #1#2 { }
2967 \cs_new:Npn \_pdf_backend_object_last: { }
2968 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1 { }
```

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

6.5.3 Structure

`_pdf_backend_compresslevel:n`
`_pdf_backend_compress_objects:n`

These are all no-ops.

```
2969 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1 { }
2970 \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`
`_pdf_backend_version_minor_gset:n`

Data not available!

```
2971 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
2972 \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:
2973 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2974 \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:
2975 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2976 \cs_new_protected:Npn \__pdf_backend_emc: { }

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

2977 </divisvgn>
2978 </package>

```

7 l3backend-opacity Implementation

```

2979 <*package>
2980 <@@=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.

```

2981 <*dvips>

```

```

\__opacity_backend_select:n No stack so set values directly.
\__opacity_backend_select_aux:n
2982 \cs_new_protected:Npn \__opacity_backend_select:n #1
2983 {
2984   \exp_args:Nx \__opacity_backend_select_aux:n
2985   { \fp_eval:n { min(max(0,#1),1) } }
2986 }
2987 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
2988 {
2989   \__kernel_backend_postscript:n
2990   { #1 ~ .setfillconstantalpha ~ #1 ~ .setstrokeconstantalpha }
2991 }

(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
2992 \cs_new_protected:Npn \__opacity_backend_fill:n #1
2993 { \__opacity_backend:xn { \fp_eval:n { min(max(0,#1),1) } } { fill } }
\__opacity_backend:nn
2994 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
2995 { \__opacity_backend:xn { \fp_eval:n { min(max(0,#1),1) } } { stroke } }
\__opacity_backend:xn
2996 \cs_new_protected:Npn \__opacity_backend:nn #1#2
2997 {
2998   \__kernel_backend_postscript:n { #1 ~ .set #2 constantalpha }
2999 }
3000 \cs_generate_variant:Nn \__opacity_backend:nn { x }

(End definition for \__opacity_backend_fill:n, \__opacity_backend_stroke:n, and \__opacity_backend:nn.)

3001 </dvips>

```

3002 \langle *dvi \rangle pdfmx | luatex | pdftex | xetex \rangle

\backslash c_ opacity_backend_stack_int Set up a stack.

```
3003 \cs_if_exist:NT \pdfmanagement_add:nnn
3004 {
3005   \__kernel_color_backend_stack_init:Nnn \c_ opacity_backend_stack_int
3006   { page ~ direct } { /opacity 1 ~ gs }
3007   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3008   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3009 }
```

(End definition for \backslash c_ opacity_backend_stack_int.)

\backslash l_ opacity_backend_fill_tl We use tl here for speed: at the backend, this should be reasonable.

```
\l_ opacity_backend_stroke_tl
3010 \tl_new:N \l_ opacity_backend_fill_tl
3011 \tl_new:N \l_ opacity_backend_stroke_tl
```

(End definition for \backslash l_ opacity_backend_fill_tl and \backslash l_ opacity_backend_stroke_tl.)

\backslash _ 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:
3012 \cs_new_protected:Npn \_ opacity_backend_select:n #1
3013 {
3014   \exp_args:Nx \_ opacity_backend_select_aux:n
3015   { \fp_eval:n { min(max(0,#1),1) } }
3016 }
3017 \cs_new_protected:Npn \_ opacity_backend_select_aux:n #1
3018 {
3019   \tl_set:Nn \l_ opacity_backend_fill_tl {#1}
3020   \tl_set:Nn \l_ opacity_backend_stroke_tl {#1}
3021   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3022   { opacity #1 }
3023   { << /ca ~ #1 /CA ~ #1 >> }
3024   \__kernel_color_backend_stack_push:nn \c_ opacity_backend_stack_int
3025   { /opacity #1 ~ gs }
3026   \group_insert_after:N \_ opacity_backend_reset:
3027 }
3028 \cs_if_exist:NF \pdfmanagement_add:nnn
3029 {
3030   \cs_gset_protected:Npn \_ opacity_backend_select_aux:n #1 { }
3031 }
3032 \cs_new_protected:Npn \_ opacity_backend_reset:
3033 { \__kernel_color_backend_stack_pop:n \c_ opacity_backend_stack_int }
```

(End definition for \backslash _ opacity_backend_select:n, \backslash _ opacity_backend_select_aux:n, and \backslash _ opacity_backend_reset:.)

\backslash _ opacity_backend_fill:n For separate fill and stroke, we need to work out if we need to do more work or if we can stick to a single setting.

```
\_ opacity_backend_stroke:n
\_ opacity_backend_fillstroke:nn
\_ opacity_backend_fillstroke:xx
3034 \cs_new_protected:Npn \_ opacity_backend_fill:n #1
3035 {
3036   \_ opacity_backend_fill_stroke:xx
3037   { \fp_eval:n { min(max(0,#1),1) } }
3038   \l_ opacity_backend_stroke_tl
3039 }
3040 \cs_new_protected:Npn \_ opacity_backend_stroke:n #1
```

```

3041 {
3042   \_opacity_backend_fill_stroke:xx
3043   \l\_opacity_backend_fill_tl
3044   { \fp_eval:n { min(max(0,#1),1) } }
3045 }
3046 \cs_new_protected:Npn \_opacity_backend_fill_stroke:nn #1#2
3047 {
3048   \str_if_eq:nnTF {#1} {#2}
3049   { \_opacity_backend_select_aux:n {#1} }
3050   {
3051     \tl_set:Nn \l\_opacity_backend_fill_tl {#1}
3052     \tl_set:Nn \l\_opacity_backend_stroke_tl {#2}
3053     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3054     { opacity.fill #1 }
3055     { << /ca ~ #1 >> }
3056     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3057     { opacity.stroke #1 }
3058     { << /CA ~ #2 >> }
3059     \_kernel_color_backend_stack_push:nn \c\_opacity_backend_stack_int
3060     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3061     \group_insert_after:N \_opacity_backend_reset:
3062   }
3063 }
3064 \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.)

3065 </dvipdfmx | luatex | pdftex | xetex>
3066 <*dvipdfmx | xdvipdfmx>

```

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

```

3067 \int_compare:nNnT \c\_kernel_sys_dvipdfmx_version_int < { 20201111 }
3068 {
3069   \cs_gset_protected:Npn \_opacity_backend_select_aux:n #1
3070   {
3071     \tl_set:Nn \l\_opacity_backend_fill_tl {#1}
3072     \tl_set:Nn \l\_opacity_backend_stroke_tl {#1}
3073     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3074     { opacity #1 }
3075     { << /ca ~ #1 /CA ~ #1 >> }
3076     \_kernel_backend_literal_pdf:n { /opacity #1 ~ gs }
3077   }
3078   \cs_gset_protected:Npn \_opacity_backend_fill_stroke:nn #1#2
3079   {
3080     \str_if_eq:nnTF {#1} {#2}
3081     { \_opacity_backend_select_aux:n {#1} }
3082     {
3083       \tl_set:Nn \l\_opacity_backend_fill_tl {#1}
3084       \tl_set:Nn \l\_opacity_backend_stroke_tl {#2}
3085       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3086       { opacity.fill #1 }
3087       { << /ca ~ #1 >> }
3088       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3089       { opacity.stroke #1 }

```



```

3090         { << /CA ~ #2 >> }
3091         \_kernel_backend_literal_pdf:n
3092         { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3093     }
3094 }
3095 }

(End definition for \_opacity_backend_select:n.)

3096 </dvipdfmx | xdvipdfmx>
3097 <*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
3098 \cs_new_protected:Npn \_opacity_backend_select:n #1
3099 { \_opacity_backend:nn {#1} { } }
3100 \cs_new_protected:Npn \_opacity_backend_fill:n #1
3101 { \_opacity_backend:nn {#1} { fill- } }
3102 \cs_new_protected:Npn \_opacity_backend_stroke:n #1
3103 { \_opacity_backend:nn { {#1} } { stroke- } }
3104 \cs_new_protected:Npn \_opacity_backend:nn #1#2
3105 { \_kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

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

3106 </dvisvgm>
3107 </package>

```

8 l3backend-header Implementation

```

3108 <*dvips & header>

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

```

```

3109 /color.sc { } def

```

(End definition for color.sc. This function is documented on page ??.)

