

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}{2021-10-12}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2021-10-12}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2021-10-12}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2021-10-12}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2021-10-12}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2021-10-12}{ }
25   {L3 backend support: XeTeX}
26 </xetex>
```

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

```
27 \cs_if_exist:NTF \_kernel_dependency_version_check:nn
28 {
29   \_kernel_dependency_version_check:nn {2021-02-18}
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`.)

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

```

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

```

(End definition for `__kernel_backend_first_shipout:n`.)

1.1 dvips backend

```

60 ⟨*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.

```

61 \cs_new_protected:Npn __kernel_backend_literal_postscript:n #1
62 { __kernel_backend_literal:n { ps:: #1 } }
63 \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 `"`).

`_kernel_backend_postscript:x`

```

64 \cs_new_protected:Npn \_kernel_backend_postscript:n #1
65   { \_kernel_backend_literal:n { ps: SDict ~ begin ~ #1 ~ end } }
66 \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.

```

67 \bool_if:NT \g__kernel_backend_header_bool
68   {
69     \_kernel_backend_first_shipout:n
70     { \_kernel_backend_literal:n { header = l3backend-dvips.pro } }
71   }

```

`_kernel_backend_align_begin:`

`_kernel_backend_align_end:`

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.

```

72 \cs_new_protected:Npn \_kernel_backend_align_begin:
73   {
74     \_kernel_backend_literal:n { ps::[begin] }
75     \_kernel_backend_literal_postscript:n { currentpoint }
76     \_kernel_backend_literal_postscript:n { currentpoint-translate }
77   }
78 \cs_new_protected:Npn \_kernel_backend_align_end:
79   {
80     \_kernel_backend_literal_postscript:n { neg-exch-neg-exch-translate }
81     \_kernel_backend_literal:n { ps::[end] }
82   }

```

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

`_kernel_backend_scope_begin:`

`_kernel_backend_scope_end:`

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.

```

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

```

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

```

87 </dvips>

```

1.2 LuaTeX and pdfTeX backends

88 $\langle *luatex | pdftex \rangle$

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
_kernel_backend_literal_pdf:x

This is equivalent to $\backslash special\{pdf:\}$ 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).

```
89 \cs_new_protected:Npn \_kernel_backend_literal_pdf:n #1
90 {
91   \langle *luatex \rangle
92   \tex_pdfextension:D literal
93   \langle /luatex \rangle
94   \langle *pdftex \rangle
95   \tex_pdfliteral:D
96   \langle /pdftex \rangle
97   { \exp_not:n {#1} }
98 }
99 \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.

```
100 \cs_new_protected:Npn \_kernel_backend_literal_page:n #1
101 {
102   \langle *luatex \rangle
103   \tex_pdfextension:D literal ~
104   \langle /luatex \rangle
105   \langle *pdftex \rangle
106   \tex_pdfliteral:D
107   \langle /pdftex \rangle
108   page { \exp_not:n {#1} }
109 }
```

(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:

```
110 \cs_new_protected:Npn \_kernel_backend_scope_begin:
111 {
112   \langle *luatex \rangle
113   \tex_pdfextension:D save \scan_stop:
114   \langle /luatex \rangle
115   \langle *pdftex \rangle
116   \tex_pdfsave:D
117   \langle /pdftex \rangle
118 }
119 \cs_new_protected:Npn \_kernel_backend_scope_end:
120 {
121   \langle *luatex \rangle
122   \tex_pdfextension:D restore \scan_stop:
123   \langle /luatex \rangle
124   \langle *pdftex \rangle
125   \tex_pdfrestore:D
```

```

126 </pdfTeX>
127 }

```

(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.

```

128 \cs_new_protected:Npn \_kernel_backend_matrix:n #1
129 {
130   <*luatex>
131     \tex_pdfextension:D setmatrix
132   </luatex>
133   <*pdfTeX>
134     \tex_pdfsetmatrix:D
135   </pdfTeX>
136     { \exp_not:n {#1} }
137   }
138 \cs_generate_variant:Nn \_kernel_backend_matrix:n { x }

```

(End definition for `_kernel_backend_matrix:n.`)

```

139 </luatex | pdfTeX>

```

1.3 dvipdfmx backend

```

140 <*dvipdfmx | xetex>

```

The dvipdfmx shares code with the PDF mode one (using the common section to this file) but also with XeTeX. 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 XeTeX 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

```

```

141 \cs_new_protected:Npn \_kernel_backend_literal_pdf:n #1
142 { \_kernel_backend_literal:n { pdf:literal~ #1 } }
143 \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!

```

144 \cs_new_protected:Npn \_kernel_backend_literal_page:n #1
145 { \_kernel_backend_literal:n { pdf:literal~direct~ #1 } }

```

(End definition for `_kernel_backend_literal_page:n.`)

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

```

146 \cs_new_protected:Npn \_kernel_backend_scope_begin:
147 { \_kernel_backend_literal:n { x:gsave } }
148 \cs_new_protected:Npn \_kernel_backend_scope_end:
149 { \_kernel_backend_literal:n { x:grestore } }

```

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

```

150 <@@=sys>

```

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

```

151 \group_begin:
152 \cs_set:Npn \__sys_tmp:w #1 Version ~ #2 ~ #3 \q_stop {#2}
153 \sys_get_shell:nnNTF { extractbb~--version }
154 { \char_set_catcode_space:n { '\ } }
155 \l__sys_internal_tl
156 {
157   \int_const:Nn \c__kernel_sys_dvipdfmx_version_int
158   {
159     \exp_after:wN \__sys_tmp:w \l__sys_internal_tl
160     \q_stop
161   }
162 }
163 { \int_const:Nn \c__kernel_sys_dvipdfmx_version_int { 0 } }
164 \group_end:

```

(End definition for `\c__kernel_sys_dvipdfmx_version_int`.)

```

165 <@@=
166 </dvipdfmx | xetex>

```

1.4 dvisvgm backend

```

167 <*dvisvgm>

```

`__kernel_backend_literal_svg:n` 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.

`__kernel_backend_literal_svg:x`

```

168 \cs_new_protected:Npn \__kernel_backend_literal_svg:n #1
169 { \__kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }
170 \cs_generate_variant:Nn \__kernel_backend_literal_svg:n { x }

```

(End definition for `__kernel_backend_literal_svg:n`.)

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

`\l__kernel_backend_scope_int`

```

171 \int_new:N \g__kernel_backend_scope_int
172 \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:` 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.

`__kernel_backend_scope_end:`

`__kernel_backend_scope_begin:n`

`__kernel_backend_scope_begin:x`

```

173 \cs_new_protected:Npn \__kernel_backend_scope_begin:
174 {
175   \__kernel_backend_literal_svg:n { <g> }
176   \int_set_eq:NN
177   \l__kernel_backend_scope_int
178   \g__kernel_backend_scope_int
179   \group_begin:
180   \int_gset:Nn \g__kernel_backend_scope_int { 1 }

```

```

181 }
182 \cs_new_protected:Npn \__kernel_backend_scope_end:
183 {
184   \prg_replicate:nn
185     { \g__kernel_backend_scope_int }
186     { \__kernel_backend_literal_svg:n { </g> } }
187   \group_end:
188   \int_gset_eq:NN
189     \g__kernel_backend_scope_int
190     \l__kernel_backend_scope_int
191 }
192 \cs_new_protected:Npn \__kernel_backend_scope_begin:n #1
193 {
194   \__kernel_backend_literal_svg:n { <g ~ #1 > }
195   \int_set_eq:NN
196     \l__kernel_backend_scope_int
197     \g__kernel_backend_scope_int
198   \group_begin:
199     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
200 }
201 \cs_generate_variant:Nn \__kernel_backend_scope_begin:n { x }
202 \cs_new_protected:Npn \__kernel_backend_scope:n #1
203 {
204   \__kernel_backend_literal_svg:n { <g ~ #1 > }
205   \int_gincr:N \g__kernel_backend_scope_int
206 }
207 \cs_generate_variant:Nn \__kernel_backend_scope:n { x }

```

(End definition for __kernel_backend_scope_begin: and others.)

```

208 </dvisvgm>
209 </package>

```

2 l3backend-box Implementation

```

210 <*package>
211 <@@=box>

```

2.1 dvips backend

```

212 <*dvips>

```

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

```

213 \cs_new_protected:Npn \__box_backend_clip:N #1
214 {
215   \__kernel_backend_scope_begin:
216   \__kernel_backend_align_begin:
217   \__kernel_backend_literal_postscript:n { matrix~currentmatrix }
218   \__kernel_backend_literal_postscript:n
219     { Resolution~72~div~VResolution~72~div~scale }

```

```

220 \__kernel_backend_literal_postscript:n { DVImag-dup~scale }
221 \__kernel_backend_literal_postscript:x
222 {
223   0 ~
224   \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
225   \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
226   \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
227   rectclip
228 }
229 \__kernel_backend_literal_postscript:n { setmatrix }
230 \__kernel_backend_align_end:
231 \hbox_overlap_right:n { \box_use:N #1 }
232 \__kernel_backend_scope_end:
233 \skip_horizontal:n { \box_wd:N #1 }
234 }

```

(End definition for __box_backend_clip:N.)

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

```

235 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
236 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
237 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
238 {
239   \__kernel_backend_scope_begin:
240   \__kernel_backend_align_begin:
241   \__kernel_backend_literal_postscript:x
242   {
243     \fp_compare:nNnTF {#2} = \c_zero_fp
244     { 0 }
245     { \fp_eval:n { round ( -(#2) , 5 ) } } ~
246     rotate
247   }
248   \__kernel_backend_align_end:
249   \box_use:N #1
250   \__kernel_backend_scope_end:
251 }

```

(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.

```

252 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
253 {
254   \__kernel_backend_scope_begin:
255   \__kernel_backend_align_begin:
256   \__kernel_backend_literal_postscript:x
257   {
258     \fp_eval:n { round ( #2 , 5 ) } ~
259     \fp_eval:n { round ( #3 , 5 ) } ~
260     scale
261   }
262   \__kernel_backend_align_end:
263   \hbox_overlap_right:n { \box_use:N #1 }

```



```

264     \__kernel_backend_scope_end:
265 }
(End definition for \__box_backend_scale:Nnn.)
266 </dvips>

```

2.2 LuaTeX and pdfTeX backends

```

267 <*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.

```

268 \cs_new_protected:Npn \__box_backend_clip:N #1
269 {
270     \__kernel_backend_scope_begin:
271     \__kernel_backend_literal_pdf:x
272     {
273         0~
274         \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
275         \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
276         \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
277         re~W~n
278     }
279     \hbox_overlap_right:n { \box_use:N #1 }
280     \__kernel_backend_scope_end:
281     \skip_horizontal:n { \box_wd:N #1 }
282 }

```

(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.

```

283 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
284 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
285 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
286 {
287     \__kernel_backend_scope_begin:
288     \box_set_wd:Nn #1 { Opt }
289     \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
290     \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
291     { \fp_zero:N \l__box_backend_cos_fp }
292     \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
293     \__kernel_backend_matrix:x
294     {
295         \fp_use:N \l__box_backend_cos_fp \c_space_tl
296         \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp

```

```

297         { 0~0 }
298         {
299             \fp_use:N \l__box_backend_sin_fp
300             \c_space_tl
301             \fp_eval:n { -\l__box_backend_sin_fp }
302         }
303         \c_space_tl
304         \fp_use:N \l__box_backend_cos_fp
305     }
306     \box_use:N #1
307     \__kernel_backend_scope_end:
308 }
309 \fp_new:N \l__box_backend_cos_fp
310 \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.

```

311 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
312 {
313     \__kernel_backend_scope_begin:
314     \__kernel_backend_matrix:x
315     {
316         \fp_eval:n { round ( #2 , 5 ) } ~
317         0~0~
318         \fp_eval:n { round ( #3 , 5 ) }
319     }
320     \hbox_overlap_right:n { \box_use:N #1 }
321     \__kernel_backend_scope_end:
322 }

```

(End definition for __box_backend_scale:Nnn.)

323 </luatex | pdftex>

2.3 dvipdfmx/X_YTeX backend

324 <*dvipdfmx | xetex>

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

```

325 \cs_new_protected:Npn \__box_backend_clip:N #1
326 {
327     \__kernel_backend_scope_begin:
328     \__kernel_backend_literal_pdf:x
329     {
330         0~
331         \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
332         \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
333         \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
334         re~W~n
335     }
336     \hbox_overlap_right:n { \box_use:N #1 }
337     \__kernel_backend_scope_end:
338     \skip_horizontal:n { \box_wd:N #1 }
339 }

```

(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.

`_box_backend_rotate_aux:Nn`

```

340 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
341 { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
342 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
343 {
344   \__kernel_backend_scope_begin:
345   \__kernel_backend_literal:x
346   {
347     x:rotate~
348     \fp_compare:nNnTF {#2} = \c_zero_fp
349     { 0 }
350     { \fp_eval:n { round ( #2 , 5 ) } }
351   }
352   \box_use:N #1
353   \__kernel_backend_scope_end:
354 }

```

(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.

```

355 \cs_new_protected:Npn \_box_backend_scale:Nnn #1#2#3
356 {
357   \__kernel_backend_scope_begin:
358   \__kernel_backend_literal:x
359   {
360     x:scale~
361     \fp_eval:n { round ( #2 , 5 ) } ~
362     \fp_eval:n { round ( #3 , 5 ) }
363   }
364   \hbox_overlap_right:n { \box_use:N #1 }
365   \__kernel_backend_scope_end:
366 }

```

(End definition for `_box_backend_scale:Nnn`.)

367 `</dvipdfmx | xetex>`

2.4 dvisvgm backend

368 `<*dvisvgm>`

`_box_backend_clip:N` 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!

```

369 \cs_new_protected:Npn \__box_backend_clip:N #1
370 {
371   \int_gincr:N \g__box_clip_path_int
372   \__kernel_backend_literal_svg:x
373   { < clipPath~id = " l3cp \int_use:N \g__box_clip_path_int " > }
374   \__kernel_backend_literal_svg:x
375   {
376     <
377     path ~ d =
378     "
379       M ~ 0 ~
380       \dim_to_decimal:n { -\box_dp:N #1 } ~
381       L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
382       \dim_to_decimal:n { -\box_dp:N #1 } ~
383       L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
384       \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
385       L ~ 0 ~
386       \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
387       Z
388     "
389   />
390   }
391   \__kernel_backend_literal_svg:n
392   { < /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 it lines up with the current point, as does the \TeX box.

```

393 \__kernel_backend_scope_begin:n
394 {
395   transform =
396   "
397     translate ( { ?x } , { ?y } ) ~
398     scale ( 1 , -1 )
399   "
400 }
401 \__kernel_backend_scope:x
402 {
403   clip-path =
404   "url ( \c_hash_str l3cp \int_use:N \g__box_clip_path_int ) "
405 }
406 \__kernel_backend_scope:n
407 {
408   transform =
409   "
410     scale ( -1 , 1 ) ~
411     translate ( { ?x } , { ?y } ) ~
412     scale ( -1 , -1 )
413   "
414 }

```

```

415     \box_use:N #1
416     \__kernel_backend_scope_end:
417   }
418   \int_new:N \g__box_clip_path_int

```

(End definition for __box_backend_clip:N and \g__box_clip_path_int.)

__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.

```

419   \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
420   {
421     \__kernel_backend_scope_begin:x
422     {
423       transform =
424       "
425         rotate
426         ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
427       "
428     }
429     \box_use:N #1
430     \__kernel_backend_scope_end:
431   }

```

(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).

```

432   \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
433   {
434     \__kernel_backend_scope_begin:x
435     {
436       transform =
437       "
438         translate ( { ?x } , { ?y } ) ~
439         scale
440         (
441           \fp_eval:n { round ( -#2 , 5 ) } ,
442           \fp_eval:n { round ( -#3 , 5 ) }
443         ) ~
444         translate ( { ?x } , { ?y } ) ~
445         scale ( -1 )
446       "
447     }
448     \hbox_overlap_right:n { \box_use:N #1 }
449     \__kernel_backend_scope_end:
450   }

```

(End definition for __box_backend_scale:Nnn.)

```

451   </dvisvgm>
452   </package>

```

3 l3backend-color Implementation

```

453 <*package>
454 <@@=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

```

455 <*dvisvgm | dvipdfmx | dvips | xetex>

```

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). The x-type expansion is there to cover the case where xcolor is in use.

```

456 \cs_new_protected:Npn \__color_backend_pickup:N #1 { }
457 \cs_if_exist:cT { ver@color.sty }
458 {
459   \cs_set_protected:Npn \__color_backend_pickup:N #1
460   {
461     \exp_args:NW \tl_if_head_is_space:nTF \current@color
462     {
463       \tl_set:Nx #1
464       {
465         { \exp_after:wN \use:n \current@color }
466         { 1 }
467       }
468     }
469     {
470       \exp_last_unbraced:Nx \__color_backend_pickup:w
471       { \current@color } \s__color_stop #1
472     }
473   }
474   \cs_new_protected:Npn \__color_backend_pickup:w #1 ~ #2 \s__color_stop #3
475   { \tl_set:Nn #3 { {#1} {#2} } }
476 }

```

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

```

477 </dvisvgm | dvipdfmx | dvips | xetex>

```

3.1.2 Lua_T_EX and pdf_T_EX

```

478 <*luatex | pdftex>

```

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

```

479 \cs_new_protected:Npn \__color_backend_pickup:N #1 { }
480 \cs_if_exist:cT { ver@color.sty }

```

```

481 {
482   \cs_set_protected:Npn \__color_backend_pickup:N #1
483   {
484     \exp_last_unbraced:Nx \__color_backend_pickup:w
485     { \current@color } ~ 0 ~ 0 ~ 0 \s__color_stop #1
486   }
487   \cs_new_protected:Npn \__color_backend_pickup:w
488   #1 ~ #2 ~ #3 ~ #4 ~ #5 ~ #6 \s__color_stop #7
489   {
490     \str_if_eq:nnTF {#2} { g }
491     { \tl_set:Nn #7 { { gray } {#1} } }
492     {
493       \str_if_eq:nnTF {#4} { rg }
494       { \tl_set:Nn #7 { { rgb } { #1 ~ #2 ~ #3 } } }
495       {
496         \str_if_eq:nnTF {#5} { k }
497         { \tl_set:Nn #7 { { cmyk } { #1 ~ #2 ~ #3 ~ #4 } } }
498         {
499           \str_if_eq:nnTF {#2} { cs }
500           {
501             \tl_set:Nx #7 { { \use:n #1 } { #5 } }
502           }
503           {
504             \tl_set:Nn #7 { { gray } { 0 } }
505           }
506         }
507       }
508     }
509   }
510 }

```

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

511 </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/X_YTeX the backend version.

3.2.1 Common code

512 <*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.

513 \int_new:N \l__color_backend_stack_int

(End definition for \l__color_backend_stack_int.)

514 </dvipdfmx | luatex | pdftex | xetex>

3.2.2 dvipdfmx/X_YTeX

515 $\langle *dvipdfmx | xetex \rangle$

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.

```

516 \int_compare:nNnTF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
517 { \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3 { } }
518 {
519   \int_new:N \g__color_backend_stack_int
520   \cs_new_protected:Npx \__kernel_color_backend_stack_init:Nnn #1#2#3
521   {
522     \int_gincr:N \exp_not:N \g__color_backend_stack_int
523     \int_const:Nn #1 { \exp_not:N \g__color_backend_stack_int }
524     \use:x
525     {
526       \__kernel_backend_first_shipout:n
527       {
528         \__kernel_backend_literal:n
529         {
530           pdfcolorstackinit ~
531           \exp_not:N \int_use:N \exp_not:N \g__color_backend_stack_int
532           \c_space_tl
533           \exp_not:N \tl_if_blank:nF {#2} { #2 ~ }
534           (#3)
535         }
536       }
537     }
538   }
539   \cs_if_exist:cTF { main@pdfcolorstack }
540   {
541     \int_set:Nn \l__color_backend_stack_int
542     { \int_use:c { main@pdfcolorstack } }
543   }
544   {
545     \__kernel_color_backend_stack_init:Nnn \c__color_backend_main_stack_int
546     { page ~ direct } { 0 ~ g ~ 0 ~ G }
547     \int_set_eq:NN \l__color_backend_stack_int
548     \c__color_backend_main_stack_int
549     \int_const:cn { main@pdfcolorstack } { \c__color_backend_main_stack_int }
550   }

```

The backend automatically restores the stack color from the “classical” approach (pdf:bcolor) after a scope. That will be an issue for us, so we manually ensure that the one we are using is inserted.

```

551   \cs_gset_protected:Npn \__kernel_backend_scope_end:
552   {
553     \__kernel_backend_literal:n { x:grestore }
554     \__kernel_backend_literal:n
555     { pdfcolorstack ~ \g__color_backend_stack_int current }
556   }
557 }

```

(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.

```

558 \int_compare:nNnF \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
559 {
560   \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
561   {
562     \__kernel_backend_literal:x
563     {
564       pdfcolorstack ~
565       \int_eval:n {#1} ~
566       push ~ (#2)
567     }
568   }
569   \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
570   \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
571   {
572     \__kernel_backend_literal:x
573     {
574       pdfcolorstack ~
575       \int_eval:n {#1} ~
576       pop
577     }
578   }
579 }

```

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

```

580 </dvipdfmx | xetex>

```

3.2.3 LuaTeX and pdfTeX

```

581 <*luatex | pdftex>

```

```

\__kernel_color_backend_stack_init:Nnn

```

```

582 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
583 {
584   \int_const:Nn #1
585   {
586     <*luatex>
587     \tex_pdffeedback:D colorstackinit ~
588     </luatex>
589     <*pdftex>
590     \tex_pdfcolorstackinit:D
591     </pdftex>
592     \tl_if_blank:nF {#2} { #2 ~ }
593     {#3}
594   }
595 }

```

(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

```

```

596 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
597 {
598   <*luatex>

```

```

599     \tex_pdfextension:D colorstack ~
600   </luatex>
601   <*pdfTex>
602     \tex_pdfcolorstack:D
603   </pdfTex>
604     \int_eval:n {#1} ~ push ~ {#2}
605   }
606   \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
607   \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
608   {
609     <*luatex>
610     \tex_pdfextension:D colorstack ~
611   </luatex>
612   <*pdfTex>
613     \tex_pdfcolorstack:D
614   </pdfTex>
615     \int_eval:n {#1} ~ pop \scan_stop:
616   }

(End definition for \__kernel_color_backend_stack_push:nn and \__kernel_color_backend_stack_
pop:n.)

617 </luatex | pdfTex>

```

3.3 General color

3.3.1 dvips-style

```

618 <*dvips | dvisvgm>

\__color_backend_select_cmyk:n Push the data to the stack. In the case of dvips also saves the drawing color in raw
\__color_backend_select_gray:n PostScript.
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc

619 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
620 { \__color_backend_select:n { cmyk ~ #1 } }
621 \cs_new_protected:Npn \__color_backend_select_gray:n #1
622 { \__color_backend_select:n { gray ~ #1 } }
623 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
624 { \__color_backend_select:n { rgb ~ #1 } }
625 \cs_new_protected:Npn \__color_backend_select:n #1
626 {
627   \__kernel_backend_literal:n { color~push~ #1 }
628 <*dvips>
629   \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
630 </dvips>
631   \group_insert_after:N \__color_backend_reset:
632 }
633 \cs_new_protected:Npn \__color_backend_reset:
634 { \__kernel_backend_literal:n { color~pop } }

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

635 </dvips | dvisvgm>

```

3.3.2 LuaTeX and pdfTeX

636 $\langle *dvipdfmx \mid \text{luatex} \mid \text{pdfTeX} \mid \text{xetex} \rangle$

$\backslash l_color_backend_fill_tl$
 $\backslash l_color_backend_stroke_tl$

637 $\backslash tl_new:N \backslash l_color_backend_fill_tl$
 638 $\backslash tl_new:N \backslash l_color_backend_stroke_tl$

(End definition for $\backslash l_color_backend_fill_tl$ and $\backslash l_color_backend_stroke_tl$.)

$\backslash_color_backend_select_cmyk:n$
 $\backslash_color_backend_select_gray:n$
 $\backslash_color_backend_select_rgb:n$
 $\backslash_color_backend_select:nn$
 $\backslash_color_backend_reset:$

Store the values then pass to the stack.

639 $\backslash cs_new_protected:Npn \backslash_color_backend_select_cmyk:n \#1$
 640 $\{ \backslash_color_backend_select:nn \{ \#1 \sim k \} \{ \#1 \sim K \} \}$
 641 $\backslash cs_new_protected:Npn \backslash_color_backend_select_gray:n \#1$
 642 $\{ \backslash_color_backend_select:nn \{ \#1 \sim g \} \{ \#1 \sim G \} \}$
 643 $\backslash cs_new_protected:Npn \backslash_color_backend_select_rgb:n \#1$
 644 $\{ \backslash_color_backend_select:nn \{ \#1 \sim rg \} \{ \#1 \sim RG \} \}$
 645 $\backslash cs_new_protected:Npn \backslash_color_backend_select:nn \#1\#2$
 646 $\{$
 647 $\backslash tl_set:Nn \backslash l_color_backend_fill_tl \{ \#1 \}$
 648 $\backslash tl_set:Nn \backslash l_color_backend_stroke_tl \{ \#2 \}$
 649 $\backslash_kernel_color_backend_stack_push:nn \backslash l_color_backend_stack_int \{ \#1 \sim \#2 \}$
 650 $\backslash group_insert_after:N \backslash_color_backend_reset:$
 651 $\}$
 652 $\backslash cs_new_protected:Npn \backslash_color_backend_reset:$
 653 $\{ \backslash_kernel_color_backend_stack_pop:n \backslash l_color_backend_stack_int \}$

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

654 $\langle /dvipdfmx \mid \text{luatex} \mid \text{pdfTeX} \mid \text{xetex} \rangle$

3.3.3 dvipdfmx/X_YTeX

655 $\langle *dvipdfmx \mid \text{xetex} \rangle$

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.

$\backslash_color_backend_select_cmyk:n$
 $\backslash_color_backend_select_gray:n$
 $\backslash_color_backend_select_rgb:n$
 $\backslash_color_backend_reset:$

656 $\backslash int_compare:nNnT \backslash c_kernel_sys_dvipdfmx_version_int < \{ 20201111 \}$
 657 $\{$
 658 $\backslash cs_gset_protected:Npn \backslash_color_backend_select_cmyk:n \#1$
 659 $\{$
 660 $\backslash_kernel_backend_literal:n \{ pdf: bc \sim [\#1] \}$
 661 $\backslash group_insert_after:N \backslash_color_backend_reset:$
 662 $\}$
 663 $\backslash cs_gset_eq:NN \backslash_color_backend_select_gray:n \backslash_color_backend_select_cmyk:n$
 664 $\backslash cs_gset_eq:NN \backslash_color_backend_select_rgb:n \backslash_color_backend_select_cmyk:n$
 665 $\backslash cs_gset_protected:Npn \backslash_color_backend_reset:$
 666 $\{ \backslash_kernel_backend_literal:n \{ pdf: ec \} \}$
 667 $\}$

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

668 $\langle /dvipdfmx \mid \text{xetex} \rangle$

3.4 Separations

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

669 \langle *dvips \rangle

$\backslash_color_backend_select_separation:nn$
 $\backslash_color_backend_select_devicen:nn$

670 $\backslash cs_new_protected:Npn _color_backend_select_separation:nn \#1\#2$
671 $\{ _color_backend_select:n \{ separation \sim \#1 \sim \#2 \} \}$
672 $\backslash cs_new_eq:NN _color_backend_select_devicen:nn _color_backend_select_separation:nn$

(End definition for $\backslash_color_backend_select_separation:nn$ and $\backslash_color_backend_select_devicen:nn$.)

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.

$\backslash_color_backend_separation_init:nnnnn$
 $\backslash_color_backend_separation_init:nxxnn$
 $\backslash_color_backend_separation_init_aux:nnnnnn$
 $\backslash_color_backend_separation_init_DeviceCMYK:nnn$
 $\backslash_color_backend_separation_init_DeviceGray:nnn$
 $\backslash_color_backend_separation_init_DeviceRGB:nnn$
 $\backslash_color_backend_separation_init_Device:Nn$
 $\backslash_color_backend_separation_init:nnn$
 $\backslash_color_backend_separation_init_count:n$
 $\backslash_color_backend_separation_init_count:w$
 $\backslash_color_backend_separation_init:nnnn$
 $\backslash_color_backend_separation_init:w$
 $\backslash_color_backend_separation_init:n$
 $\backslash_color_backend_separation_init:nw$
 $\backslash_color_backend_separation_init_CIELAB:nnn$

673 $\backslash cs_new_protected:Npx _color_backend_separation_init:nnnnn \#1\#2\#3\#4\#5$
674 $\{$
675 $\backslash bool_if:NT _g_kernel_backend_header_bool$
676 $\{$
677 $\backslash exp_args:Nx _kernel_backend_first_shipout:n$
678 $\{$
679 $\backslash exp_not:N _color_backend_separation_init_aux:nnnnnn$
680 $\{ \backslash exp_not:N \backslash int_use:N _g_color_model_int \}$
681 $\{ \#1 \} \{ \#2 \} \{ \#3 \} \{ \#4 \} \{ \#5 \}$
682 $\}$
683 $\}$
684 $\}$
685 $\backslash cs_generate_variant:Nn _color_backend_separation_init:nnnnn \{ nxx \}$
686 $\backslash cs_new_protected:Npn _color_backend_separation_init_aux:nnnnnn \#1\#2\#3\#4\#5\#6$
687 $\{$
688 $\backslash_kernel_backend_literal:e$
689 $\{$
690 $!$
691 $TeXDict \sim begin \sim$
692 $/color \#1$
693 $\{$
694 $[\sim$
695 $/Separation \sim (\backslash str_convert_pdfname:n \{ \#2 \}) \sim$
696 $[\sim \#3 \sim] \sim$
697 $\{$
698 $\backslash cs_if_exist_use:cF \{ _color_backend_separation_init_ \#3 :nnn \}$
699 $\{ _color_backend_separation_init:nnn \}$
700 $\{ \#4 \} \{ \#5 \} \{ \#6 \}$
701 $\}$
702 $] \sim setcolorspace$
703 $\} \sim def \sim$
704 end
705 $\}$
706 $\}$
707 $\backslash cs_new:cpn \{ _color_backend_separation_init_ /DeviceCMYK :nnn \} \#1\#2\#3$
708 $\{ _color_backend_separation_init_Device:Nn 4 \{ \#3 \} \}$
709 $\backslash cs_new:cpn \{ _color_backend_separation_init_ /DeviceGray :nnn \} \#1\#2\#3$
710 $\{ _color_backend_separation_init_Device:Nn 1 \{ \#3 \} \}$

```

711 \cs_new:cpn { __color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
712 { \__color_backend_separation_init_Device:Nn 2 {#3} }
713 \cs_new:Npn \__color_backend_separation_init_Device:Nn #1#2
714 {
715     #2 ~
716     \prg_replicate:nn {#1}
717     { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
718     \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
719 }

```

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.

```

720 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3
721 {
722     \exp_args:Ne \__color_backend_separation_init:nnnn
723     { \__color_backend_separation_init_count:n {#2} }
724     {#1} {#2} {#3}
725 }
726 \cs_new:Npn \__color_backend_separation_init_count:n #1
727 { \int_eval:n { 0 \__color_backend_separation_init_count:w #1 ~ \s_color_stop } }
728 \cs_new:Npn \__color_backend_separation_init_count:w #1 ~ #2 \s_color_stop
729 {
730     +1
731     \tl_if_blank:nF {#2}
732     { \__color_backend_separation_init_count:w #2 \s_color_stop }
733 }

```

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.

```

734 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
735 {
736     \__color_backend_separation_init:w #3 ~ \s_color_stop #4 ~ \s_color_stop
737     \prg_replicate:nn {#1}
738     {
739         pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
740         \int_eval:n { 3 * #1 } ~ index ~ mul ~
741         2 ~ index ~ add ~
742         \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
743     }
744     \int_step_function:nnnN {#1} { -1 } { 1 }
745     \__color_backend_separation_init:n
746     \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
747     \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }

```

```

748     \tl_if_blank:nF {#2}
749     { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
750   }
751   \cs_new:Npn \__color_backend_separation_init:w
752     #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
753   {
754     #1 ~ #3 ~ 0 ~
755     \tl_if_blank:nF {#2}
756     { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }
757   }
758   \cs_new:Npn \__color_backend_separation_init:n #1
759   { \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.