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

```

3110 TeXDict begin
3111 /TeXcolorseparation { setcolor } def
3112 end

```

(End definition for TeXcolorseparation and separation. These functions are documented on page ??.)

pdf.globaldict A small global dictionary for backend use.

```

3113 true setglobal
3114 /pdf.globaldict 4 dict def
3115 false setglobal

```

(End definition for pdf.globaldict. This function is documented on page ??.)

pdf.cvs	Small utilities for PostScript manipulations. Conversion to DVI dimensions is done here
pdf.dvi.pt	to allow for Resolution. The total height of a rectangle (an array) needs a little maths,
pdf.pt.dvi	in contrast to simply extracting a value.
pdf.rect.ht	<pre> 3116 /pdf.cvs { 65534 string cvs } def 3117 /pdf.dvi.pt { 72.27 mul Resolution div } def 3118 /pdf.pt.dvi { 72.27 div Resolution mul } def 3119 /pdf.rect.ht { dup 1 get neg exch 3 get add } def </pre>
	<i>(End definition for pdf.cvs and others. These functions are documented on page ??.)</i>
pdf.linkmargin	Settings which are defined up-front in SDict.
pdf.linkdp.pad	<pre> 3120 /pdf.linkmargin { 1 pdf.pt.dvi } def </pre>
pdf.linkht.pad	<pre> 3121 /pdf.linkdp.pad { 0 } def 3122 /pdf.linkht.pad { 0 } def </pre>
	<i>(End definition for pdf.linkmargin, pdf.linkdp.pad, and pdf.linkht.pad. These functions are documented on page ??.)</i>
pdf.rect	Functions for marking the limits of an annotation/link, plus drawing the border. We
pdf.save.ll	separate links for generic annotations to support adding a margin and setting a minimal
pdf.save.ur	size.
pdf.save.linkll	<pre> 3123 /pdf.rect </pre>
pdf.save.linkur	<pre> 3124 { /Rect [pdf.llx pdf.lly pdf.urx pdf.ury] } def </pre>
pdf.llx	<pre> 3125 /pdf.save.ll </pre>
pdf.lly	<pre> 3126 { </pre>
pdf.urx	<pre> 3127 currentpoint </pre>
pdf.ury	<pre> 3128 /pdf.lly exch def 3129 /pdf.llx exch def 3130 } </pre>
	<pre> 3131 def 3132 /pdf.save.ur </pre>
	<pre> 3133 { </pre>
	<pre> 3134 currentpoint </pre>
	<pre> 3135 /pdf.ury exch def 3136 /pdf.urx exch def 3137 } </pre>
	<pre> 3138 def 3139 /pdf.save.linkll </pre>
	<pre> 3140 { </pre>
	<pre> 3141 currentpoint 3142 pdf.linkmargin add 3143 pdf.linkdp.pad add 3144 /pdf.lly exch def 3145 pdf.linkmargin sub 3146 /pdf.llx exch def 3147 } </pre>
	<pre> 3148 def 3149 /pdf.save.linkur </pre>
	<pre> 3150 { </pre>
	<pre> 3151 currentpoint 3152 pdf.linkmargin sub 3153 pdf.linkht.pad sub 3154 /pdf.ury exch def 3155 pdf.linkmargin add </pre>

```

3156     /pdf.urx exch def
3157   }
3158   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 function as it comes up a lot, and as this makes it easier to adjust if we need additional effects. We also need a more complex approach to convert a co-ordinate pair correctly when defining a rectangle: this can otherwise be out when using a landscape page. (Thanks to Alexander Grahn for the approach here.)

```

pdf.dev.x 3159 /pdf.dest.anchor
pdf.dev.y 3160 {
pdf.tmpa 3161   currentpoint exch
pdf.tmpb 3162   pdf.dvi.pt 72 add
pdf.tmpc 3163   /pdf.dest.x exch def
pdf.tmpd 3164   pdf.dvi.pt
3165   vsize 72 sub exch sub
3166   /pdf.dest.y exch def
3167 }
3168 def
3169 /pdf.dest.point
3170 { pdf.dest.x pdf.dest.y } def
3171 /pdf.dest2device
3172 {
3173   /pdf.dest.y exch def
3174   /pdf.dest.x exch def
3175   matrix currentmatrix
3176   matrix defaultmatrix
3177   matrix invertmatrix
3178   matrix concatmatrix
3179   cvx exec
3180   /pdf.dev.y exch def
3181   /pdf.dev.x exch def
3182   /pdf.tmpd exch def
3183   /pdf.tmpc exch def
3184   /pdf.tmpb exch def
3185   /pdf.tmpa exch def
3186   pdf.dest.x pdf.tmpa mul
3187   pdf.dest.y pdf.tmpc mul add
3188   pdf.dev.x add
3189   pdf.dest.x pdf.tmpb mul
3190   pdf.dest.y pdf.tmpd mul add
3191   pdf.dev.y add
3192 }
3193 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 can be done by hooking into a and x operations: those names have to be retained. The boundary is stored at the end of the operation. Special effort is needed at the start and end of pages (or rather galleys), such that everything works properly.

```

pdf.bordertracking.begin 3194 /pdf.bordertracking false def
pdf.bordertracking.end
pdf.leftboundary
pdf.rightboundary
pdf.brokenlink.rect
pdf.brokenlink.skip
pdf.brokenlink.dict
pdf.bordertracking.endpage
pdf.bordertracking.continue
pdf.originx
pdf.originy

```

```

3195 /pdf.bordertracking.begin
3196 {
3197     SDict /pdf.bordertracking true put
3198     SDict /pdf.leftboundary undef
3199     SDict /pdf.rightboundary undef
3200     /a where
3201     {
3202         /a
3203         {
3204             currentpoint pop
3205             SDict /pdf.rightboundary known dup
3206             {
3207                 SDict /pdf.rightboundary get 2 index lt
3208                 { not }
3209                 if
3210             }
3211             if
3212             { pop }
3213             { SDict exch /pdf.rightboundary exch put }
3214             ifelse
3215             moveto
3216             currentpoint pop
3217             SDict /pdf.leftboundary known dup
3218             {
3219                 SDict /pdf.leftboundary get 2 index gt
3220                 { not }
3221                 if
3222             }
3223             if
3224             { pop }
3225             { SDict exch /pdf.leftboundary exch put }
3226             ifelse
3227         }
3228         put
3229     }
3230     if
3231 }
3232 def
3233 /pdf.bordertracking.end
3234 {
3235     /a where { /a { moveto } put } if
3236     /x where { /x { 0 exch rmoveto } put } if
3237     SDict /pdf.leftboundary known
3238     { pdf.outerbox 0 pdf.leftboundary put }
3239     if
3240     SDict /pdf.rightboundary known
3241     { pdf.outerbox 2 pdf.rightboundary put }
3242     if
3243     SDict /pdf.bordertracking false put
3244 }
3245 def
3246 /pdf.bordertracking.endpage
3247 {
3248     pdf.bordertracking