```

760   \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s__color_stop
761   {
762     #2 ~ #3 ~
763     2 ~ index ~ 2 ~ index ~ lt ~
764     { ~ pop ~ excl ~ pop ~ } ~
765     { ~
766       2 ~ index ~ 1 ~ index ~ gt ~
767       { ~ excl ~ pop ~ excl ~ pop ~ } ~
768       { ~ pop ~ pop ~ } ~
769       ifelse ~
770     }
771     ifelse ~
772     #1 ~ 1 ~ roll ~
773     \tl_if_blank:nF {#4}
774     { \__color_backend_separation_init:nw {#1} #4 \s__color_stop }
775   }

```

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.

```

776   \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
777   {
778     \__color_backend_separation_init:nxxxnn
779     {#2}
780     {
781       /CIEBasedABC ~
782       << ~
783       /RangeABC ~ [ ~ \c__color_model_range_CIELAB_tl \c_space_tl ] ~
784       /DecodeABC ~
785       [ ~
786         { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
787         { ~ 500 ~ div ~ } ~ bind ~
788         { ~ 200 ~ div ~ } ~ bind ~
789       ] ~
790       /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
791       /DecodeLMN ~
792       [ ~
793         { ~
794           dup ~ 6 ~ 29 ~ div ~ ge ~
795           { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~

```

```

796         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
797         ifelse ~
798         0.9505 ~ mul ~
799     } ~ bind ~
800     { ~
801         dup ~ 6 ~ 29 ~ div ~ ge ~
802         { ~ dup ~ dup ~ mul ~ mul ~ } ~
803         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
804         ifelse ~
805     } ~ bind ~
806     { ~
807         dup ~ 6 ~ 29 ~ div ~ ge ~
808         { ~ dup ~ dup ~ mul ~ mul ~ } ~
809         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
810         ifelse ~
811         1.0890 ~ mul ~
812     } ~ bind
813 ] ~
814 /WhitePoint ~
815 [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
816 >>
817 }
818 { \c__color_model_range_CIELAB_tl }
819 { 100 ~ 0 ~ 0 }
820 {#3}
821 }

```

(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.

```

822 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
823 {
824     \__kernel_backend_literal:e
825     {
826         !
827         TeXDict ~ begin ~
828         /color \int_use:N \g__color_model_int
829         {
830             [ ~
831                 /DeviceN ~
832                 [ ~ #1 ~ ] ~
833                 #2 ~
834                 { ~ #3 ~ } ~
835             ] ~ setcolorspace
836         } ~ def ~
837     end
838 }
839 }

```

(End definition for `__color_backend_devicen_init:nnn`.)

```

840 </dvips>
841 <*dvisvgm>

```

`_color_backend_select_separation:nn` No support at present.
`_color_backend_select_devicen:nn`

```

842 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2 { }
843 \cs_new_protected:Npn \_color_backend_select_devicen:nn #1#2 { }

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

```

`_color_backend_separation_init:nnnnn` No support at present.
`_color_backend_separation_init_CIELAB:nnn`

```

844 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }
845 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnnnn #1#2#3 { }

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

```

```

846 </dvisvgm>
847 <*dvipdfmx | luatex | pdfTeX | xetex>

```

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

```

848 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
849 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
850 \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.)

```

`_color_backend_separation_init:nnnnn` 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.
`_color_backend_separation_init:n`

```

851 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5
852 {
853   \pdf_object_unnamed_write:nx { dict }
854   {
855     /FunctionType ~ 2
856     /Domain ~ [0 ~ 1]
857     \tl_if_blank:nF {#3} { /Range ~ [#3] }
858     /C0 ~ [#4] ~
859     /C1 ~ [#5] /N ~ 1
860   }
861   \_color_backend_separation_init:n
862   {
863     /Separation ~
864     / \str_convert_pdfname:n {#1} ~ #2 ~
865     \pdf_object_ref_last:
866   }
867   \bool_lazy_and:nnT
868   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
869   { \pdfmanagement_if_active_p: }
870   {
871     \use:x
872     {
873       \pdfmanagement_add:nnn
874       { Page / Resources / ColorSpace }
875       { color \int_use:N \g__color_model_int }
876       { \pdf_object_ref_last: }

```



```

877     }
878   }
879 }
880 \cs_new_protected:Npn \__color_backend_separation_init:n #1
881 {
882   \pdf_object_unnamed_write:nx { array } {#1}
883 }

```

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

```

884 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
885 {
886   \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
887   {
888     \pdf_object_new:nn { __color_illuminant_CIELAB_ #1 } { array }
889     \pdf_object_write:nx { __color_illuminant_CIELAB_ #1 }
890     {
891       /Lab ~
892       <<
893       /WhitePoint ~
894       [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
895       /Range ~ [ \c__color_model_range_CIELAB_tl ]
896       >>
897     }
898   }
899   \__color_backend_separation_init:nnnnn
900   {#2}
901   { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
902   { \c__color_model_range_CIELAB_tl }
903   { 100 ~ 0 ~ 0 }
904   {#3}
905 }

```

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

```

\__color_backend_devicen_init:nnn
\__color_backend_devicen_init:w
\__color_backend_devicen_init:n

```

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

```

906 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
907 {
908   \pdf_object_unnamed_write:nx { stream }
909   {
910     {
911       /FunctionType ~ 4 ~
912       /Domain ~
913       [ ~
914         \prg_replicate:nn
915         { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
916         { 0 ~ 1 ~ }
917       ] ~
918       /Range ~
919       [ ~
920         \str_case:nn {#2}
921         {
922           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }

```

```

923         { /DeviceGray } { 0 ~ 1 }
924         { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
925     } ~
926 ]
927 }
928 { {#3} }
929 }
930 \__color_backend_separation_init:n
931 {
932     /DeviceN ~
933     [ ~ #1 ~ ] ~
934     #2 ~
935     \pdf_object_ref_last:
936 }
937 \bool_lazy_and:nnT
938 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
939 { \pdfmanagement_if_active_p: }
940 {
941     \use:x
942     {
943         \pdfmanagement_add:nnn
944         { Page / Resources / ColorSpace }
945         { color \int_use:N \g__color_model_int }
946         { \pdf_object_ref_last: }
947     }
948 }
949 }
950 \cs_new:Npn \__color_backend_devicen_init:w #1 ~ #2 \s__color_stop
951 {
952     + 1
953     \tl_if_blank:nF {#2}
954     { \__color_backend_devicen_init:w #2 \s__color_stop }
955 }
956 \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.)
957 </dvipdfmx | luatex | pdftex | xetex>
958 <*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.

```

959 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
960 {
961     \cs_gset_protected:Npn \__color_backend_select_separation:nn #1#2 { }
962     \cs_gset_eq:NN \__color_backend_select_devicen:nn
963     \__color_backend_select_separation:nn
964 }
(End definition for \__color_backend_select_separation:nn and \__color_backend_select_devicen:nn.)
965 </dvipdfmx | xetex>

```

3.5 Fill and stroke color

Here, `dvipdfmx/XqTeX` 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.

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

Drawing (fill/stroke) color is handled in `dvipdfmx/XqTeX` 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.

```

967 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
968   { \__color_backend_fill:n { #1 ~ k } }
969 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
970   { \__color_backend_fill:n { #1 ~ g } }
971 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
972   { \__color_backend_fill:n { #1 ~ rg } }
973 \cs_new_protected:Npn \__color_backend_fill:n #1
974   {
975     \tl_set:Nn \l__color_backend_fill_tl {#1}
976     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
977       { #1 ~ \l__color_backend_stroke_tl }
978     \group_insert_after:N \__color_backend_reset:
979   }
980 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
981   { \__color_backend_stroke:n { #1 ~ K } }
982 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
983   { \__color_backend_stroke:n { #1 ~ G } }
984 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
985   { \__color_backend_stroke:n { #1 ~ RG } }
986 \cs_new_protected:Npn \__color_backend_stroke:n #1
987   {
988     \tl_set:Nn \l__color_backend_stroke_tl {#1}
989     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
990       { \l__color_backend_fill_tl \c_space_tl #1 }
991     \group_insert_after:N \__color_backend_reset:
992   }

```

(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
993 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
994   { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
995 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
996   { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
997 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
998 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

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

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

1000 `<*dvipdfmx | xetex>`

```

\__color_backend_fill_cmyk:n Deal with older (x)dvipdfmx.
\__color_backend_fill_gray:n 1001 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
\__color_backend_fill_rgb:n 1002 {
  \__color_backend_reset: 1003 \cs_gset_protected:Npn \__color_backend_fill_cmyk:n #1
  \__color_backend_stroke:n 1004 {
    \__kernel_backend_literal:n { pdf: bc ~ [#1] } 1005
    \group_insert_after:N \__color_backend_reset: 1006
  } 1007
  \cs_gset_eq:NN \__color_backend_fill_gray:n \__color_backend_fill_cmyk:n 1008
  \cs_gset_eq:NN \__color_backend_fill_rgb:n \__color_backend_fill_cmyk:n 1009
  \cs_gset_protected:Npn \__color_backend_reset: 1010
  { \__kernel_backend_literal:n { pdf: ec } } 1011
  \cs_gset_protected:Npn \__color_backend_stroke:n #1 1012
  { \__kernel_backend_literal:n {#1} } 1013
  \cs_gset_protected:Npn \__color_backend_fill_separation:nn #1#2 { } 1014
  \cs_gset_eq:NN \__color_backend_fill_devicen:nn 1015
  \__color_backend_fill_separation:nn 1016
  \cs_gset_eq:NN \__color_backend_stroke_separation:nn 1017
  \__color_backend_fill_separation:nn 1018
  \cs_gset_eq:NN \__color_backend_stroke_devicen:nn 1019
  \__color_backend_stroke_separation:nn 1020
} 1021

```

(End definition for __color_backend_fill_cmyk:n and others.)

1022 </dvipdfmx | xetex>

1023 <*dvips>

```

\__color_backend_fill_cmyk:n Fill color here is the same as general color except we skip the stroke part.
\__color_backend_fill_gray:n 1024 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
\__color_backend_fill_rgb:n 1025 { \__color_backend_fill:n { cmyk ~ #1 } }
  \__color_backend_fill:n 1026 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
    \color_backend_stroke_cmyk:n 1027 { \__color_backend_fill:n { gray ~ #1 } }
    \color_backend_stroke_gray:n 1028 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
    \__color_backend_stroke_rgb:n 1029 { \__color_backend_fill:n { rgb ~ #1 } }
1030 \cs_new_protected:Npn \__color_backend_fill:n #1
1031 {
1032   \__kernel_backend_literal:n { color~push~ #1 }
1033   \group_insert_after:N \__color_backend_reset:
1034 }
1035 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1036 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1037 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1038 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1039 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1040 { \__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 1041 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
  \__color_backend_fill_devicen:nn 1042 { \__color_backend_fill:n { separation ~ #1 ~ #2 } }
  \__color_backend_stroke_devicen:nn 1043 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1044 { \__kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }

```

```

1045 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1046 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

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

1047 </dvips>
1048 <*dvisvgm>

```

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

```

\_color_backend_fill_gray:n 1049 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
\_color_backend_fill_rgb:n   1050 { \_color_backend_fill:n { cmyk ~ #1 } }
\_color_backend_fill:n       1051 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
                             1052 { \_color_backend_fill:n { gray ~ #1 } }
                             1053 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
                             1054 { \_color_backend_fill:n { rgb ~ #1 } }
                             1055 \cs_new_protected:Npn \_color_backend_fill:n #1
                             1056 {
                             1057     \_kernel_backend_literal:n { color~push~ #1 }
                             1058     \group_insert_after:N \_color_backend_reset:
                             1059 }

```

(End definition for _color_backend_fill_cmyk:n and others.)

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

```

\_color_backend_stroke_cmyk:w 1060 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
\_color_backend_stroke_gray:n  1061 { \_color_backend_cmyk:w #1 \s_color_stop }
\_color_backend_stroke_gray_aux:n 1062 \cs_new_protected:Npn \_color_backend_stroke_cmyk:w
\_color_backend_stroke_rgb:n    1063 #1 ~ #2 ~ #3 ~ #4 \s_color_stop
\_color_backend_stroke_rgb:w    1064 {
\_color_backend:nnn             1065     \use:x
                             1066     {
                             1067         \_color_backend:nnn
                             1068         { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
                             1069         { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
                             1070         { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }
                             1071     }
                             1072 }
1073 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1074 {
1075     \use:x
1076     {
1077         \_color_backend_stroke_gray_aux:n
1078         { \fp_eval:n { 100 * (#1) } }
1079     }
1080 }
1081 \cs_new_protected:Npn \_color_backend_stroke_gray_aux:n #1
1082 { \_color_backend:nnn {#1} {#1} {#1} }
1083 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1084 { \_color_backend_rgb:w #1 \s_color_stop }
1085 \cs_new_protected:Npn \_color_backend_stroke_rgb:w
1086 #1 ~ #2 ~ #3 \s_color_stop
1087 {
1088     \use:x
1089     {

```

```

1090         \_color_backend:nnn
1091         { \fp_eval:n { 100 * (#1) } }
1092         { \fp_eval:n { 100 * (#2) } }
1093         { \fp_eval:n { 100 * (#3) } }
1094     }
1095 }
1096 \cs_new_protected:Npx \_color_backend:nnn #1#2#3
1097 {
1098     \_kernel_backend_scope:n
1099     {
1100         stroke =
1101         "
1102         rgb
1103         (
1104             #1 \c_percent_str ,
1105             #2 \c_percent_str ,
1106             #3 \c_percent_str
1107         )
1108         "
1109     }
1110 }

```

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

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn

```

At present, these are no-ops.

```

1111 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1112 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2 { }
1113 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1114 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

```

1115 </divisvgn>
1116 </package>

```

4 l3backend-draw Implementation

```

1117 <*package>
1118 <@@=draw>

```

4.1 dvips backend

```

1119 <*dvips>

```

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

```

\_draw_backend_literal:n
\_draw_backend_literal:x
1120 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
1121 \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.