```

```

3249 {
3250     pdf.bordertracking.end
3251     true setglobal
3252     pdf.globaldict
3253     /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3254     pdf.globaldict
3255     /pdf.brokenlink.skip pdf.baselineskip put
3256     pdf.globaldict
3257     /pdf.brokenlink.dict
3258     pdf.link.dict pdf.cvs put
3259     false setglobal
3260     mark pdf.link.dict cvx exec /Rect
3261     [
3262         pdf.llx
3263         pdf.lly
3264         pdf.outerbox 2 get pdf.linkmargin add
3265         currentpoint exch pop
3266         pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3267     ]
3268     /ANN pdf.pdfmark
3269 }
3270 if
3271 }
3272 def
3273 /pdf.bordertracking.continue
3274 {
3275     /pdf.link.dict pdf.globaldict
3276     /pdf.brokenlink.dict get def
3277     /pdf.outerbox pdf.globaldict
3278     /pdf.brokenlink.rect get def
3279     /pdf.baselineskip pdf.globaldict
3280     /pdf.brokenlink.skip get def
3281     pdf.globaldict dup dup
3282     /pdf.brokenlink.dict undef
3283     /pdf.brokenlink.skip undef
3284     /pdf.brokenlink.rect undef
3285     currentpoint
3286     /pdf.originy exch def
3287     /pdf.originx exch def
3288     /a where
3289     {
3290         /a
3291         {
3292             moveto
3293             SDict
3294             begin
3295                 currentpoint pdf.originy ne exch
3296                 pdf.originx ne or
3297                 {
3298                     pdf.save.linkll
3299                     /pdf.lly
3300                     pdf.lly pdf.outerbox 1 get sub def
3301                     pdf.bordertracking.begin
3302                 }

```

```

3303         if
3304         end
3305     }
3306     put
3307 }
3308 if
3309 /x where
3310 {
3311     /x
3312     {
3313         0 exch rmoveto
3314         SDict
3315         begin
3316         currentpoint
3317         pdf.originy ne exch pdf.originx ne or
3318         {
3319             pdf.save.link11
3320             /pdf.lly
3321             pdf.lly pdf.outerbox 1 get sub def
3322             pdf.bordertracking.begin
3323         }
3324         if
3325         end
3326     }
3327     put
3328 }
3329 if
3330 }
3331 def

```

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

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

```

3332 /pdf.breaklink
3333 {
3334     pop
3335     counttomark 2 mod 0 eq
3336     {
3337         counttomark /pdf.count exch def
3338         {
3339             pdf.count 0 eq { exit } if
3340             counttomark 2 roll
3341             1 index /Rect eq
3342             {
3343                 dup 4 array copy
3344                 dup dup
3345                 1 get
3346                 pdf.outerbox pdf.rect.ht
3347                 pdf.linkmargin 2 mul add sub
3348                 3 exch put

```

```

3349     dup
3350     pdf.outerbox 2 get
3351     pdf.linkmargin add
3352     2 exch put
3353   dup dup
3354     3 get
3355     pdf.outerbox pdf.rect.ht
3356     pdf.linkmargin 2 mul add add
3357     1 exch put
3358   /pdf.currentrect exch def
3359   pdf.breaklink.write
3360   {
3361     pdf.currentrect
3362     dup
3363       pdf.outerbox 0 get
3364       pdf.linkmargin sub
3365       0 exch put
3366     dup
3367       pdf.outerbox 2 get
3368       pdf.linkmargin add
3369       2 exch put
3370     dup dup
3371       1 get
3372       pdf.baselineskip add
3373       1 exch put
3374     dup dup
3375       3 get
3376       pdf.baselineskip add
3377       3 exch put
3378     /pdf.currentrect exch def
3379     pdf.breaklink.write
3380   }
3381   1 index 3 get
3382   pdf.linkmargin 2 mul add
3383   pdf.outerbox pdf.rect.ht add
3384   2 index 1 get sub
3385   pdf.baselineskip div round cvi 1 sub
3386   exch
3387 repeat
3388 pdf.currentrect
3389 dup
3390   pdf.outerbox 0 get
3391   pdf.linkmargin sub
3392   0 exch put
3393 dup dup
3394   1 get
3395   pdf.baselineskip add
3396   1 exch put
3397 dup dup
3398   3 get
3399   pdf.baselineskip add
3400   3 exch put
3401 dup 2 index 2 get 2 exch put
3402 /pdf.currentrect exch def

```

```

3403         pdf.breaklink.write
3404         SDict /pdf.pdfmark.good false put
3405         exit
3406     }
3407     { pdf.count 2 sub /pdf.count exch def }
3408     ifelse
3409 }
3410 loop
3411 }
3412 if
3413 /ANN
3414 }
3415 def
3416 /pdf.breaklink.write
3417 {
3418     counttomark 1 sub
3419     index /_objdef eq
3420     {
3421         counttomark -2 roll
3422         dup wcheck
3423         {
3424             readonly
3425             counttomark 2 roll
3426         }
3427         { pop pop }
3428         ifelse
3429     }
3430     if
3431     counttomark 1 add copy
3432     pop pdf.currentrect
3433     /ANN pdfmark
3434 }
3435 def

```

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

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

```

3436 /pdf.pdfmark
3437 {
3438     SDict /pdf.pdfmark.good true put
3439     dup /ANN eq
3440     {
3441         pdf.pdfmark.store
3442         pdf.pdfmark.dict
3443         begin
3444             Subtype /Link eq
3445             currentdict /Rect known and
3446             SDict /pdf.outerbox known and
3447             SDict /pdf.baselineskip known and
3448             {

```



```

3449         Rect 3 get
3450         pdf.linkmargin 2 mul add
3451         pdf.outerbox pdf.rect.ht add
3452         Rect 1 get sub
3453         pdf.baselineskip div round cvi 0 gt
3454         { pdf.breaklink }
3455         if
3456     }
3457     if
3458     end
3459     SDict /pdf.outerbox undef
3460     SDict /pdf.baselineskip undef
3461     currentdict /pdf.pdfmark.dict undef
3462 }
3463 if
3464 pdf.pdfmark.good
3465 { pdfmark }
3466 { cleartomark }
3467 ifelse
3468 }
3469 def
3470 /pdf.pdfmark.store
3471 {
3472     /pdf.pdfmark.dict 65534 dict def
3473     counttomark 1 add copy
3474     pop
3475     {
3476         dup mark eq
3477         {
3478             pop
3479             exit
3480         }
3481         {
3482             pdf.pdfmark.dict
3483             begin def end
3484         }
3485         ifelse
3486     }
3487     loop
3488 }
3489 def

```

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

```

3490 </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, 516, 517, 661, 662
- B**
- bool commands:
 - \bool_gset_false:N 1160, 1179, 1202, 1224, 1240, 1341, 1578, 1614, 2179, 2225, 2803
 - \bool_gset_true:N 1158, 1227, 1339, 1593, 2172, 2178, 2801
 - \bool_if:NTF 57, 659, 1170, 1174, 1190, 1193, 1197, 1208, 1215, 1219, 1231, 1235, 1352, 1357, 1362, 1552, 1597, 1710, 1745, 1855, 1897, 2167, 2182, 2187, 2192, 2808
 - \bool_if:nTF 2401, 2661, 2938
 - \bool_lazy_or:nnTF 1737, 1890
 - \bool_new:N 1161, 1228, 1342, 1594, 2155, 2156, 2797
 - \bool_set_false:N 1720, 1822, 1915, 1979
- box commands:
 - \box_dp:N 216, 218, 266, 268, 323, 325, 372, 374, 376, 378, 2204, 2237, 2238, 2263
 - \box_ht:N 218, 268, 325, 376, 378, 1757, 1952, 2209, 2248, 2249, 2265
 - \box_if_empty:NTF 2298
 - \box_move_down:nn 2129, 2204
 - \box_move_up:nn 2131, 2209, 2810
 - \box_new:N 2014, 2119, 2120
 - \box_set_dp:Nn 1677
 - \box_set_ht:Nn 1676
 - \box_set_wd:Nn 280, 1675
 - \box_use:N 223, 241, 255, 271, 298, 312, 328, 344, 356, 407, 421, 440, 1292, 1487, 1678, 2160
 - \box_wd:N 217, 225, 267, 273, 324, 330, 373, 375, 1756, 1951
- 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 ... 1248, 1372
 - \clist_map_function:nn 1621
- color internal commands:
 - __color_backend:nnn 1046
 - __color_backend_cmyk:w 1047
 - __color_backend_devicen_init:n 894
 - __color_backend_devicen_init:nnn 807, 894
 - __color_backend_devicen_init:w 894
 - __color_backend_fill:n 953, 980, 1010, 1028, 1035
 - __color_backend_fill_cmyk:n 953, 987, 1010, 1035
 - __color_backend_fill_devicen:nn 979, 1001, 1027, 1097
 - __color_backend_fill_gray:n 953, 987, 1010, 1035
 - __color_backend_fill_rgb:n 953, 987, 1010, 1035
 - __color_backend_fill_separation:nn 979, 987, 1027, 1097
 - \l__color_backend_fill_tl 621, 631, 961, 976
 - \c__color_backend_main_stack_int 508
 - __color_backend_pickup:N .. 448, 471
 - __color_backend_pickup:w 14, 448, 471
 - __color_backend_reset: 603, 623, 640, 964, 977, 987, 1019, 1044
 - __color_backend_rgb:w 1070
 - __color_backend_select:n 603, 655, 833
 - __color_backend_select:nn . 623, 834
 - __color_backend_select_cmyk:n .. 603, 623, 640
 - __color_backend_select_devicen:nn 654, 827, 833, 945

_color_backend_select_gray:n ..	_color_backend_stroke_separation:nn
..... 603, 623, 640 979, 987, 1027, 1097
_color_backend_select_rgb:n ...	\l_color_backend_stroke_tl
..... 603, 623, 640 621, 632, 963, 974
_color_backend_select_separation:nn	\g_color_model_int 677, 813, 858, 931
..... 654, 827, 833, 945	\c_color_model_range_CIELAB_tl .
_color_backend_separation_- 768, 803, 878, 885
init:n 657, 836, 918, 942	color.sc 603, 3109
_color_backend_separation_-	cs commands:
init:nnn 657	\cs_generate_variant:Nn .. 49, 53,
_color_backend_separation_-	56, 91, 130, 135, 162, 193, 199, 553,
init:nnnn 657	590, 670, 1107, 1302, 1496, 1869,
_color_backend_separation_-	1926, 1942, 2018, 2055, 2114, 2597,
init:nnnnn 657, 829, 836	2625, 2735, 2757, 2792, 3000, 3064
_color_backend_separation_-	\cs_gset:Npx 2943, 2948
init:nw 657	\cs_gset_eq:NN 647,
_color_backend_separation_-	648, 948, 994, 995, 1001, 1003, 1005
init:w 657	\cs_gset_protected:Npn 642, 649, 891, 947,
_color_backend_separation_-	989, 996, 998, 1000, 3030, 3069, 3078
init_/DeviceCMYK:nnn 657	\cs_if_exist:NTF 27, 59,
_color_backend_separation_-	449, 472, 516, 530, 661, 852, 889,
init_/DeviceGray:nnn 657	925, 2294, 2686, 2712, 2798, 3003, 3028
_color_backend_separation_-	\cs_if_exist_use:NTF 38, 683
init_/DeviceRGB:nnn 657	\cs_new:Npn 692, 694, 696,
_color_backend_separation_-	698, 705, 711, 713, 719, 736, 743,
init_aux:nnnnn 657	745, 936, 1253, 1377, 1625, 1955,
_color_backend_separation_-	1964, 2008, 2033, 2115, 2117, 2150,
init_CIELAB:nnn 657, 829, 836	2319, 2413, 2414, 2567, 2598, 2599,
_color_backend_separation_-	2717, 2750, 2793, 2795, 2834, 2951,
init_CIELAB:nnnnnn 830	2952, 2962, 2967, 2968, 2973, 2974
_color_backend_separation_-	\cs_new:Npx 2434, 2469, 2626, 2637, 2704, 2864
init_count:n 657	\cs_new_eq:NN .. 46, 656, 835, 942,
_color_backend_separation_-	983, 984, 1031, 1032, 1099, 1100,
init_count:w 657	1106, 1301, 1307, 1308, 1495, 1502,
_color_backend_separation_-	1687, 1716, 1767, 1768, 1810, 1818,
init_Device:Nn 657	1840, 1911, 1968, 1975, 2007, 2160
\g_color_backend_stack_int ... 508	\cs_new_protected:Npn ... 47, 51,
\l_color_backend_stack_int	54, 64, 70, 75, 77, 81, 92, 102, 111,
.... 505, 532, 538, 633, 637, 962, 975	120, 133, 136, 138, 140, 160, 165,
_color_backend_stroke:n 953, 982, 987	174, 184, 194, 205, 227, 229, 244,
_color_backend_stroke_cmyk:n ..	260, 275, 277, 303, 317, 332, 334,
..... 953, 1010, 1046	347, 361, 411, 424, 448, 466, 471,
_color_backend_stroke_cmyk:w 1046	479, 509, 544, 554, 566, 580, 591,
_color_backend_stroke_devicen:nn	603, 605, 607, 609, 617, 623, 625,
..... 979, 1005, 1027, 1097	627, 629, 636, 654, 671, 761, 807,
_color_backend_stroke_gray:n ..	827, 828, 829, 830, 833, 836, 863,
..... 953, 1010, 1046	867, 894, 953, 955, 957, 959, 966,
_color_backend_stroke_gray_-	968, 970, 972, 979, 981, 1010, 1012,
aux:n 1046	1014, 1016, 1021, 1023, 1025, 1027,
_color_backend_stroke_rgb:n ...	1029, 1035, 1037, 1039, 1041, 1046,
..... 953, 1010, 1046	1048, 1059, 1067, 1069, 1071, 1097,
_color_backend_stroke_rgb:w . 1046	1098, 1108, 1113, 1118, 1120, 1122,

1130, 1138, 1147, 1157, 1159, 1162,
 1164, 1181, 1186, 1204, 1226, 1229,
 1242, 1255, 1260, 1262, 1264, 1266,
 1268, 1270, 1272, 1274, 1279, 1303,
 1305, 1309, 1314, 1319, 1329, 1338,
 1340, 1343, 1345, 1347, 1349, 1354,