```

1122 \cs_new_protected:Npn \_draw_backend_begin:
1123 {
1124   \_kernel_backend_literal:n { ps::[begin] }
1125   \_draw_backend_literal:n { @beginspecial }
1126 }
1127 \cs_new_protected:Npn \_draw_backend_end:
1128 {
1129   \_draw_backend_literal:n { @endspecial }
1130   \_kernel_backend_literal:n { ps::[end] }
1131 }

```

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

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

```

1132 \cs_new_protected:Npn \_draw_backend_scope_begin:
1133 { \_draw_backend_literal:n { save } }
1134 \cs_new_protected:Npn \_draw_backend_scope_end:
1135 { \_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.

```

1136 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1137 {
1138   \_draw_backend_literal:x
1139   {
1140     \dim_to_decimal_in_bp:n {#1} ~
1141     \dim_to_decimal_in_bp:n {#2} ~ moveto
1142   }
1143 }
1144 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1145 {
1146   \_draw_backend_literal:x
1147   {
1148     \dim_to_decimal_in_bp:n {#1} ~
1149     \dim_to_decimal_in_bp:n {#2} ~ lineto
1150   }
1151 }
1152 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1153 {
1154   \_draw_backend_literal:x
1155   {
1156     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~

```

```

1157         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1158         moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1159     }
1160 }
1161 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1162 {
1163     \__draw_backend_literal:x
1164     {
1165         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1166         \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1167         \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1168         curveto
1169     }
1170 }

```

(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
1171 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1172 { \bool_gset_true:N \g__draw_draw_eor_bool }
1173 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1174 { \bool_gset_false:N \g__draw_draw_eor_bool }
1175 \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 also desirable to have the clip keyword after a stroke or fill. To achieve those outcomes, there is some work to do. For color, the stroke color is simple but the fill one has to be inserted by hand. For clipping, the required ordering is achieved using a T_EX switch. All of the operations end with a new path instruction as they do not terminate (again in contrast to PDF).

```

\__draw_backend_closepath:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
1176 \cs_new_protected:Npn \__draw_backend_closepath:
1177 { \__draw_backend_literal:n { closepath } }
1178 \cs_new_protected:Npn \__draw_backend_stroke:
1179 {
1180     \__draw_backend_literal:n { gsave }
1181     \__draw_backend_literal:n { color.sc }
1182     \__draw_backend_literal:n { stroke }
1183     \__draw_backend_literal:n { grestore }
1184     \bool_if:NT \g__draw_draw_clip_bool
1185     {
1186         \__draw_backend_literal:x
1187         {
1188             \bool_if:NT \g__draw_draw_eor_bool { eo }
1189             clip
1190         }
1191     }
1192     \__draw_backend_literal:n { newpath }
1193     \bool_gset_false:N \g__draw_draw_clip_bool
1194 }
1195 \cs_new_protected:Npn \__draw_backend_closestroke:
1196 {

```



```

1197     \__draw_backend_closepath:
1198     \__draw_backend_stroke:
1199   }
1200   \cs_new_protected:Npn \__draw_backend_fill:
1201   {
1202     \__draw_backend_literal:x
1203     {
1204       \bool_if:NT \g__draw_draw_eor_bool { eo }
1205       fill
1206     }
1207     \bool_if:NT \g__draw_draw_clip_bool
1208     {
1209       \__draw_backend_literal:x
1210       {
1211         \bool_if:NT \g__draw_draw_eor_bool { eo }
1212         clip
1213       }
1214     }
1215     \__draw_backend_literal:n { newpath }
1216     \bool_gset_false:N \g__draw_draw_clip_bool
1217   }
1218   \cs_new_protected:Npn \__draw_backend_fillstroke:
1219   {
1220     \__draw_backend_literal:x
1221     {
1222       \bool_if:NT \g__draw_draw_eor_bool { eo }
1223       fill
1224     }
1225     \__draw_backend_literal:n { gsave }
1226     \__draw_backend_literal:n { color.sc }
1227     \__draw_backend_literal:n { stroke }
1228     \__draw_backend_literal:n { grestore }
1229     \bool_if:NT \g__draw_draw_clip_bool
1230     {
1231       \__draw_backend_literal:x
1232       {
1233         \bool_if:NT \g__draw_draw_eor_bool { eo }
1234         clip
1235       }
1236     }
1237     \__draw_backend_literal:n { newpath }
1238     \bool_gset_false:N \g__draw_draw_clip_bool
1239   }
1240   \cs_new_protected:Npn \__draw_backend_clip:
1241   { \bool_gset_true:N \g__draw_draw_clip_bool }
1242   \bool_new:N \g__draw_draw_clip_bool
1243   \cs_new_protected:Npn \__draw_backend_discardpath:
1244   {
1245     \bool_if:NT \g__draw_draw_clip_bool
1246     {
1247       \__draw_backend_literal:x
1248       {
1249         \bool_if:NT \g__draw_draw_eor_bool { eo }
1250         clip

```

```

1251     }
1252   }
1253   \__draw_backend_literal:n { newpath }
1254   \bool_gset_false:N \g__draw_draw_clip_bool
1255 }

```

(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_but:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1256 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1257 {
1258   \__draw_backend_literal:x
1259   {
1260     [
1261       \exp_args:Nf \use:n
1262       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1263     ] ~
1264     \dim_to_decimal_in_bp:n {#2} ~ setdash
1265   }
1266 }
1267 \cs_new:Npn \__draw_backend_dash:n #1
1268 { ~ \dim_to_decimal_in_bp:n {#1} }
1269 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1270 {
1271   \__draw_backend_literal:x
1272   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1273 }
1274 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1275 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1276 \cs_new_protected:Npn \__draw_backend_cap_but:
1277 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1278 \cs_new_protected:Npn \__draw_backend_cap_round:
1279 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1280 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1281 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1282 \cs_new_protected:Npn \__draw_backend_join_miter:
1283 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1284 \cs_new_protected:Npn \__draw_backend_join_round:
1285 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1286 \cs_new_protected:Npn \__draw_backend_join_bevel:
1287 { \__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.

```

1288 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1289 {
1290   \__draw_backend_literal:n
1291   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1292 }

```

(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).

```

1293 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
1294 {
1295   \_draw_backend_literal:n { @endspecial }
1296   \_draw_backend_literal:n { [end] }
1297   \_draw_backend_literal:n { [begin] }
1298   \_draw_backend_literal:n { save }
1299   \_draw_backend_literal:n { currentpoint }
1300   \_draw_backend_literal:n { currentpoint~translate }
1301   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1302   \_draw_backend_cm:nnnn { #2 } { #3 } { #4 } { #5 }
1303   \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1304   \_draw_backend_literal:n { neg~exch~neg~exch~translate }
1305   \_draw_backend_literal:n { [end] }
1306   \hbox_overlap_right:n { \box_use:N #1 }
1307   \_draw_backend_literal:n { [begin] }
1308   \_draw_backend_literal:n { restore }
1309   \_draw_backend_literal:n { [end] }
1310   \_draw_backend_literal:n { [begin] }
1311   \_draw_backend_literal:n { @beginspecial }
1312 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

1313 `</dvips>`

4.2 Lua \TeX , pdf \TeX , dvipdfmx and X \TeX

Lua \TeX , pdf \TeX , dvipdfmx and X \TeX directly produce PDF output and understand a shared set of specials for drawing commands.

1314 `<*dvipdfmx | luatex | pdftex | xetex>`

4.2.1 Drawing

`_draw_backend_literal:n`
`_draw_backend_literal:x`

Pass data through using a dedicated interface.

```

1315 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_pdf:n
1316 \cs_generate_variant:Nn \_draw_backend_literal:n { x }

```

(End definition for `_draw_backend_literal:n`.)

`_draw_backend_begin:`
`_draw_backend_end:`

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

```

1317 \cs_new_protected:Npn \_draw_backend_begin:
1318 { \_draw_backend_scope_begin: }

```

```

1319 \cs_new_protected:Npn \__draw_backend_end:
1320 { \__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:
1321 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1322 \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_curveto:nnnnnn
  \__draw_backend_rectangle:nnnn
1323 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1324 {
1325   \__draw_backend_literal:x
1326   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1327 }
1328 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1329 {
1330   \__draw_backend_literal:x
1331   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1332 }
1333 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1334 {
1335   \__draw_backend_literal:x
1336   {
1337     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1338     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1339     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1340     c
1341   }
1342 }
1343 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1344 {
1345   \__draw_backend_literal:x
1346   {
1347     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1348     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1349     re
1350   }
1351 }

```

(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
1352 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1353 { \bool_gset_true:N \g__draw_draw_eor_bool }
1354 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1355 { \bool_gset_false:N \g__draw_draw_eor_bool }
1356 \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.)

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

```

\__draw_backend_closepath:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
1357 \cs_new_protected:Npn \__draw_backend_closepath:
1358 { \__draw_backend_literal:n { h } }
1359 \cs_new_protected:Npn \__draw_backend_stroke:
1360 { \__draw_backend_literal:n { S } }
1361 \cs_new_protected:Npn \__draw_backend_closestroke:
1362 { \__draw_backend_literal:n { s } }
1363 \cs_new_protected:Npn \__draw_backend_fill:
1364 {
1365   \__draw_backend_literal:x
1366   { f \bool_if:NT \g__draw_draw_eor_bool * }
1367 }
1368 \cs_new_protected:Npn \__draw_backend_fillstroke:
1369 {
1370   \__draw_backend_literal:x
1371   { B \bool_if:NT \g__draw_draw_eor_bool * }
1372 }
1373 \cs_new_protected:Npn \__draw_backend_clip:
1374 {
1375   \__draw_backend_literal:x
1376   { W \bool_if:NT \g__draw_draw_eor_bool * }
1377 }
1378 \cs_new_protected:Npn \__draw_backend_discardpath:
1379 { \__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_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1380 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1381 {
1382   \__draw_backend_literal:x
1383   {
1384     [
1385       \exp_args:Nf \use:n
1386       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1387     ] ~
1388     \dim_to_decimal_in_bp:n {#2} ~ d
1389   }
1390 }
1391 \cs_new:Npn \__draw_backend_dash:n #1
1392 { ~ \dim_to_decimal_in_bp:n {#1} }
1393 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1394 {
1395   \__draw_backend_literal:x
1396   { \dim_to_decimal_in_bp:n {#1} ~ w }
1397 }
1398 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1399 { \__draw_backend_literal:x { #1 ~ M } }
1400 \cs_new_protected:Npn \__draw_backend_cap_butt:
1401 { \__draw_backend_literal:n { 0 ~ J } }
1402 \cs_new_protected:Npn \__draw_backend_cap_round:
1403 { \__draw_backend_literal:n { 1 ~ J } }
1404 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1405 { \__draw_backend_literal:n { 2 ~ J } }

```

```

1406 \cs_new_protected:Npn \__draw_backend_join_miter:
1407 { \__draw_backend_literal:n { 0 ~ j } }
1408 \cs_new_protected:Npn \__draw_backend_join_round:
1409 { \__draw_backend_literal:n { 1 ~ j } }
1410 \cs_new_protected:Npn \__draw_backend_join_bevel:
1411 { \__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!

```

1412 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1413 {
1414 <*luatex | pdftex>
1415 \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1416 </luatex | pdftex>
1417 <*dvipdfmx | xetex>
1418 \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1419 \__draw_backend_cm_aux:nnnn
1420 </dvipdfmx | xetex>
1421 }
1422 <*dvipdfmx | xetex>
1423 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1424 {
1425 \__kernel_backend_literal:x
1426 {
1427 x:rotate~
1428 \fp_compare:nNnTF {#1} = \c_zero_fp
1429 { 0 }
1430 { \fp_eval:n { round ( -#1 , 5 ) } }
1431 }
1432 \__kernel_backend_literal:x
1433 {
1434 x:scale~
1435 \fp_eval:n { round ( #2 , 5 ) } ~
1436 \fp_eval:n { round ( #3 , 5 ) }
1437 }
1438 \__kernel_backend_literal:x
1439 {
1440 x:rotate~
1441 \fp_compare:nNnTF {#4} = \c_zero_fp
1442 { 0 }
1443 { \fp_eval:n { round ( -#4 , 5 ) } }
1444 }
1445 }
1446 </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.

```

1447 <*dvipdfmx | xetex>
1448 \cs_new_protected:Npn \_draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1449 {
1450   \use:x
1451   {
1452     \_draw_backend_cm_decompose_auxi:nnnnN
1453     { \fp_eval:n { (#1 + #4) / 2 } }
1454     { \fp_eval:n { (#1 - #4) / 2 } }
1455     { \fp_eval:n { (#3 + #2) / 2 } }
1456     { \fp_eval:n { (#3 - #2) / 2 } }
1457   }
1458   #5
1459 }
1460 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1461 {
1462   \use:x
1463   {
1464     \_draw_backend_cm_decompose_auxii:nnnnN
1465     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1466     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1467     { \fp_eval:n { atand ( #3 , #2 ) } }
1468     { \fp_eval:n { atand ( #4 , #1 ) } }
1469   }
1470   #5
1471 }
1472 \cs_new_protected:Npn \_draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5

```

```

1473 {
1474   \use:x
1475   {
1476     \__draw_backend_cm_decompose_auxiii:nnnnN
1477     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1478     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1479     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1480     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1481   }
1482   #5
1483 }
1484 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1485 {
1486   \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1487   { #5 {#1} {#2} {#3} {#4} }
1488   { #5 {#1} {#3} {#2} {#4} }
1489 }
1490 </dvipdfmx | xetex>

```

(End definition for `__draw_backend_cm_decompose:nnnnN` and others.)

`__draw_backend_box_use:Nnnnn`

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

```

1491 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1492 {
1493   \__kernel_backend_scope_begin:
1494   <*luatex | pdftex>
1495   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1496   </luatex | pdftex>
1497   <*dvipdfmx | xetex>
1498   \__kernel_backend_literal:n
1499   { pdf:btrans-matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1500   </dvipdfmx | xetex>
1501   \hbox_overlap_right:n { \box_use:N #1 }
1502   <*dvipdfmx | xetex>
1503   \__kernel_backend_literal:n { pdf:etrans }
1504   </dvipdfmx | xetex>
1505   \__kernel_backend_scope_end:
1506 }

```

(End definition for `__draw_backend_box_use:Nnnnn`.)