 1359, 1364, 1366, 1379, 1384, 1386,
 1388, 1390, 1392, 1394, 1396, 1398,
 1409, 1434, 1446, 1458, 1470, 1477,
 1497, 1503, 1508, 1513, 1524, 1534,
 1544, 1546, 1548, 1550, 1581, 1583,
 1588, 1590, 1592, 1595, 1616, 1627,
 1640, 1642, 1644, 1646, 1648, 1650,
 1652, 1654, 1656, 1664, 1688, 1702,
 1717, 1729, 1734, 1762, 1774, 1787,
 1797, 1812, 1819, 1827, 1838, 1842,
 1845, 1860, 1870, 1905, 1912, 1918,
 1924, 1927, 1934, 1943, 1948, 1956,
 1969, 1976, 1982, 1984, 1986, 1997,
 2016, 2019, 2021, 2025, 2035, 2056,
 2061, 2066, 2071, 2081, 2086, 2094,
 2122, 2127, 2159, 2161, 2163, 2165,
 2170, 2185, 2190, 2227, 2256, 2275,
 2284, 2321, 2328, 2354, 2359, 2387,
 2399, 2411, 2412, 2415, 2417, 2421,
 2445, 2447, 2449, 2460, 2480, 2490,
 2513, 2527, 2537, 2548, 2569, 2600,
 2648, 2659, 2665, 2693, 2727, 2729,
 2736, 2738, 2742, 2752, 2758, 2763,
 2768, 2773, 2775, 2777, 2785, 2806,
 2821, 2837, 2839, 2862, 2872, 2874,
 2896, 2901, 2934, 2936, 2941, 2946,
 2953, 2955, 2959, 2960, 2961, 2963,
 2964, 2965, 2966, 2969, 2970, 2971,
 2972, 2975, 2976, 2982, 2987, 2992,
 2994, 2996, 3012, 3017, 3032, 3034,
 3040, 3046, 3098, 3100, 3102, 3104
 \cs_new_protected:Npx
 512, 657, 1082, 2676, 2733, 2841
 \cs_set:Npn 144
 \cs_set_eq:NN 2315, 2316
 \cs_set_protected:Npn 451, 474

D

dim commands:

\dim_eval:n 2125, 2357,
 2429, 2430, 2431, 2488, 2523, 2524,
 2525, 2828, 2829, 2830, 2873, 2899
 \dim_max:nn 2235, 2246
 \dim_set:Nn ... 1756, 1757, 1951, 1952
 \dim_to_decimal:n .. 372, 373, 374,
 375, 376, 378, 1506, 1511, 1517,
 1518, 1519, 1520, 1529, 1530, 1531,
 1622, 1641, 2002, 2003, 2233, 2244,

2262, 2263, 2264, 2265, 2269, 2325
 \dim_to_decimal_in_bp:n
 216, 217, 218, 266, 267, 268,
 323, 324, 325, 1126, 1127, 1134,
 1135, 1142, 1143, 1151, 1152, 1153,
 1250, 1254, 1258, 1312, 1317, 1323,
 1324, 1325, 1333, 1334, 1374, 1378,
 1382, 1626, 1693, 1694, 1695, 1696,
 1832, 1833, 1834, 1835, 1884, 1885,
 1886, 1887, 1991, 1992, 1993, 1994
 draw internal commands:
 __draw_align_currentpoint... .. 34
 __draw_backend_add_to_path:n ...
 1503, 1549
 __draw_backend_begin:
 1108, 1303, 1497
 __draw_backend_box_use:Nnnnn ...
 30, 1279, 1477, 1664
 __draw_backend_cap_but:
 1242, 1366, 1616
 __draw_backend_cap_rectangle: ..
 1242, 1366, 1616
 __draw_backend_cap_round:
 1242, 1366, 1616
 __draw_backend_clip: 1162, 1343, 1548
 __draw_backend_closepath:
 1162, 1343, 1548
 __draw_backend_closestroke: ...
 1162, 1343, 1548
 __draw_backend_cm:nnnn 1274, 1287,
 1288, 1289, 1398, 1481, 1656, 1667
 __draw_backend_cm_aux:nnnn .. 1398
 __draw_backend_cm_decompose:nnnnN
 1404, 1433
 __draw_backend_cm_decompose_-
 auxi:nnnnN 1433
 __draw_backend_cm_decompose_-
 auxii:nnnnN 1433
 __draw_backend_cm_decompose_-
 auxiii:nnnnN 1433
 __draw_backend_curveto:nnnnnn ..
 1122, 1309, 1503
 __draw_backend_dash:n
 1242, 1366, 1616
 __draw_backend_dash_aux:nn .. 1616
 __draw_backend_dash_pattern:nn ..
 1242, 1366, 1616
 __draw_backend_discardpath: ...
 1162, 1343, 1548
 __draw_backend_end: 1108, 1303, 1497
 __draw_backend_evenodd_rule: ...
 1157, 1338, 1544
 __draw_backend_fill: 1162, 1343, 1548

__draw_backend_fillstroke:	
. 1162 , 1343 , 1548	
__draw_backend_join_bevel:	
. 1242 , 1366 , 1616	
__draw_backend_join_miter:	
. 1242 , 1366 , 1616	
__draw_backend_join_round:	
. 1242 , 1366 , 1616	
__draw_backend_lineto:nn	
. 1122 , 1309 , 1503	
__draw_backend_linewidth:n	
. 1242 , 1366 , 1616	
__draw_backend_literal:n	
. 1106 , 1111 , 1115 , 1119 ,	
1121 , 1124 , 1132 , 1140 , 1149 , 1163 ,	
1166 , 1167 , 1168 , 1169 , 1172 , 1178 ,	
1188 , 1195 , 1201 , 1206 , 1211 , 1212 ,	
1213 , 1214 , 1217 , 1223 , 1233 , 1239 ,	
1244 , 1257 , 1261 , 1263 , 1265 , 1267 ,	
1269 , 1271 , 1273 , 1276 , 1281 , 1282 ,	
1283 , 1284 , 1285 , 1286 , 1290 , 1291 ,	
1293 , 1294 , 1295 , 1296 , 1297 , 1301 ,	
1311 , 1316 , 1321 , 1331 , 1344 , 1346 ,	
1348 , 1351 , 1356 , 1361 , 1365 , 1368 ,	
1381 , 1385 , 1387 , 1389 , 1391 , 1393 ,	
1395 , 1397 , 1495 , 1555 , 1574 , 1600	
__draw_backend_miterlimit:n . . .	
. 1242 , 1366 , 1616	
__draw_backend_moveto:nn	
. 1122 , 1309 , 1503	
__draw_backend_nonzero_rule: . . .	
. 1157 , 1338 , 1544	
__draw_backend_path:n 1548	
__draw_backend_rectangle:nnnn . .	
. 1122 , 1309 , 1503	
__draw_backend_scope:n 1545 , 1547 ,	
1567 , 1607 , 1629 , 1641 , 1643 , 1645 ,	
1647 , 1649 , 1651 , 1653 , 1655 , 1658	
__draw_backend_scope_begin: . . .	
. 1118 , 1304 , 1307	
__draw_backend_scope_end:	
. 1118 , 1306 , 1307	
__draw_backend_stroke:	
. 1162 , 1343 , 1548	
\g__draw_clip_path_int	
. . . 1554 , 1557 , 1570 , 1599 , 1602 , 1610	
\g__draw_draw_clip_bool . . . 1162 , 1548	
\g__draw_draw_eor_bool	
. . . 1157 , 1174 , 1190 , 1197 , 1208 ,	
1219 , 1235 , 1338 , 1352 , 1357 , 1362	
\g__draw_draw_path_int 1548	
\g__draw_draw_path_tl	
. . . 1503 , 1559 , 1575 , 1577 , 1604 , 1613	
\g__draw_path_int 1563 , 1580	
	E
	\endlandscape 2802
	\errmessage 38
	\evensidemargin 2202
	exp commands:
	\exp_after:wN 151 , 457 , 1962
	\exp_args:Ne 707 , 2356 , 2898
	\exp_args:Nf 1247 , 1371 , 2124
	\exp_args:NNf 228 , 276 , 333
	\exp_args:Nnx 2111 , 2788
	\exp_args:NV 453
	\exp_args:Nx 1780 , 1801 ,
	2068 , 2083 , 2198 , 2754 , 2984 , 3014
	\exp_last_unbraced:Nx 462 , 476
	\exp_not:N 514 , 515 , 517 , 518 , 523 ,
	525 , 662 , 665 , 2436 , 2438 , 2441 ,
	2471 , 2473 , 2476 , 2628 , 2630 , 2633 ,
	2639 , 2641 , 2644 , 2681 , 2682 , 2688 ,
	2689 , 2708 , 2713 , 2845 , 2853 , 2869
	\exp_not:n 48 , 89 , 100 , 128 , 2059 ,
	2064 , 2350 , 2583 , 2584 , 2598 , 2599 ,
	2611 , 2612 , 2766 , 2771 , 2782 , 2878
	\ExplBackendFileDate 1
	F
	file commands:
	\file_compare_timestamp:nNnTF . . . 1789
	\file_parse_full_name:nNNN 1776 , 1799
	fp commands:
	\fp_compare:nNnTF
	. . . 235 , 282 , 288 , 340 , 1414 , 1427 , 1472
	\fp_eval:n 228 , 237 , 250 , 251 , 276 ,
	293 , 308 , 310 , 333 , 342 , 353 , 354 ,
	418 , 433 , 434 , 1054 , 1055 , 1056 ,
	1064 , 1077 , 1078 , 1079 , 1416 , 1421 ,
	1422 , 1429 , 1439 , 1440 , 1441 , 1442 ,
	1451 , 1452 , 1453 , 1454 , 1463 , 1464 ,
	1465 , 1466 , 2347 , 2510 , 2892 , 2985 ,
	2993 , 2995 , 3015 , 3037 , 3044 , 3105
	\fp_new:N 301 , 302
	\fp_set:Nn 281 , 284
	\fp_use:N 287 , 291 , 296
	\fp_zero:N 283
	\c_zero_fp 235 , 282 , 288 , 340 , 1414 , 1427
	G
	graphics commands:
	\graphics_bb_restore:nTF . . . 1731 , 1945
	\graphics_bb_save:n 1760 , 1953
	\l_graphics_decodearray_tl
 1708 , 1709 ,
	1719 , 1739 , 1743 , 1744 , 1821 , 1853 ,
	1854 , 1892 , 1895 , 1896 , 1914 , 1978
	\graphics_extract_bb:n
 1816 , 1823 , 1973 , 1980

\l_graphics_interpolate_bool . . .	__graphics_backend_include:nn 1982
. 1710 , 1720 , 1738 , 1745 ,	__graphics_backend_include_-
1822 , 1855 , 1891 , 1897 , 1915 , 1979	auxi:nn 1827
\l_graphics_llx_dim	__graphics_backend_include_-
. 1693 , 1832 , 1884 , 1991	auxii:nnn 1827
\l_graphics_lly_dim	__graphics_backend_include_-
. 1694 , 1833 , 1885 , 1992	auxiii:nnn 1827
\l_graphics_name_tl 1794	__graphics_backend_include_-
\l_graphics_page_int	bitmap_quote:w 1956 , 1997
. 1704 , 1724 , 1725 , 1749 ,	__graphics_backend_include_-
1750 , 1814 , 1851 , 1852 , 1878 , 1879 ,	eps:n 1688 , 1769 , 1827 , 1982
1907 , 1920 , 1921 , 1960 , 1961 , 1971	__graphics_backend_include_-
\l_graphics_pagebox_tl	jpg:n 1762 , 1827 , 1997
. 51 , 1705 , 1723 ,	__graphics_backend_include_-
1751 , 1752 , 1815 , 1849 , 1850 , 1880 ,	pdf:n 1762 , 1801 , 1827 , 1956 , 1982
1882 , 1908 , 1929 , 1930 , 1962 , 1972	__graphics_backend_include_pdf_-
\graphics_read_bb:n 1687 , 1810 , 1968	quote:w 1959 , 1964
\l_graphics_urx_dim	__graphics_backend_include_-
. 1695 , 1756 , 1834 , 1886 , 1951 , 1993	png:n 1762 , 1827 , 1997
\l_graphics_ury_dim 1696 , 1757 ,	\l_graphics_backend_name_str . 1769
1835 , 1887 , 1952 , 1994 , 2002 , 2003	\l_graphics_graphics_attr_tl . . .
graphics internal commands: 1701 , 1706 ,
\l_graphics_backend_dir_str . 1769	1713 , 1721 , 1731 , 1758 , 1760 , 1765
\l_graphics_backend_ext_str . 1769	\l_graphics_internal_box
__graphics_backend_getbb_auxi:n 1754 , 1756 , 1757 , 1950 , 1951 , 1952
. 1702	\g_graphics_track_int
__graphics_backend_getbb_- 1826 , 1872 , 1873
auxi:nN 1905	group commands:
__graphics_backend_getbb_-	\group_begin: 143 , 171 , 190
auxii:n 1702	\group_end: 156 , 179
__graphics_backend_getbb_-	\group_insert_after:N 615 , 634 , 645 ,
auxii:nnN 1905	964 , 977 , 992 , 1019 , 1044 , 3026 , 3061
__graphics_backend_getbb_-	
auxiii:nNnn 1905	H
__graphics_backend_getbb_-	hbox commands:
auxiv:nnNnn 1905	\hbox:n 2130 , 2133 ,
__graphics_backend_getbb_-	2205 , 2211 , 2364 , 2371 , 2906 , 2917
auxv:nNnn 1905	\hbox_overlap_right:n 223 ,
__graphics_backend_getbb_-	255 , 271 , 312 , 328 , 356 , 440 , 1292 , 1487
auxvi:nNnn 1946 , 1948	\hbox_set:Nn 1754 , 1950 , 2197 , 2229
__graphics_backend_getbb_eps:n .	\hbox_set:Nw 2180
. 1687 , 1769 , 1810 , 1968	\hbox_set_end: 2195
__graphics_backend_getbb_eps:nm	\hbox_unpack:N 2316
. 1769	
__graphics_backend_getbb_eps:nn	I
. 1780 , 1787	int commands:
__graphics_backend_getbb_jpg:n .	\int_compare:nNnTF
. 1702 , 1810 , 1905 , 1969	508 , 542 , 640 , 945 , 987 , 1724 , 1749 ,
__graphics_backend_getbb_-	1851 , 1878 , 1920 , 1960 , 2288 , 2389 ,
pagebox:w 1905 , 1962	2679 , 2707 , 2843 , 2850 , 2866 , 3067
__graphics_backend_getbb_pdf:n .	\int_const:Nn 149 , 155 ,
. 1702 , 1795 , 1810 , 1905 , 1976	515 , 568 , 1758 , 1873 , 2028 , 2557 , 2745
__graphics_backend_getbb_png:n .	\int_eval:n 549 , 559 , 588 , 599 ,
. 1702 , 1810 , 1905 , 1969	703 , 712 , 725 , 727 , 731 , 744 , 2657 ,
	2682 , 2689 , 2702 , 2935 , 2943 , 2948

`\int_gincr:N` 197, 363,
 514, 1554, 1599, 1872, 2027, 2096,
 2140, 2214, 2744, 2787, 2823, 2845
`\int_gset:Nn` 172, 191, 2277
`\int_gset_eq:NN` 180, 2141, 2215, 2824
`\int_if_exist:NTF` 1862
`\int_if_odd:nTF` 2200
`\int_new:N` 163, 164,
 410, 505, 511, 1580, 1826, 2023,
 2121, 2152, 2154, 2740, 2805, 2836
`\int_set:Nn` 532
`\int_set_eq:NN` ... 168, 187, 538, 2289
`\int_step_function:nnnN` 729
`\int_use:N` 365,
 396, 523, 533, 677, 813, 858, 931,
 1557, 1563, 1570, 1602, 1610, 1725,
 1750, 1765, 1852, 1865, 1877, 1879,
 1961, 2034, 2099, 2112, 2116, 2144,
 2151, 2219, 2320, 2568, 2578, 2751,
 2789, 2794, 2827, 2835, 2853, 2869
`\int_value:w`
 2436, 2471, 2628, 2639, 2657
`\int_zero:N` ... 1704, 1814, 1907, 1971

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, 659
`__kernel_backend_literal:n` . 46,
 52, 55, 62, 66, 73, 76, 78, 134, 137,
 139, 141, 161, 337, 350, 520, 546,
 556, 611, 618, 644, 650, 673, 809,
 991, 997, 999, 1018, 1043, 1110,
 1116, 1411, 1418, 1424, 1484, 1489,
 1690, 1829, 1864, 1874, 1988, 1999,
 2734, 2873, 2935, 2939, 2944, 2949
`__kernel_backend_literal_page:n`
 92, 136, 2728, 2730, 2954, 2956
`__kernel_backend_literal_pdf:n` .
 .. 81, 133, 263, 320, 1301, 3076, 3091
`__kernel_backend_literal_-`
`postscript:n`
 51, 67, 68, 72, 209, 210, 212,
 213, 221, 233, 248, 1106, 2391, 2403
`__kernel_backend_literal_svg:n` .
 160, 167, 178, 186,
 196, 364, 366, 383, 1495, 1668, 1679
`__kernel_backend_matrix:n`
 120, 285, 306, 1401

`__kernel_backend_postscript:n` ..
 54, 613,
 1022, 1024, 1026, 1030, 2017, 2073,
 2088, 2130, 2136, 2173, 2205, 2212,
 2216, 2230, 2258, 2302, 2309, 2315,
 2323, 2330, 2364, 2371, 2989, 2998
`__kernel_backend_scope:n`
 165, 393, 398, 1084, 1500, 3105
`__kernel_backend_scope_begin:` ..
 75, 102, 138,
 165, 207, 231, 246, 262, 279, 305,
 319, 336, 349, 1307, 1479, 1499, 1666
`__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, 1308, 1491, 1502, 1680
`\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, 566, 3005
`__kernel_color_backend_stack_-`
`pop:n` 542, 580, 637, 3033
`__kernel_color_backend_stack_-`
`push:nn`
 .. 542, 580, 633, 962, 975, 3024, 3059
`__kernel_dependency_version_-`
`check:Nn` 1
`__kernel_dependency_version_-`
`check:nn` 27, 29
`__kernel_kern:n`
 2135, 2137, 2363, 2367,
 2370, 2374, 2905, 2913, 2916, 2932
`\c_kernel_sys_dvipdfmx_version_-`
`int` 143, 508, 542,
 640, 945, 987, 2843, 2850, 2866, 3067

L

`\landscape` 2798, 2800

M

math commands:

`\c_math_toggle_token` 2183, 2193

`\MessageBreak` 40

mode commands:

`\mode_if_horizontal:TF` ... 2279, 2286

`\mode_if_math:TF` 2177

O

`\oddsidemargin` 2201

opacity internal commands:

`__opacity_backend:nn` ... 2992, 3098

_opacity_backend_fill:n	2992, 3034, 3098
_opacity_backend_fill_stroke:nn	3036, 3042, 3046, 3064, 3078
\l_opacity_backend_fill_tl	3010, 3019, 3043, 3051, 3071, 3083
_opacity_backend_fillstroke:nn	3034
_opacity_backend_reset:	3012, 3061
_opacity_backend_select:n	2982, 3012, 3067, 3098
_opacity_backend_select_aux:n	2982, 3012, 3049, 3069, 3081
\c_opacity_backend_stack_int	3003, 3024, 3033, 3059
_opacity_backend_stroke:n	2992, 3034, 3098
\l_opacity_backend_stroke_tl	3010, 3020, 3038, 3052, 3072, 3084
P	
pdf commands:	
\pdf_object_if_exist:nTF	869
\pdf_object_new:nn	871
\pdf_object_ref:n	884
\pdf_object_ref_last:	850, 859, 923, 932
\pdf_object_unnamed_write:nn	838, 865, 889, 896
\pdf_object_write:nn	872
pdf internal commands:	
_pdf_backend:n	2733, 2737, 2739, 2765, 2770, 2779, 2825, 2847, 2863, 2876, 2908, 2909, 2919
_pdf_backend_annotation:nnnn	2122, 2421, 2806
_pdf_backend_annotation_	aux:nnnn 2124, 2127, 2806
\g_pdf_backend_annotation_int	2121, 2141, 2151, 2805, 2824, 2835
_pdf_backend_annotation_last:	2150, 2434, 2834
_pdf_backend_bdc:nn	2415, 2727, 2953, 2975
_pdf_backend_catalog_gput:nn	2019, 2527, 2736, 2959
_pdf_backend_compress_objects:n	2387, 2648, 2934, 2969
_pdf_backend_compresslevel:n	2387, 2648, 2934, 2969
\l_pdf_backend_content_box	2119, 2180, 2204, 2207, 2209, 2238, 2249
_pdf_backend_destination:nn	2328, 2490, 2874
_pdf_backend_destination:nnnn	2328, 2490, 2874
_pdf_backend_destination_	aux:nnnn 2328, 2874
_pdf_backend_emc:	2415, 2727, 2953, 2975
_pdf_backend_info_gput:nn	2019, 2527, 2736, 2959
_pdf_backend_link:nw	2161
_pdf_backend_link_aux:nw	2161
_pdf_backend_link_begin:n	2837
_pdf_backend_link_begin:nnnw	2445
_pdf_backend_link_begin:nw	2162, 2164, 2165
_pdf_backend_link_begin_aux:nw	2168, 2170
_pdf_backend_link_begin_	goto:nnw 2161, 2445, 2837
_pdf_backend_link_begin_	user:nnw 2161, 2445, 2837
\g_pdf_backend_link_bool	2156, 2167, 2172, 2187, 2225
\g_pdf_backend_link_dict_tl	2153, 2175, 2220
_pdf_backend_link_end:	2161, 2445, 2837
_pdf_backend_link_end_aux:	2161
\g_pdf_backend_link_int	2152, 2215, 2219, 2320, 2836, 2845, 2853, 2869
_pdf_backend_link_last:	2319, 2469, 2864
_pdf_backend_link_margin:n	2321, 2480, 2872
\g_pdf_backend_link_math_bool	2155, 2178, 2179, 2182, 2192
_pdf_backend_link_minima:	2161
_pdf_backend_link_outabox:n	2161
\g_pdf_backend_link_sf_int	2154, 2277, 2288, 2289
_pdf_backend_link_sf_restore:	2161
_pdf_backend_link_sf_save:	2161
\l_pdf_backend_model_box	2120, 2197, 2229, 2237, 2248, 2263, 2265
_pdf_backend_objcompresslevel:n	2648
\g_pdf_backend_object_int	2023, 2027, 2030, 2096, 2099, 2112, 2116, 2140, 2141, 2144, 2214, 2215, 2740, 2744, 2747, 2787, 2789, 2794, 2823, 2824, 2827
_pdf_backend_object_last:	2115, 2626, 2793, 2961

__pdf_backend_object_new:nn . . .	pdf.bordertracking.endpage	3194
. 2025, 2548, 2742, 2961	pdf.breaklink	3332
__pdf_backend_object_now:nn . . .	pdf.breaklink.write	3332
. 2094, 2600, 2785, 2961	pdf.brokenlink.dict	3194
\g_pdf_backend_object_prop	pdf.brokenlink.rect	3194
. 2023, 2031, 2042, 2052,	pdf.brokenlink.skip	3194
2547, 2565, 2581, 2740, 2748, 2755	pdf.count	3332
__pdf_backend_object_ref:n 2025,	pdf.currentrect	3332
2039, 2053, 2548, 2742, 2761, 2961	pdf.cvs	3116
__pdf_backend_object_write:nn . .	pdf.dest.anchor	3159
. 2035, 2569, 2752, 2961	pdf.dest.point	3159
__pdf_backend_object_write:nnn 2752	pdf.dest.x	3159
__pdf_backend_object_write_-	pdf.dest.y	3159
array:nn 2035, 2752	pdf.dest2device	3159
__pdf_backend_object_write_-	pdf.dev.x	3159
dict:nn 2035, 2752	pdf.dev.y	3159
__pdf_backend_object_write_-	pdf.dvi.pt	3116
fstream:nn 2035, 2752	pdf.globaldict	3113
__pdf_backend_object_write_-	pdf.leftboundary	3194
fstream:nnn 2069, 2071	pdf.link.dict	2161
__pdf_backend_object_write_-	pdf.linkdp.pad 2161, 3120	
stream:nn 2035, 2752	pdf.linkht.pad 2161, 3120	
__pdf_backend_object_write_-	pdf.linkmargin	3120
stream:nnn 2035	pdf.llx	2161, 3123
__pdf_backend_object_write_-	pdf.lly	2161, 3123
stream:nnnn 2752	pdf.originx	3194
__pdf_backend_pageobject_ref:n .	pdf.originy	3194
. 2117, 2637, 2795, 2961	pdf.outerbox 2161, 3436	
__pdf_backend_pdfmark:n	pdf.pdfmark	3436
2016, 2020, 2022, 2037, 2058, 2063,	pdf.pdfmark.dict	3436
2097, 2142, 2331, 2375, 2416, 2418	pdf.pdfmark.good	3436
__pdf_backend_version_major: . . .	pdf.pt.dvi	3116
. . . 2413, 2704, 2943, 2944, 2951, 2973	pdf.rect	3123
__pdf_backend_version_major_-	pdf.rect.ht	3116
gset:n 2411, 2676, 2941, 2971	pdf.rightboundary	3194
__pdf_backend_version_minor: . . .	pdf.save.linkll	3123
. . . 2413, 2704, 2948, 2949, 2951, 2973	pdf.save.linkur	3123