```

1507 </dvipdfmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1508 <*dvisvgm>

```

`__draw_backend_literal:n`

The same as the more general literal call.

`__draw_backend_literal:x`

```

1509 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1510 \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
 _draw_backend_end: done inside a scope, which as described below

```

1511 \cs_new_protected:Npn \_draw_backend_begin:
1512 {
1513   \_kernel_backend_scope_begin:
1514   \_kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1515 }
1516 \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
 _draw_backend_lineto:nn values as they are given, the entire path needs to be collected up before being output
 _draw_backend_rectangle:nnnn in one go. For that we use a dedicated storage routine, which adds spaces as required.
 _draw_backend_curveto:nnnnnn Since paths should be fully expanded there is no need to worry about the internal x-type
 _draw_backend_add_to_path:n expansion.
 \g__draw_draw_path_tl

```

1517 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1518 {
1519   \_draw_backend_add_to_path:n
1520   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1521 }
1522 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1523 {
1524   \_draw_backend_add_to_path:n
1525   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1526 }
1527 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1528 {
1529   \_draw_backend_add_to_path:n
1530   {
1531     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1532     h ~ \dim_to_decimal:n {#3} ~
1533     v ~ \dim_to_decimal:n {#4} ~
1534     h ~ \dim_to_decimal:n { -#3 } ~
1535     Z
1536   }
1537 }
1538 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1539 {
1540   \_draw_backend_add_to_path:n
1541   {
1542     C ~
1543     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1544     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1545     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1546   }
1547 }
1548 \cs_new_protected:Npn \_draw_backend_add_to_path:n #1
1549 {
1550   \tl_gset:Nx \g__draw_draw_path_tl
1551   {
1552     \g__draw_draw_path_tl

```

```

1553         \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1554         #1
1555     }
1556 }
1557 \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:
1558 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1559 { \__draw_backend_scope:n { fill-rule="evenodd" } }
1560 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1561 { \__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 means setting up the various requirements in a shared auxiliary which deals with the bits and pieces. Clipping paths are reused for path drawing; not essential but avoids constructing them twice. Discarding a path needs a separate function as it's not quite the same.

```

\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int
1562 \cs_new_protected:Npn \__draw_backend_closepath:
1563 { \__draw_backend_add_to_path:n { Z } }
1564 \cs_new_protected:Npn \__draw_backend_path:n #1
1565 {
1566     \bool_if:NTF \g__draw_draw_clip_bool
1567     {
1568         \int_gincr:N \g__draw_clip_path_int
1569         \__draw_backend_literal:x
1570         {
1571             < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1572             { ?nl }
1573             <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1574             < /clipPath > { ? nl }
1575             <
1576                 use~xlink:href =
1577                 "\c_hash_str l3path \int_use:N \g__draw_path_int " ~
1578                 #1
1579             />
1580         }
1581         \__draw_backend_scope:x
1582         {
1583             clip-path =
1584             "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int )"
1585         }
1586     }
1587     {
1588         \__draw_backend_literal:x
1589         { <path ~ d=" \g__draw_draw_path_tl " ~ #1 /> }
1590     }
1591     \tl_gclear:N \g__draw_draw_path_tl
1592     \bool_gset_false:N \g__draw_draw_clip_bool
1593 }
1594 \int_new:N \g__draw_path_int

```

```

1595 \cs_new_protected:Npn \__draw_backend_stroke:
1596 { \__draw_backend_path:n { style="fill:none" } }
1597 \cs_new_protected:Npn \__draw_backend_closestroke:
1598 {
1599   \__draw_backend_closepath:
1600   \__draw_backend_stroke:
1601 }
1602 \cs_new_protected:Npn \__draw_backend_fill:
1603 { \__draw_backend_path:n { style="stroke:none" } }
1604 \cs_new_protected:Npn \__draw_backend_fillstroke:
1605 { \__draw_backend_path:n { } }
1606 \cs_new_protected:Npn \__draw_backend_clip:
1607 { \bool_gset_true:N \g__draw_draw_clip_bool }
1608 \bool_new:N \g__draw_draw_clip_bool
1609 \cs_new_protected:Npn \__draw_backend_discardpath:
1610 {
1611   \bool_if:NT \g__draw_draw_clip_bool
1612   {
1613     \int_gincr:N \g__draw_clip_path_int
1614     \__draw_backend_literal:x
1615     {
1616       < clipPath~id = " l3cp \int_use:N \g__draw_clip_path_int " >
1617       { ?nl }
1618       <path~d=" \g__draw_draw_path_tl "/> { ?nl }
1619       < /clipPath >
1620     }
1621     \__draw_backend_scope:x
1622     {
1623       clip-path =
1624       "url( \c_hash_str l3cp \int_use:N \g__draw_clip_path_int)"
1625     }
1626   }
1627   \tl_gclear:N \g__draw_draw_path_tl
1628   \bool_gset_false:N \g__draw_draw_clip_bool
1629 }

```

(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:
1630 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1631 {
1632   \use:x
1633   {
1634     \__draw_backend_dash_aux:nn
1635     { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1636     { \dim_to_decimal:n {#2} }
1637   }
1638 }
1639 \cs_new:Npn \__draw_backend_dash:n #1
1640 { , \dim_to_decimal_in_bp:n {#1} }
1641 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1642 {
1643   \__draw_backend_scope:x

```

```

1644     {
1645         stroke-dasharray =
1646         "
1647             \tl_if_empty:oTF { \use_none:n #1 }
1648             { none }
1649             { \use_none:n #1 }
1650         " ~
1651         stroke-offset=" #2 "
1652     }
1653 }
1654 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1655 { \__draw_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1656 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1657 { \__draw_backend_scope:x { stroke-miterlimit=" #1 " } }
1658 \cs_new_protected:Npn \__draw_backend_cap_but:
1659 { \__draw_backend_scope:n { stroke-linecap="butt" } }
1660 \cs_new_protected:Npn \__draw_backend_cap_round:
1661 { \__draw_backend_scope:n { stroke-linecap="round" } }
1662 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1663 { \__draw_backend_scope:n { stroke-linecap="square" } }
1664 \cs_new_protected:Npn \__draw_backend_join_miter:
1665 { \__draw_backend_scope:n { stroke-linejoin="miter" } }
1666 \cs_new_protected:Npn \__draw_backend_join_round:
1667 { \__draw_backend_scope:n { stroke-linejoin="round" } }
1668 \cs_new_protected:Npn \__draw_backend_join_bevel:
1669 { \__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.

```

1670 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1671 {
1672     \__draw_backend_scope:n
1673     {
1674         transform =
1675         " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1676     }
1677 }

```

(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.

```

1678 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1679 {
1680     \__kernel_backend_scope_begin:
1681     \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1682     \__kernel_backend_literal_svg:n
1683     {
1684         < g~
1685         stroke="none"~
1686         transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1687         >

```

```

1688     }
1689     \box_set_wd:Nn #1 { Opt }
1690     \box_set_ht:Nn #1 { Opt }
1691     \box_set_dp:Nn #1 { Opt }
1692     \box_use:N #1
1693     \__kernel_backend_literal_svg:n { </g> }
1694     \__kernel_backend_scope_end:
1695 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1696 </dvisvgm>
1697 </package>

```

5 l3backend-graphics Implementation

```

1698 <*package>
1699 <@@=graphics>

```

5.1 dvips backend

```

1700 <*dvips>

```

__graphics_backend_getbb_eps:n Simply use the generic function.

```

1701 \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.

```

1702 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1703 {
1704     \__kernel_backend_literal:x
1705     {
1706         PSfile = #1 \c_space_tl
1707         llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1708         lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1709         urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1710         ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1711     }
1712 }

```

(End definition for __graphics_backend_include_eps:n.)

```

1713 </dvips>

```

5.2 LuaTeX and pdfTeX backends

```

1714 <*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.

```

1715 \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.

```

1716 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1717 {
1718   \int_zero:N \l_graphics_page_int
1719   \tl_clear:N \l_graphics_pagebox_tl
1720   \tl_set:Nx \l__graphics_graphics_attr_tl
1721   {
1722     \tl_if_empty:NF \l_graphics_decodearray_tl
1723     { :D \l_graphics_decodearray_tl }
1724     \bool_if:NT \l_graphics_interpolate_bool
1725     { :I }
1726   }
1727   \tl_clear:N \l__graphics_graphics_attr_tl
1728   \_graphics_backend_getbb_auxi:n {#1}
1729 }
1730 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1731 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1732 {
1733   \tl_clear:N \l_graphics_decodearray_tl
1734   \bool_set_false:N \l_graphics_interpolate_bool
1735   \tl_set:Nx \l__graphics_graphics_attr_tl
1736   {
1737     : \l_graphics_pagebox_tl
1738     \int_compare:nNnT \l_graphics_page_int > 1
1739     { :P \int_use:N \l_graphics_page_int }
1740   }
1741   \_graphics_backend_getbb_auxi:n {#1}
1742 }
1743 \cs_new_protected:Npn \_graphics_backend_getbb_auxi:n #1
1744 {
1745   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1746   { \_graphics_backend_getbb_auxii:n {#1} }
1747 }

```

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.

```

1748 \cs_new_protected:Npn \_graphics_backend_getbb_auxii:n #1
1749 {
1750   \tex_immediate:D \tex_pdfximage:D
1751   \bool_lazy_or:nnT
1752   { \l_graphics_interpolate_bool }
1753   { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1754   {
1755     attr ~
1756     {
1757       \tl_if_empty:NF \l_graphics_decodearray_tl
1758       { /Decode~[ \l_graphics_decodearray_tl ] }

```

```

1759         \bool_if:NT \l_graphics_interpolate_bool
1760         { /Interpolate~true }
1761     }
1762 }
1763 \int_compare:nNnT \l_graphics_page_int > 0
1764 { page ~ \int_use:N \l_graphics_page_int }
1765 \tl_if_empty:NF \l_graphics_pagebox_tl
1766 { \l_graphics_pagebox_tl }
1767 {#1}
1768 \hbox_set:Nn \l__graphics_internal_box
1769 { \tex_pdfrefximage:D \tex_pdflastximage:D }
1770 \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1771 \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1772 \int_const:cn { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1773 { \tex_the:D \tex_pdflastximage:D }
1774 \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1775 }

```

(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.

```

1776 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1777 {
1778     \tex_pdfrefximage:D
1779     \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1780 }
1781 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1782 \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.)

_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

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.

```

1783 \sys_if_shell:T
1784 {
1785     \str_new:N \l__graphics_backend_dir_str
1786     \str_new:N \l__graphics_backend_name_str
1787     \str_new:N \l__graphics_backend_ext_str
1788     \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1789     {
1790         \file_parse_full_name:nNNN {#1}
1791         \l__graphics_backend_dir_str
1792         \l__graphics_backend_name_str
1793         \l__graphics_backend_ext_str
1794         \exp_args:Nx \__graphics_backend_getbb_eps:nm
1795         {
1796             \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1797             -converted-to.pdf
1798         }
1799         {#1}

```

```

1800     }
1801     \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1#2
1802     {
1803         \file_compare_timestamp:nNnT {#2} > {#1}
1804         {
1805             \sys_shell_now:n
1806             { repstopdf ~ #2 ~ #1 }
1807         }
1808         \tl_set:Nn \l_graphics_name_tl {#1}
1809         \__graphics_backend_getbb_pdf:n {#1}
1810     }
1811     \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1812     {
1813         \file_parse_full_name:nNNN {#1}
1814         \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1815         \exp_args:Nx \__graphics_backend_include_pdf:n
1816         {
1817             \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1818             -converted-to.pdf
1819         }
1820     }
1821 }

```

(End definition for __graphics_backend_getbb_eps:n and others.)

```
1822 </luatex | pdftex>
```

5.3 dvipdfmx backend

```
1823 < *dvipdfmx | xetex>
```

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

```

\__graphics_backend_getbb_jpg:n 1824 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n
\__graphics_backend_getbb_pdf:n 1825 < *dvipdfmx>
\__graphics_backend_getbb_png:n 1826 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1827 {
1828     \int_zero:N \l_graphics_page_int
1829     \tl_clear:N \l_graphics_pagebox_tl
1830     \graphics_extract_bb:n {#1}
1831 }
1832 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1833 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1834 {
1835     \tl_clear:N \l_graphics_decodearray_tl
1836     \bool_set_false:N \l_graphics_interpolate_bool
1837     \graphics_extract_bb:n {#1}
1838 }
1839 < /dvipdfmx>

```

(End definition for __graphics_backend_getbb_eps:n and others.)

\g__graphics_track_int Used to track the object number associated with each graphic.

```
1840 \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 Xe_{La}TeX: for the latter it is better to use the primitive route. The relevant code for that is included later in this file.

```

1841 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1842 {
1843   \__kernel_backend_literal:x
1844   {
1845     PSfile = #1 \c_space_tl
1846     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1847     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1848     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1849     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1850   }
1851 }
1852 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1853 { \_graphics_backend_include_auxi:nn {#1} { image } }
1854 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1855 <*/dvipdfmx>
1856 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1857 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1858 </dvipdfmx>

```

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.

```

1859 \cs_new_protected:Npn \_graphics_backend_include_auxi:nn #1#2
1860 {
1861   \_graphics_backend_include_auxii:xnn
1862   {
1863     \tl_if_empty:NF \l_graphics_pagebox_tl
1864     { : \l_graphics_pagebox_tl }
1865     \int_compare:nNnT \l_graphics_page_int > 1
1866     { :P \int_use:N \l_graphics_page_int }
1867     \tl_if_empty:NF \l_graphics_decodearray_tl
1868     { :D \l_graphics_decodearray_tl }
1869     \bool_if:NT \l_graphics_interpolate_bool
1870     { :I }
1871   }
1872   {#1} {#2}
1873 }
1874 \cs_new_protected:Npn \_graphics_backend_include_auxii:nnn #1#2#3
1875 {
1876   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1877   {
1878     \__kernel_backend_literal:x
1879     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1880   }
1881   { \_graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1882 }
1883 \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!

```

1884 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1885 {
1886   \int_gincr:N \g__graphics_track_int
1887   \int_const:cn { c__graphics_graphics_ #1#2_int } { \g__graphics_track_int }
1888   \__kernel_backend_literal:x
1889   {
1890     pdf:#3~
1891     @graphic \int_use:c { c__graphics_graphics_ #1#2_int } ~
1892     \int_compare:nNnT \l_graphics_page_int > 1
1893     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1894     \tl_if_empty:NF \l_graphics_pagebox_tl
1895     {
1896       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1897       bbox ~
1898         \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1899         \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1900         \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1901         \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1902     }
1903     (#1)
1904     \bool_lazy_or:nnT
1905     { \l_graphics_interpolate_bool }
1906     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1907     {
1908       <<
1909       \tl_if_empty:NF \l_graphics_decodearray_tl
1910       { /Decode~[ \l_graphics_decodearray_tl ] }
1911       \bool_if:NT \l_graphics_interpolate_bool
1912       { /Interpolate~true> }
1913     }
1914     >>
1915   }
1916 }

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

1917 </dvipdfmx|xetex>

```

5.4 Xe_{La}TeX backend

```

1918 <*xetex>

```

5.4.1 Images

For Xe_{La}TeX, 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 Xe_{La}TeX primitive omits the text box from the page box specification, so there is also some “trimming” to do here.

```

1919 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1920 {
1921   \int_zero:N \l_graphics_page_int
1922   \tl_clear:N \l_graphics_pagebox_tl
1923   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1924 }

```

```

\__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_auxiii:nNnN
\__graphics_backend_getbb_auxiv:nnNnN
\__graphics_backend_getbb_auxiv:NnNnN
\__graphics_backend_getbb_auxv:nNnN
\__graphics_backend_getbb_auxv:NnNnN
\__graphics_backend_getbb_pagebox:w

```

```

1925 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1926 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1927 {
1928   \tl_clear:N \l_graphics_decodearray_tl
1929   \bool_set_false:N \l_graphics_interpolate_bool
1930   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1931 }
1932 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
1933 {
1934   \int_compare:nNnTF \l_graphics_page_int > 1
1935     { \__graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
1936     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1937 }
1938 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
1939 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1940 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
1941 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1942 {
1943   \tl_if_empty:NTF \l_graphics_pagebox_tl
1944     { \__graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1945     { \__graphics_backend_getbb_auxv:nNnn {#1} #2 {#3} {#4} }
1946 }
1947 }
1948 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
1949 {
1950   \use:x
1951   {
1952     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1953     { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1954   }
1955 }
1956 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1957 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1958 {
1959   \graphics_bb_restore:nF {#1#3}
1960   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1961 }
1962 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1963 {
1964   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1965   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1966   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1967   \graphics_bb_save:n {#1#3}
1968 }
1969 \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 Xe_{La}TeX is best done using the `\tex_XeTeXpdffile: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`.

```

1970 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1971 {

```

```

1972 \tex_XeTeXpdffile:D
1973 \__graphics_backend_include_pdf_quote:w #1 "#1" \s__graphics_stop \c_space_tl
1974 \int_compare:nNnT \l_graphics_page_int > 0
1975 { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1976 \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1977 }
1978 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \s__graphics_stop
1979 { " #2 " }

```

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

```

1980 </xetex>

```

5.5 dvisvgm backend

```

1981 <*dvisvgm>

```

__graphics_backend_getbb_eps:n Simply use the generic function.

```

1982 \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

```

1983 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1984 {
1985   \int_zero:N \l_graphics_page_int
1986   \tl_clear:N \l_graphics_pagebox_tl
1987   \graphics_extract_bb:n {#1}
1988 }
1989 \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

```

1990 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1991 {
1992   \tl_clear:N \l_graphics_decodearray_tl
1993   \bool_set_false:N \l_graphics_interpolate_bool
1994   \graphics_extract_bb:n {#1}
1995 }

```

(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

```

1996 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1997 { __graphics_backend_include:nn { PSfile } {#1} }
1998 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1999 { __graphics_backend_include:nn { pdffile } {#1} }
2000 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
2001 {
2002   \__kernel_backend_literal:x
2003   {
2004     #1 = #2 \c_space_tl
2005     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl

```

```

2006         lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2007         urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2008         ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
2009     }
2010 }

```

(End definition for `__graphics_backend_include_eps:n`, `__graphics_backend_include_pdf:n`, and `__graphics_backend_include:nn`.)

```

\__graphics_backend_include_png:n
\__graphics_backend_include_jpg:n
\__graphics_backend_include_bitmap_quote:w

```

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

```

2011 \cs_new_protected:Npn \__graphics_backend_include_png:n #1
2012 {
2013     \__kernel_backend_literal:x
2014     {
2015         dvisvgm:img~
2016         \dim_to_decimal:n { \l_graphics_ury_dim } ~
2017         \dim_to_decimal:n { \l_graphics_ury_dim } ~
2018         \__graphics_backend_include_bitmap_quote:w #1 " #1 " \s__graphics_stop
2019     }
2020 }
2021 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_png:n
2022 \cs_new:Npn \__graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \s__graphics_stop
2023 { " #2 " }

```

(End definition for `__graphics_backend_include_png:n`, `__graphics_backend_include_jpg:n`, and `__graphics_backend_include_bitmap_quote:w`.)

```

2024 </dvisvgm>
2025 </package>

```

6 l3backend-pdf Implementation

```

2026 <*package>
2027 <@@=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

2028 \box_new:N \l__pdf_internal_box

(End definition for \l__pdf_internal_box.)

```

6.2 dvips backend

2029 $\langle *dvips \rangle$

$\backslash_pdf_backend_pdfmark:n$
 $\backslash_pdf_backend_pdfmark:x$

Used often enough it should be a separate function.

2030 $\backslash cs_new_protected:Npn \backslash_pdf_backend_pdfmark:n \#1$
 2031 $\{ \backslash_kernel_backend_postscript:n \{ mark \#1 \sim pdfmark \} \}$
 2032 $\backslash cs_generate_variant:Nn \backslash_pdf_backend_pdfmark:n \{ x \}$

(End definition for $\backslash_pdf_backend_pdfmark:n$.)

6.2.1 Catalogue entries

$\backslash_pdf_backend_catalog_gput:nn$
 $\backslash_pdf_backend_info_gput:nn$

2033 $\backslash cs_new_protected:Npn \backslash_pdf_backend_catalog_gput:nn \#1\#2$
 2034 $\{ \backslash_pdf_backend_pdfmark:n \{ \{ Catalog \} << / \#1 \sim \#2 >> /PUT \} \}$
 2035 $\backslash cs_new_protected:Npn \backslash_pdf_backend_info_gput:nn \#1\#2$
 2036 $\{ \backslash_pdf_backend_pdfmark:n \{ / \#1 \sim \#2 /DOCINFO \} \}$

(End definition for $\backslash_pdf_backend_catalog_gput:nn$ and $\backslash_pdf_backend_info_gput:nn$.)

6.2.2 Objects

$\backslash g_pdf_backend_object_int$
 $\backslash g_pdf_backend_object_prop$

For tracking objects to allow finalisation.

2037 $\backslash int_new:N \backslash g_pdf_backend_object_int$
 2038 $\backslash prop_new:N \backslash g_pdf_backend_object_prop$

(End definition for $\backslash g_pdf_backend_object_int$ and $\backslash g_pdf_backend_object_prop$.)

$\backslash_pdf_backend_object_new:nn$
 $\backslash_pdf_backend_object_ref:n$

Tracking objects is similar to dvipdfmx.

2039 $\backslash cs_new_protected:Npn \backslash_pdf_backend_object_new:nn \#1\#2$
 2040 $\{$
 2041 $\quad \backslash int_gincr:N \backslash g_pdf_backend_object_int$
 2042 $\quad \backslash int_const:cn$
 2043 $\quad \{ c_pdf_backend_object_ \tl_to_str:n \{ \#1 \} _int \}$
 2044 $\quad \{ \backslash g_pdf_backend_object_int \}$
 2045 $\quad \backslash prop_gput:Nnn \backslash g_pdf_backend_object_prop \{ \#1 \} \{ \#2 \}$
 2046 $\}$
 2047 $\backslash cs_new:Npn \backslash_pdf_backend_object_ref:n \#1$
 2048 $\{ \{ pdf.obj \int_use:c \{ c_pdf_backend_object_ \tl_to_str:n \{ \#1 \} _int \} \} \}$

(End definition for $\backslash_pdf_backend_object_new:nn$ and $\backslash_pdf_backend_object_ref:n$.)

$\backslash_pdf_backend_object_write:nn$
 $\backslash_pdf_backend_object_write:nx$
 $\backslash_pdf_backend_object_write_array:nn$
 $\backslash_pdf_backend_object_write_dict:nn$
 $\backslash_pdf_backend_object_write_fstream:nn$
 $\backslash_pdf_backend_object_write_stream:nn$
 $\backslash_pdf_backend_object_write_stream:nnn$

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

2049 $\backslash cs_new_protected:Npn \backslash_pdf_backend_object_write:nn \#1\#2$
 2050 $\{$
 2051 $\quad \backslash_pdf_backend_pdfmark:x$
 2052 $\quad \{$
 2053 $\quad \quad /_objdef \sim \backslash_pdf_backend_object_ref:n \{ \#1 \}$
 2054 $\quad \quad /type$
 2055 $\quad \quad \backslash str_case_e:nn$
 2056 $\quad \quad \{ \backslash prop_item:Nn \backslash g_pdf_backend_object_prop \{ \#1 \} \}$
 2057 $\quad \quad \{$
 2058 $\quad \quad \quad \{ array \} \quad \{ /array \}$
 2059 $\quad \quad \quad \{ dict \} \quad \{ /dict \}$

```

2060         { fstream } { /stream }
2061         { stream } { /stream }
2062     }
2063     /OBJ
2064 }
2065 \use:c
2066 { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2067 { \__pdf_backend_object_ref:n {#1} } {#2}
2068 }
2069 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2070 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2071 {
2072     \__pdf_backend_pdfmark:x
2073     { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2074 }
2075 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2076 {
2077     \__pdf_backend_pdfmark:x
2078     { #1 << \exp_not:n {#2} >> /PUT }
2079 }
2080 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2081 {
2082     \exp_args:Nx
2083     \__pdf_backend_object_write_fstream:nnn {#1} #2
2084 }
2085 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2086 {
2087     \__kernel_backend_postscript:n
2088     {
2089         SDict ~ begin ~
2090         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2091         mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2092         end
2093     }
2094 }
2095 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2096 {
2097     \exp_args:Nx
2098     \__pdf_backend_object_write_stream:nnn {#1} #2
2099 }
2100 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2101 {
2102     \__kernel_backend_postscript:n
2103     {
2104         mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2105         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2106     }
2107 }

```

(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.

```

2108 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2109 {

```

```

2110 \int_gincr:N \g__pdf_backend_object_int
2111 \__pdf_backend_pdfmark:x
2112 {
2113   /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2114   /type
2115   \str_case:nn
2116     {#1}
2117     {
2118       { array } { /array }
2119       { dict } { /dict }
2120       { fstream } { /stream }
2121       { stream } { /stream }
2122     }
2123   /OBJ
2124 }
2125 \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2126 { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2127 }
2128 \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.

```

2129 \cs_new:Npn \__pdf_backend_object_last:
2130 { { 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.

```

2131 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2132 { { 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.

```

2133 \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.

```

2134 \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.

```

2135 \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.

```
2136 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2137 {
2138   \exp_args:Nf \__pdf_backend_annotation_aux:nnnn
2139   { \dim_eval:n {#1} } {#2} {#3} {#4}
2140 }
2141 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2142 {
2143   \box_move_down:nn {#3}
2144   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2145   \box_move_up:nn {#2}
2146   {
2147     \hbox:n
2148     {
2149       \__kernel_kern:n {#1}
2150       \__kernel_backend_postscript:n { pdf.save.ur }
2151       \__kernel_kern:n { -#1 }
2152     }
2153   }
2154   \int_gincr:N \g__pdf_backend_object_int
2155   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2156   \__pdf_backend_pdfmark:x
2157   {
2158     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2159     pdf.rect
2160     #4 ~
2161     /ANN
2162   }
2163 }
```

(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.

```
2164 \cs_new:Npn \__pdf_backend_annotation_last:
2165 { { 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.

```
2166 \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.

```
2167 \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.

```
2168 \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.

```

2169 \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.

```

2170 \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.

```

2171 \tl_new:N \l__pdf_breaklink_pdfmark_tl
2172 \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.

```

2173 \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.

```

2174 \cs_new_eq:NN \__pdf_breaklink_usebox:N \box_use:N
(End definition for \__pdf_breaklink_usebox:N.)

```

`__pdf_backend_link_begin_goto:nnw` 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`.

`__pdf_backend_link_begin_user:nnw`

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

`__pdf_backend_link_aux:nw`

`__pdf_backend_link_end:`

`__pdf_backend_link_end_aux:`

`__pdf_backend_link_minima:` 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.

`__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`

```

2175 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2176 {
2177   \__pdf_backend_link_begin:nw
2178   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2179 }
2180 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2
2181 { \__pdf_backend_link_begin:nw {#1#2} }
2182 \cs_new_protected:Npn \__pdf_backend_link_begin:nw #1
2183 {

```

```

2184 \bool_if:NF \g__pdf_backend_link_bool
2185 { \__pdf_backend_link_begin_aux:nw {#1} }
2186 }

```

The definition of `pdf.link.dict` here is needed as there is code in the PostScript headers for breaking links, and that can only work with this available.

```

2187 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2188 {
2189   \bool_gset_true:N \g__pdf_backend_link_bool
2190   \__kernel_backend_postscript:n
2191   { /pdf.link.dict ( #1 ) def }
2192   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2193   \__pdf_backend_link_sf_save:
2194   \mode_if_math:TF
2195     { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2196     { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2197   \hbox_set:Nw \l__pdf_backend_content_box
2198   \__pdf_backend_link_sf_restore:
2199   \bool_if:NT \g__pdf_backend_link_math_bool
2200     { \c_math_toggle_token }
2201 }
2202 \cs_new_protected:Npn \__pdf_backend_link_end:
2203 {
2204   \bool_if:NT \g__pdf_backend_link_bool
2205     { \__pdf_backend_link_end_aux: }
2206 }
2207 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2208 {
2209   \bool_if:NT \g__pdf_backend_link_math_bool
2210     { \c_math_toggle_token }
2211   \__pdf_backend_link_sf_save:
2212   \hbox_set_end:
2213   \__pdf_backend_link_minima:
2214   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2215   \exp_args:Nx \__pdf_backend_link_outerbox:n
2216   {
2217     \int_if_odd:nTF { \value { page } }
2218       { \oddsidemargin }
2219       { \evensidemargin }
2220   }
2221   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2222     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2223   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2224   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2225   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2226   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2227     {
2228       \hbox:n
2229         { \__kernel_backend_postscript:n { pdf.save.linkur } }
2230     }
2231   \int_gincr:N \g__pdf_backend_object_int
2232   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2233   \__kernel_backend_postscript:x
2234   {

```

```

2235     mark
2236     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2237     \g__pdf_backend_link_dict_tl \c_space_tl
2238     pdf.rect
2239     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2240   }
2241   \__pdf_backend_link_sf_restore:
2242   \bool_gset_false:N \g__pdf_backend_link_bool
2243 }
2244 \cs_new_protected:Npn \__pdf_backend_link_minima:
2245 {
2246   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2247   \__kernel_backend_postscript:x
2248   {
2249     /pdf.linkdp.pad ~
2250     \dim_to_decimal:n
2251     {
2252       \dim_max:nn
2253       {
2254         \box_dp:N \l__pdf_backend_model_box
2255         - \box_dp:N \l__pdf_backend_content_box
2256       }
2257       { Opt }
2258     } ~
2259     pdf.pt.dvi ~ def
2260   /pdf.linkht.pad ~
2261   \dim_to_decimal:n
2262   {
2263     \dim_max:nn
2264     {
2265       \box_ht:N \l__pdf_backend_model_box
2266       - \box_ht:N \l__pdf_backend_content_box
2267     }
2268     { Opt }
2269   } ~
2270   pdf.pt.dvi ~ def
2271 }
2272 }
2273 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2274 {
2275   \__kernel_backend_postscript:x
2276   {
2277     /pdf.outerbox
2278     [
2279       \dim_to_decimal:n {#1} ~
2280       \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2281       \dim_to_decimal:n { #1 + \textwidth } ~
2282       \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2283     ]
2284     [ exch { pdf.pt.dvi } forall ] def
2285   /pdf.baselineskip ~
2286   \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2287   { pdf.pt.dvi ~ def }
2288   { pop ~ pop }

```

```

2289         ifelse
2290     }
2291 }
2292 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2293 {
2294     \int_gset:Nn \g__pdf_backend_link_sf_int
2295     {
2296         \mode_if_horizontal:TF
2297         { \tex_spacefactor:D }
2298         { 0 }
2299     }
2300 }
2301 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2302 {
2303     \mode_if_horizontal:T
2304     {
2305         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2306         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2307     }
2308 }

```

(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.

```

2309 \use_none:n
2310 {
2311     \cs_if_exist:NT \@makecol@hook
2312     {
2313         \tl_put_right:Nn \@makecol@hook
2314         {
2315             \box_if_empty:NF \@cclv
2316             {
2317                 \vbox_set:Nn \@cclv
2318                 {
2319                     \__kernel_backend_postscript:n
2320                     {
2321                         pdf.globaldict /pdf.brokenlink.rect ~ known
2322                         { pdf.bordertracking.continue }
2323                         if
2324                     }
2325                     \vbox_unpack_drop:N \@cclv
2326                     \__kernel_backend_postscript:n
2327                     { pdf.bordertracking.endpage }
2328                 }
2329             }
2330         }
2331         \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2332         \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2333         \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2334     }
2335 }

```

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

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

```
2336 \cs_new:Npn \_pdf_backend_link_last:
2337 { { 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.

```
2338 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2339 {
2340   \__kernel_backend_postscript:x
2341   {
2342     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2343   }
2344 }
```

(End definition for `_pdf_backend_link_margin:n`.)

`_pdf_backend_destination:nn` 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. `ftr` without rule spec doesn't work, so it falls back to `/Fit` here.