__pdf_backend_version_minor_-	pdf.save.ll	3123
gset:n 2411, 2676, 2941, 2971	pdf.save.ur	3123
\l_pdf_breaklink_pdfmark_tl . . .	pdf.tmpa	3159
. 2157, 2222, 2314	pdf.tmpb	3159
__pdf_breaklink_postscript:n . . .	pdf.tmpc	3159
. 2159, 2206, 2208, 2315	pdf.tmpd	3159
__pdf_breaklink_usebox:N	pdf.urx	3123
. 2160, 2207, 2316	pdf.ury	2161, 3123
__pdf_exp_not_i:nn . 2569, 2615, 2620	pdfmanagement commands:	
__pdf_exp_not_ii:nn 2569, 2616, 2621	\pdfmanagement_add:nnn 852,	
\l_pdf_internal_box 2014	856, 925, 929, 3003, 3007, 3021,	
\g_pdf_landscape_bool . . . 2797, 2808	3028, 3053, 3056, 3073, 3085, 3088	
pdf.baselineskip 2161, 3436	prg commands:	
pdf.bordertracking 3194	\prg_replicate:nn	
pdf.bordertracking.begin 3194 176, 701, 722, 732, 902	
pdf.bordertracking.continue 3194	prop commands:	
pdf.bordertracking.end 3194	\prop_gput:Nnn 2031, 2565, 2748	

<code>\prop_item:Nn</code> .	2042, 2052, 2581, 2755	<code>\tex luatexversion:D</code>	2679, 2707
<code>\prop_new:N</code>	2024, 2547, 2741	<code>\tex pdfannot:D</code>	2427
<code>\ProvidesExplFile</code>	2	<code>\tex pdfcatalog:D</code>	2533
Q			
quark commands:		<code>\tex pdfcolorstack:D</code>	586, 597
<code>\q_stop</code>	144, 152	<code>\tex pdfcolorstackinit:D</code>	574
quark internal commands:		<code>\tex pdfcompresslevel:D</code>	2655
<code>\s__color_stop</code>		<code>\tex pdfdest:D</code>	2496, 2519
.	463, 466, 477, 480, 712, 713,	<code>\tex pdfendlink:D</code>	2466
.	717, 721, 734, 737, 741, 745, 759,	<code>\tex pdfextension:D</code>	
.	903, 936, 940, 1047, 1049, 1070, 1072	84, 95, 105, 114, 123,
<code>\s__graphics_stop</code>	583, 594, 2424, 2452, 2463, 2493,
.	1959, 1964, 2004, 2008	2516, 2530, 2540, 2551, 2572, 2603
S			
scan commands:		<code>\tex pdffeedback:D</code>	
<code>\scan_stop:</code>	571, 2438, 2473, 2560, 2630, 2641
.	105, 114, 599, 2463, 2488, 2511,	<code>\tex pdfinfo:D</code>	2543
.	2525, 2657, 2674, 2682, 2689, 2702	<code>\tex pdflastannot:D</code>	2441
separation	3110	<code>\tex pdflastlink:D</code>	2476
skip commands:		<code>\tex pdflastobj:D</code>	2563, 2633
<code>\skip_horizontal:n</code>	225, 273, 330	<code>\tex pdflastximage:D</code>	1755, 1759
str commands:		<code>\tex pdflinkmargin:D</code>	2486
<code>\c_hash_str</code>	396, 1563, 1570, 1610	<code>\tex pdfliteral:D</code>	87, 98
<code>\c_percent_str</code>	1090, 1091, 1092	<code>\tex pdfmajorversion:D</code>	
<code>\str_case:nn</code>	908, 2101, 2608	2686, 2688, 2712, 2713
<code>\str_case:nnTF</code>	2335, 2499, 2881	<code>\tex pdfminorversion:D</code> . . .	2700, 2724
<code>\str_case:e:nn</code>	2041, 2580	<code>\tex pdfobj:D</code>	2554, 2575, 2606
<code>\str_convert_pdfname:n</code>	680, 849	<code>\tex pdfobjcompresslevel:D</code> . . .	2672
<code>\str_if_eq:nnTF</code>		<code>\tex pdfpageref:D</code>	2644
.	482, 485, 488, 491, 3048, 3080	<code>\tex pdfrefximage:D</code>	1755, 1764
<code>\str_new:N</code>	1771, 1772, 1773	<code>\tex pdfrestore:D</code>	117
<code>\str_tail:N</code>	1782, 1803	<code>\tex pdfsave:D</code>	108
sys commands:		<code>\tex pdfsetmatrix:D</code>	126
<code>\sys_get_shell:nnNTF</code>	145	<code>\tex pdfstartlink:D</code>	2455
<code>\sys_if_shell:TF</code>	1769	<code>\tex pdfvariable:D</code>	2483,
<code>\sys_shell_now:n</code>	1791	2652, 2669, 2681, 2697, 2708, 2721
sys internal commands:		<code>\tex pdfximage:D</code>	1736
<code>\l_sys_internal_tl</code>	147, 151	<code>\tex spacefactor:D</code>	2280, 2289
<code>__sys_tmp:w</code>	144, 151	<code>\tex special:D</code>	46
T			
TeX and L ^A T _E X 2 _ε commands:		<code>\tex the:D</code>	1759, 2708, 2713, 2719
<code>\@cclv</code>	2298, 2300, 2308	<code>\tex vss:D</code>	2365, 2372, 2911, 2930
<code>\@makecol@hook</code>	2292	<code>\tex XeTeXpdffile:D</code>	1916, 1958
<code>\current@color</code>	14, 453, 457, 463, 477	<code>\tex XeTeXpicfile:D</code>	1909
<code>\special</code>	2	TeXcolorseparation	3110
tex commands:		<code>\textwidth</code>	2264
<code>\tex baselineskip:D</code>	2269	tl commands:	
<code>\tex endinput:D</code>	44	<code>\c_space_tl</code>	287, 292, 295, 524,
<code>\tex global:D</code>	768, 976, 1539, 1692, 1693, 1694,
.	2650, 2667, 2681, 2688, 2695	1695, 1831, 1832, 1833, 1834, 1879,
<code>\tex immediate:D</code>	1882, 1884, 1885, 1886, 1887, 1959,
.	1736, 2572, 2575, 2603, 2606	1961, 1990, 1991, 1992, 1993, 2220,
		2443, 2478, 2635, 2646, 2827, 2854
		<code>\tl_clear:N</code>	1705, 1713, 1719,
		1815, 1821, 1908, 1914, 1972, 1978
		<code>\tl_gclear:N</code>	1577, 1613
		<code>\tl_gset:Nn</code>	1536, 2175

<code>\tl_if_blank:nTF</code>	<code>\tl_use:N</code> 800, 877
525, 576, 716, 733, 740, 758, 842, 939	
<code>\tl_if_empty:N</code> 1539, 1708, 1743,	U
1751, 1849, 1853, 1880, 1895, 1929	use commands:
<code>\tl_if_empty:nTF</code> 1633	<code>\use:N</code> 43, 2051, 2111, 2760, 2788
<code>\tl_if_empty_p:N</code> 1739, 1892	<code>\use:n</code> 61, 457, 493, 518, 663,
<code>\tl_if_head_is_space:nTF</code> 453	854, 927, 1051, 1061, 1074, 1247,
<code>\tl_new:N</code> 621,	1371, 1436, 1448, 1460, 1618, 1936
622, 1543, 1701, 2153, 2157, 3010, 3011	<code>\use_none:n</code> 1633, 1635, 2292
<code>\tl_put_left:Nn</code> 2802	V
<code>\tl_put_right:Nn</code> 2296, 2800	<code>\value</code> 2200
<code>\tl_set:Nn</code> 455, 467, 483, 486, 489,	vbox commands:
493, 496, 631, 632, 961, 974, 1706,	<code>\vbox:n</code> 2812
1721, 1794, 2158, 2314, 3019, 3020,	<code>\vbox_set:Nn</code> 2300
3051, 3052, 3071, 3072, 3083, 3084	<code>\vbox_to_zero:n</code> 2361, 2368, 2903, 2914
<code>\tl_to_str:n</code> 2029,	<code>\vbox_unpack_drop:N</code> 2308
2034, 2558, 2568, 2579, 2746, 2751	