`_pdf_backend_destination:nnnn`
`_pdf_backend_destination_aux:nnnn`

```
2345 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2346 {
2347   \__kernel_backend_postscript:n { pdf.dest.anchor }
2348   \_pdf_backend_pdfmark:x
2349   {
2350     /View
2351     [
2352       \str_case:nnF {#2}
2353       {
2354         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2355         { fit } { /Fit }
2356         { fitb } { /FitB }
2357         { fitbh } { /FitBH ~ pdf.dest.y }
2358         { fitbv } { /FitBV ~ pdf.dest.x }
2359         { fith } { /FitH ~ pdf.dest.y }
2360         { fitv } { /FitV ~ pdf.dest.x }
2361         { fitr } { /Fit }
2362       }
2363       {
2364         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2365       }
2366     ]
2367     /Dest ( \exp_not:n {#1} ) cvn
2368     /DEST
2369   }
2370 }
2371 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2372 {
2373   \exp_args:Ne \_pdf_backend_destination_aux:nnnn
2374   { \dim_eval:n {#2} } {#1} {#3} {#4}
2375 }
```

```

2376 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2377 {
2378   \vbox_to_zero:n
2379   {
2380     \__kernel_kern:n {#4}
2381     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2382     \tex_vss:D
2383   }
2384   \__kernel_kern:n {#1}
2385   \vbox_to_zero:n
2386   {
2387     \__kernel_kern:n { -#3 }
2388     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2389     \tex_vss:D
2390   }
2391   \__kernel_kern:n { -#1 }
2392   \__pdf_backend_pdfmark:n
2393   {
2394     /View
2395     [
2396       /FitR ~
2397       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2398       pdf.urx ~ pdf.ury ~ pdf.dest2device
2399     ]
2400     /Dest ( #2 ) cvn
2401     /DEST
2402   }
2403 }

```

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

6.2.4 Structure

Doable for the usual ps2pdf method.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2404 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2405 {
2406   \int_compare:nNnT {#1} = 0
2407   {
2408     \__kernel_backend_literal_postscript:n
2409     {
2410       /setdistillerparams ~ where
2411       { pop << /CompressPages ~ false >> setdistillerparams }
2412       if
2413     }
2414   }
2415 }
2416 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2417 {
2418   \bool_if:nF {#1}
2419   {
2420     \__kernel_backend_literal_postscript:n
2421     {
2422       /setdistillerparams ~ where

```

```

2423         { pop << /CompressStreams ~ false >> setdistillerparams }
2424     if
2425     }
2426 }
2427 }

```

(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`

```

2428 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2429 {
2430     \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2431 }
2432 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2433 {
2434     \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2435 }

```

(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:`

```

2436 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2437 \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`

Simple wrappers.

`_pdf_backend_emc:`

```

2438 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2439 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2440 \cs_new_protected:Npn \_pdf_backend_emc:
2441 { \_pdf_backend_pdfmark:n { /EMC } }

```

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

2442 \langle /dvips \rangle

6.3 LuaTeX and pdfTeX backend

2443 \langle *luatex | pdftex \rangle

6.3.1 Annotations

`_pdf_backend_annotation:nnnn`

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

```

2444 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2445 {
2446      $\langle$ *luatex $\rangle$ 
2447     \tex_pdfextension:D annot ~
2448      $\langle$ /luatex $\rangle$ 
2449      $\langle$ *pdftex $\rangle$ 
2450     \tex_pdfannot:D
2451      $\langle$ /pdftex $\rangle$ 
2452     width ~ \dim_eval:n {#1} ~
2453     height ~ \dim_eval:n {#2} ~
2454     depth ~ \dim_eval:n {#3} ~

```



```

2455     {#4}
2456   }

```

(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.

```

2457 \cs_new:Npx \_pdf_backend_annotation_last:
2458 {
2459   \exp_not:N \int_value:w
2460 <*luatex>
2461   \exp_not:N \tex_pdffeedback:D lastannot ~
2462 </luatex>
2463 <*pdftex>
2464   \exp_not:N \tex_pdflastannot:D
2465 </pdftex>
2466   \c_space_tl 0 ~ R
2467 }

```

(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 2468 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
\_pdf_backend_link_begin:nnnw      2469 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
\_pdf_backend_link_end:             2470 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2471 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2472 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2473 {
2474 <*luatex>
2475   \tex_pdfextension:D startlink ~
2476 </luatex>
2477 <*pdftex>
2478   \tex_pdfstartlink:D
2479 </pdftex>
2480   attr {#1}
2481   #2 {#3}
2482 }
2483 \cs_new_protected:Npn \_pdf_backend_link_end:
2484 {
2485 <*luatex>
2486   \tex_pdfextension:D endlink \scan_stop:
2487 </luatex>
2488 <*pdftex>
2489   \tex_pdfendlink:D
2490 </pdftex>
2491 }

```

(End definition for `_pdf_backend_link_begin_goto:nnw` and others.)

`_pdf_backend_link_last:` Formatted for direct use.

```

2492 \cs_new:Npx \_pdf_backend_link_last:
2493 {
2494   \exp_not:N \int_value:w
2495 <*luatex>

```

```

2496 \exp_not:N \tex_pdffeedback:D lastlink ~
2497 </luatex>
2498 <*pdftex>
2499 \exp_not:N \tex_pdflastlink:D
2500 </pdftex>
2501 \c_space_tl 0 ~ R
2502 }

```

(End definition for _pdf_backend_link_last:.)

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

```

2503 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2504 {
2505 <*luatex>
2506 \tex_pdfvariable:D linkmargin
2507 </luatex>
2508 <*pdftex>
2509 \tex_pdflinkmargin:D
2510 </pdftex>
2511 \dim_eval:n {#1} \scan_stop:
2512 }

```

(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

```

2513 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2514 {
2515 <*luatex>
2516 \tex_pdfextension:D dest ~
2517 </luatex>
2518 <*pdftex>
2519 \tex_pdfdest:D
2520 </pdftex>
2521 name {#1}
2522 \str_case:nnF {#2}
2523 {
2524 { xyz } { xyz }
2525 { fit } { fit }
2526 { fitb } { fitb }
2527 { fitbh } { fitbh }
2528 { fitbv } { fitbv }
2529 { fith } { fith }
2530 { fitv } { fitv }
2531 { fitr } { fitr }
2532 }
2533 { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2534 \scan_stop:
2535 }
2536 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2537 {
2538 <*luatex>
2539 \tex_pdfextension:D dest ~

```

```

2540 </luatex>
2541 <*pdftex>
2542   \tex_pdfdest:D
2543 </pdftex>
2544   name {#1}
2545   fitr ~
2546   width \dim_eval:n {#2} ~
2547   height \dim_eval:n {#3} ~
2548   depth \dim_eval:n {#4} \scan_stop:
2549 }

(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
2550 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2551 {
2552 <*luatex>
2553   \tex_pdfextension:D catalog
2554 </luatex>
2555 <*pdftex>
2556   \tex_pdfcatalog:D
2557 </pdftex>
2558   { / #1 ~ #2 }
2559 }
2560 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2561 {
2562 <*luatex>
2563   \tex_pdfextension:D info
2564 </luatex>
2565 <*pdftex>
2566   \tex_pdfinfo:D
2567 </pdftex>
2568   { / #1 ~ #2 }
2569 }

(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.
2570 \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
2571 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2
2572 {
2573 <*luatex>
2574   \tex_pdfextension:D obj ~
2575 </luatex>
2576 <*pdftex>
2577   \tex_pdfobj:D

```

```

2578 </pdftex>
2579     reserveobjnum ~
2580     \int_const:cn
2581     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2582 <*luatex>
2583     { \tex_pdffeedback:D lastobj }
2584 </luatex>
2585 <*pdftex>
2586     { \tex_pdflastobj:D }
2587 </pdftex>
2588     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2589 }
2590 \cs_new:Npn \__pdf_backend_object_ref:n #1
2591 { \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:nn 2592 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
\__pdf_backend_object_write:nx 2593 {
\__pdf_exp_not_i:nn 2594 <*luatex>
\__pdf_exp_not_ii:nn 2595     \tex_immediate:D \tex_pdfextension:D obj ~
2596 </luatex>
2597 <*pdftex>
2598     \tex_immediate:D \tex_pdfobj:D
2599 </pdftex>
2600     useobjnum ~
2601     \int_use:c
2602     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2603     \str_case:e:nn
2604     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2605     {
2606         { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2607         { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2608         { fstream }
2609         {
2610             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2611             file ~ { \__pdf_exp_not_ii:nn #2 }
2612         }
2613         { stream }
2614         {
2615             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2616             { \__pdf_exp_not_ii:nn #2 }
2617         }
2618     }
2619 }
2620 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2621 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2622 \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 2623 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2

```

```

2624 {
2625 <*luatex>
2626   \tex_immediate:D \tex_pdfextension:D obj ~
2627 </luatex>
2628 <*pdftex>
2629   \tex_immediate:D \tex_pdfobj:D
2630 </pdftex>
2631   \str_case:nn
2632     {#1}
2633   {
2634     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2635     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2636     { fstream }
2637     {
2638       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2639       file ~ { \__pdf_exp_not_ii:nn #2 }
2640     }
2641     { stream }
2642     {
2643       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2644       { \__pdf_exp_not_ii:nn #2 }
2645     }
2646   }
2647 }
2648 \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.

```

2649 \cs_new:Npx \__pdf_backend_object_last:
2650 {
2651   \exp_not:N \int_value:w
2652 <*luatex>
2653   \exp_not:N \tex_pdffeedback:D lastobj ~
2654 </luatex>
2655 <*pdftex>
2656   \exp_not:N \tex_pdflastobj:D
2657 </pdftex>
2658   \c_space_tl 0 ~ R
2659 }

```

(End definition for __pdf_backend_object_last:.)

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

```

2660 \cs_new:Npx \__pdf_backend_pageobject_ref:n #1
2661 {
2662   \exp_not:N \int_value:w
2663 <*luatex>
2664   \exp_not:N \tex_pdffeedback:D pageref
2665 </luatex>
2666 <*pdftex>
2667   \exp_not:N \tex_pdfpageref:D
2668 </pdftex>
2669   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2670 }

```

(End definition for `_pdf_backend_pageobject_ref:n`.)

6.3.4 Structure

Simply pass data to the engine.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
\__pdf_backend_objcompresslevel:n
2671 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2672 {
2673   \tex_global:D
2674   \*luatex
2675   \tex_pdfvariable:D compresslevel
2676   \*luatex
2677   \*pdftex
2678   \tex_pdfcompresslevel:D
2679   \*pdftex
2680   \int_value:w \int_eval:n {#1} \scan_stop:
2681 }
2682 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2683 {
2684   \bool_if:nTF {#1}
2685   { \__pdf_backend_objcompresslevel:n { 2 } }
2686   { \__pdf_backend_objcompresslevel:n { 0 } }
2687 }
2688 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2689 {
2690   \tex_global:D
2691   \*luatex
2692   \tex_pdfvariable:D objcompresslevel
2693   \*luatex
2694   \*pdftex
2695   \tex_pdfobjcompresslevel:D
2696   \*pdftex
2697   #1 \scan_stop:
2698 }

```

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

`_pdf_backend_version_major_gset:n`
`_pdf_backend_version_minor_gset:n`

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

```

2699 \cs_new_protected:Npx \__pdf_backend_version_major_gset:n #1
2700 {
2701   \*luatex
2702   \int_compare:nNnT \tex_luatexversion:D > { 106 }
2703   {
2704     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2705     \exp_not:N \int_eval:n {#1} \scan_stop:
2706   }
2707   \*luatex
2708   \*pdftex
2709   \cs_if_exist:NT \tex_pdfmajorversion:D
2710   {
2711     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2712     \exp_not:N \int_eval:n {#1} \scan_stop:
2713   }
2714   \*pdftex

```

```

2715 }
2716 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2717 {
2718   \tex_global:D
2719   \*luatex
2720   \tex_pdfvariable:D minorversion
2721   \*luatex
2722   \*pdftex
2723   \tex_pdfminorversion:D
2724   \*pdftex
2725   \int_eval:n {#1} \scan_stop:
2726 }

```

(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: 2727 \cs_new:Npx \__pdf_backend_version_major:
2728 {
2729   \*luatex
2730   \int_compare:nNnTF \tex luatexversion:D > { 106 }
2731   { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2732   { 1 }
2733   \*luatex
2734   \*pdftex
2735   \cs_if_exist:NTF \tex_pdfmajorversion:D
2736   { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2737   { 1 }
2738   \*pdftex
2739 }
2740 \cs_new:Npn \__pdf_backend_version_minor:
2741 {
2742   \tex_the:D
2743   \*luatex
2744   \tex_pdfvariable:D minorversion
2745   \*luatex
2746   \*pdftex
2747   \tex_pdfminorversion:D
2748   \*pdftex
2749 }

```

(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:

```

2750 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2751 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2752 \cs_new_protected:Npn \__pdf_backend_emc:
2753 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2754 \*luatex | \*pdftex

```

6.4 dvipdfmx backend

2755 `*dvipdfmx|xetex`

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

`__pdf_backend:x`

```

2756 \cs_new_protected:Npx \__pdf_backend:n #1
2757 { \__kernel_backend_literal:n { pdf: #1 } }
2758 \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
2759 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2760 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2761 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2762 { \__pdf_backend:n { docinfo << /#1 ~ #2 >> } }

```

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

6.4.2 Objects

`\g__pdf_backend_object_int` For tracking objects to allow finalisation.

`\g__pdf_backend_object_prop`

```

2763 \int_new:N \g__pdf_backend_object_int
2764 \prop_new:N \g__pdf_backend_object_prop

```

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

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

`__pdf_backend_object_ref:n`

```

2765 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2766 {
2767   \int_gincr:N \g__pdf_backend_object_int
2768   \int_const:cn
2769   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2770   { \g__pdf_backend_object_int }
2771   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2772 }
2773 \cs_new:Npn \__pdf_backend_object_ref:n #1
2774 { @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` This is where we choose the actual type.

```

\__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
2775 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2776 {
2777   \exp_args:Nx \__pdf_backend_object_write:nnn
2778   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2779 }
2780 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2781 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2782 {
2783   \use:c { \__pdf_backend_object_write_ #1 :nn }
2784   { \__pdf_backend_object_ref:n {#2} } {#3}
2785 }

```



```

2786 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2787 {
2788   \__pdf_backend:x
2789   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2790 }
2791 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2792 {
2793   \__pdf_backend:x
2794   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2795 }
2796 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2797 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2798 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2799 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2800 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2801 {
2802   \__pdf_backend:x
2803   {
2804     #1 stream ~ #2 ~
2805     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2806   }
2807 }

```

(End definition for __pdf_backend_object_write:nn and others.)

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

```

\__pdf_backend_object_now:nx
2808 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2809 {
2810   \int_gincr:N \g__pdf_backend_object_int
2811   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2812   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2813   {#2}
2814 }
2815 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2816 \cs_new:Npn \__pdf_backend_object_last:
2817 { @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_TTeX.

```

2818 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2819 { @page #1 }

```

(End definition for _pdf_backend_pageobject_ref:n.)

6.4.3 Annotations

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

```
2820 \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.

```
2821 \cs_new_protected:Npn \_pdf_backend_annotation: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`.)

`_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 }
```

```

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

```

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

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

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
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 We start with the assumption that the default is active.

_pdf_backend_version_minor_gset:n

```

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

_pdf_backend_version_minor:

```

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 No-op.

__pdf_backend_info_gput:nn

```

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 All no-ops here.

__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

```

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
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
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:
\__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
\__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>

```

No stack so set values directly. The need to deal with Distiller and Ghostscript separately means we use a common auxiliary: the two systems require different PostScript for transparency. This is of course not quite as efficient as doing one test for setting all transparency, but it keeps things clearer here. Thanks to Alex Grahn for the detail on testing for GhostScript.

```

\__opacity_backend_select:n
\__opacity_backend_select_aux:n
\__opacity_backend_fill:n
\__opacity_backend_stroke:n
\__opacity_backend:nnn
\__opacity_backend:xnn
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   \__opacity_backend:nnn {#1} { fill } { ca }

```

```

2990   \_opacity_backend:nnn {#1} { stroke } { CA }
2991 }
2992 \cs_new_protected:Npn \_opacity_backend_fill:n #1
2993 {
2994   \_opacity_backend:xnn
2995   { \fp_eval:n { min(max(0,#1),1) } }
2996   { fill }
2997   { ca }
2998 }
2999 \cs_new_protected:Npn \_opacity_backend_stroke:n #1
3000 {
3001   \_opacity_backend:xnn
3002   { \fp_eval:n { min(max(0,#1),1) } }
3003   { stroke }
3004   { CA }
3005 }
3006 \cs_new_protected:Npn \_opacity_backend:nnn #1#2#3
3007 {
3008   \_kernel_backend_postscript:n
3009   {
3010     product ~ (Ghostscript) ~ search
3011     {
3012       pop ~ pop ~ pop ~
3013       #1 ~ .set #2 constantalpha
3014     }
3015     {
3016       pop ~
3017       mark ~
3018       /#3 ~ #1
3019       /SetTransparency ~
3020       pdfmark
3021     }
3022     ifelse
3023   }
3024 }
3025 \cs_generate_variant:Nn \_opacity_backend:nnn { x }

```

(End definition for _opacity_backend_select:n and others.)

```

3026 </dvips>
3027 <*dvipdfmx | luatex | pdftex | xetex>

```

\c_opacity_backend_stack_int Set up a stack.

```

3028 \bool_lazy_and:nnT
3029 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3030 { \pdfmanagement_if_active_p:}
3031 {
3032   \_kernel_color_backend_stack_init:Nnn \c_opacity_backend_stack_int
3033   { page ~ direct } { /opacity 1 ~ gs }
3034   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3035   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3036 }

```

(End definition for \c_opacity_backend_stack_int.)

`\l__opacity_backend_fill_tl` We use `tl` here for speed: at the backend, this should be reasonable.

`\l__opacity_backend_stroke_tl`

```

3037 \tl_new:N \l__opacity_backend_fill_tl
3038 \tl_new:N \l__opacity_backend_stroke_tl

```

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

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

`__opacity_backend_select_aux:n`

`__opacity_backend_reset:`

```

3039 \cs_new_protected:Npn \__opacity_backend_select:n #1
3040 {
3041   \exp_args:Nx \__opacity_backend_select_aux:n
3042   { \fp_eval:n { min(max(0,#1),1) } }
3043 }
3044 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3045 {
3046   \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3047   \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3048   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3049   { opacity #1 }
3050   { << /ca ~ #1 /CA ~ #1 >> }
3051   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3052   { /opacity #1 ~ gs }
3053   \group_insert_after:N \__opacity_backend_reset:
3054 }
3055 \bool_lazy_and:nnF
3056 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3057 { \pdfmanagement_if_active_p:}
3058 {
3059   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1 { }
3060 }
3061 \cs_new_protected:Npn \__opacity_backend_reset:
3062 { \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int }

```

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

`__opacity_backend_fill:n` For separate fill and stroke, we need to work out if we need to do more work or if we can

`__opacity_backend_stroke:n` stick to a single setting.

`__opacity_backend_fillstroke:nn`

`__opacity_backend_fillstroke:xx`

```

3063 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3064 {
3065   \__opacity_backend_fill_stroke:xx
3066   { \fp_eval:n { min(max(0,#1),1) } }
3067   \l__opacity_backend_stroke_tl
3068 }
3069 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3070 {
3071   \__opacity_backend_fill_stroke:xx
3072   \l__opacity_backend_fill_tl
3073   { \fp_eval:n { min(max(0,#1),1) } }
3074 }
3075 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3076 {
3077   \str_if_eq:nnTF {#1} {#2}
3078   { \__opacity_backend_select_aux:n {#1} }
3079   {

```



```

3080 \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3081 \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3082 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3083 { opacity.fill #1 }
3084 { << /ca ~ #1 >> }
3085 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3086 { opacity.stroke #1 }
3087 { << /CA ~ #2 >> }
3088 \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3089 { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3090 \group_insert_after:N \__opacity_backend_reset:
3091 }
3092 }
3093 \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.)

3094 </dvipdfmx | luatex | pdftex | xetex>
3095 <*dvipdfmx | xdvipdfmx>

```

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

```

3096 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
3097 {
3098   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1
3099   {
3100     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3101     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3102     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3103     { opacity #1 }
3104     { << /ca ~ #1 /CA ~ #1 >> }
3105     \__kernel_backend_literal_pdf:n { /opacity #1 ~ gs }
3106   }
3107   \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3108   {
3109     \str_if_eq:nnTF {#1} {#2}
3110     { \__opacity_backend_select_aux:n {#1} }
3111     {
3112       \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3113       \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3114       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3115       { opacity.fill #1 }
3116       { << /ca ~ #1 >> }
3117       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3118       { opacity.stroke #1 }
3119       { << /CA ~ #2 >> }
3120       \__kernel_backend_literal_pdf:n
3121       { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3122     }
3123   }
3124 }

```

(End definition for __opacity_backend_select:n.)

```

3125 </dvipdfmx | xdvipdfmx>

```

```
3126 <*dvisvgm>
```

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

```
3127 \cs_new_protected:Npn \__opacity_backend_select:n #1
3128   { \__opacity_backend:nn {#1} { } }
3129 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3130   { \__opacity_backend:nn {#1} { fill- } }
3131 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3132   { \__opacity_backend:nn { {#1} } { stroke- } }
3133 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3134   { \__kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

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

3135 </dvisvgm>

3136 </package>
```

8 l3backend-header Implementation

```
3137 <*dvips & header>
```

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

```
3138 /color.sc { } def
```

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

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

```
3139 TeXDict begin
3140 /TeXcolorseparation { setcolor } def
3141 end
```

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

pdf.globaldict A small global dictionary for backend use.

```
3142 true setglobal
3143 /pdf.globaldict 4 dict def
3144 false setglobal
```

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

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

```
3145 /pdf.cvs { 65534 string cvs } def
3146 /pdf.dvi.pt { 72.27 mul Resolution div } def
3147 /pdf.pt.dvi { 72.27 div Resolution mul } def
3148 /pdf.rect.ht { dup 1 get neg exch 3 get add } def
```

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

```

pdf.linkmargin Settings which are defined up-front in SDict.
pdf.linkdp.pad 3149 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad 3150 /pdf.linkdp.pad { 0 } def
               3151 /pdf.linkht.pad { 0 } def

(End definition for pdf.linkmargin, pdf.linkdp.pad, and pdf.linkht.pad. These functions are docu-
mented on page ??.)

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 3152 /pdf.rect
pdf.save.linkur 3153 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.llx 3154 /pdf.save.ll
pdf.lly 3155 {
pdf.urx 3156     currentpoint
pdf.ury 3157     /pdf.lly exch def
               3158     /pdf.llx exch def
               3159 }
               3160 def
               3161 /pdf.save.ur
               3162 {
               3163     currentpoint
               3164     /pdf.ury exch def
               3165     /pdf.urx exch def
               3166 }
               3167 def
               3168 /pdf.save.linkll
               3169 {
               3170     currentpoint
               3171     pdf.linkmargin add
               3172     pdf.linkdp.pad add
               3173     /pdf.lly exch def
               3174     pdf.linkmargin sub
               3175     /pdf.llx exch def
               3176 }
               3177 def
               3178 /pdf.save.linkur
               3179 {
               3180     currentpoint
               3181     pdf.linkmargin sub
               3182     pdf.linkht.pad sub
               3183     /pdf.ury exch def
               3184     pdf.linkmargin add
               3185     /pdf.urx exch def
               3186 }
               3187 def

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

pdf.dest.anchor For finding the anchor point of a destination link. We make the use case a separate
pdf.dest.x function as it comes up a lot, and as this makes it easier to adjust if we need additional
pdf.dest.y effects. We also need a more complex approach to convert a co-ordinate pair correctly
pdf.dest.point
pdf.dest2device
pdf.dev.x
pdf.dev.y
pdf.tmpa
pdf.tmpb
pdf.tmpc
pdf.tmpd

```

when defining a rectangle: this can otherwise be out when using a landscape page.
(Thanks to Alexander Grahn for the approach here.)

```

3188 /pdf.dest.anchor
3189 {
3190     currentpoint exch
3191     pdf.dvi.pt 72 add
3192     /pdf.dest.x exch def
3193     pdf.dvi.pt
3194     vsize 72 sub exch sub
3195     /pdf.dest.y exch def
3196 }
3197 def
3198 /pdf.dest.point
3199 { pdf.dest.x pdf.dest.y } def
3200 /pdf.dest2device
3201 {
3202     /pdf.dest.y exch def
3203     /pdf.dest.x exch def
3204     matrix currentmatrix
3205     matrix defaultmatrix
3206     matrix invertmatrix
3207     matrix concatmatrix
3208     cvx exec
3209     /pdf.dev.y exch def
3210     /pdf.dev.x exch def
3211     /pdf.tmpd exch def
3212     /pdf.tmpc exch def
3213     /pdf.tmpb exch def
3214     /pdf.tmpa exch def
3215     pdf.dest.x pdf.tmpa mul
3216     pdf.dest.y pdf.tmpc mul add
3217     pdf.dev.x add
3218     pdf.dest.x pdf.tmpb mul
3219     pdf.dest.y pdf.tmpd mul add
3220     pdf.dev.y add
3221 }
3222 def

```

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

pdf.bordertracking	To know where a breakable link can go, we need to track the boundary rectangle. That
pdf.bordertracking.begin	can be done by hooking into a and x operations: those names have to be retained. The
pdf.bordertracking.end	boundary is stored at the end of the operation. Special effort is needed at the start and
pdf.leftboundary	end of pages (or rather galleys), such that everything works properly.
pdf.rightboundary	
pdf.brokenlink.rect	3223 /pdf.bordertracking false def
pdf.brokenlink.skip	3224 /pdf.bordertracking.begin
pdf.brokenlink.dict	3225 {
pdf.bordertracking.endpage	3226 SDict /pdf.bordertracking true put
pdf.bordertracking.continue	3227 SDict /pdf.leftboundary undef
pdf.originx	3228 SDict /pdf.rightboundary undef
pdf.originy	3229 /a where
	3230 {
	3231 /a
	3232 {

```

3233         currentpoint pop
3234         SDict /pdf.rightboundary known dup
3235         {
3236             SDict /pdf.rightboundary get 2 index lt
3237             { not }
3238             if
3239         }
3240         if
3241         { pop }
3242         { SDict exch /pdf.rightboundary exch put }
3243         ifelse
3244         moveto
3245         currentpoint pop
3246         SDict /pdf.leftboundary known dup
3247         {
3248             SDict /pdf.leftboundary get 2 index gt
3249             { not }
3250             if
3251         }
3252         if
3253         { pop }
3254         { SDict exch /pdf.leftboundary exch put }
3255         ifelse
3256     }
3257     put
3258 }
3259 if
3260 }
3261 def
3262 /pdf.bordertracking.end
3263 {
3264     /a where { /a { moveto } put } if
3265     /x where { /x { 0 exch rmoveto } put } if
3266     SDict /pdf.leftboundary known
3267     { pdf.outerbox 0 pdf.leftboundary put }
3268     if
3269     SDict /pdf.rightboundary known
3270     { pdf.outerbox 2 pdf.rightboundary put }
3271     if
3272     SDict /pdf.bordertracking false put
3273 }
3274 def
3275 /pdf.bordertracking.endpage
3276 {
3277     pdf.bordertracking
3278     {
3279         pdf.bordertracking.end
3280         true setglobal
3281         pdf.globaldict
3282         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3283         pdf.globaldict
3284         /pdf.brokenlink.skip pdf.baselineskip put
3285         pdf.globaldict
3286         /pdf.brokenlink.dict

```

```

3287         pdf.link.dict pdf.cvs put
3288     false setglobal
3289     mark pdf.link.dict cvx exec /Rect
3290     [
3291         pdf.llx
3292         pdf.lly
3293         pdf.outerbox 2 get pdf.linkmargin add
3294         currentpoint exch pop
3295         pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3296     ]
3297     /ANN pdf.pdfmark
3298 }
3299 if
3300 }
3301 def
3302 /pdf.bordertracking.continue
3303 {
3304     /pdf.link.dict pdf.globaldict
3305     /pdf.brokenlink.dict get def
3306     /pdf.outerbox pdf.globaldict
3307     /pdf.brokenlink.rect get def
3308     /pdf.baselineskip pdf.globaldict
3309     /pdf.brokenlink.skip get def
3310     pdf.globaldict dup dup
3311     /pdf.brokenlink.dict undef
3312     /pdf.brokenlink.skip undef
3313     /pdf.brokenlink.rect undef
3314     currentpoint
3315     /pdf.originy exch def
3316     /pdf.originx exch def
3317     /a where
3318     {
3319         /a
3320         {
3321             moveto
3322             SDict
3323             begin
3324                 currentpoint pdf.originy ne exch
3325                 pdf.originx ne or
3326                 {
3327                     pdf.save.linkll
3328                     /pdf.lly
3329                     pdf.lly pdf.outerbox 1 get sub def
3330                     pdf.bordertracking.begin
3331                 }
3332                 if
3333                 end
3334             }
3335             put
3336         }
3337     if
3338     /x where
3339     {
3340         /x

```

```

3341     {
3342         0 exch rmoveto
3343         SDict
3344         begin
3345         currentpoint
3346         pdf.originy ne exch pdf.originx ne or
3347         {
3348             pdf.save.linkll
3349             /pdf.lly
3350             pdf.lly pdf.outerbox 1 get sub def
3351             pdf.bordertracking.begin
3352         }
3353         if
3354         end
3355     }
3356     put
3357 }
3358 if
3359 }
3360 def

```

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

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

```

3361 /pdf.breaklink
3362 {
3363     pop
3364     counttomark 2 mod 0 eq
3365     {
3366         counttomark /pdf.count exch def
3367         {
3368             pdf.count 0 eq { exit } if
3369             counttomark 2 roll
3370             1 index /Rect eq
3371             {
3372                 dup 4 array copy
3373                 dup dup
3374                 1 get
3375                 pdf.outerbox pdf.rect.ht
3376                 pdf.linkmargin 2 mul add sub
3377                 3 exch put
3378                 dup
3379                 pdf.outerbox 2 get
3380                 pdf.linkmargin add
3381                 2 exch put
3382                 dup dup
3383                 3 get
3384                 pdf.outerbox pdf.rect.ht
3385                 pdf.linkmargin 2 mul add add
3386                 1 exch put

```

```

3387     /pdf.currentrect exch def
3388     pdf.breaklink.write
3389     {
3390         pdf.currentrect
3391         dup
3392         pdf.outerbox 0 get
3393         pdf.linkmargin sub
3394         0 exch put
3395         dup
3396         pdf.outerbox 2 get
3397         pdf.linkmargin add
3398         2 exch put
3399         dup dup
3400         1 get
3401         pdf.baselineskip add
3402         1 exch put
3403         dup dup
3404         3 get
3405         pdf.baselineskip add
3406         3 exch put
3407         /pdf.currentrect exch def
3408         pdf.breaklink.write
3409     }
3410     1 index 3 get
3411     pdf.linkmargin 2 mul add
3412     pdf.outerbox pdf.rect.ht add
3413     2 index 1 get sub
3414     pdf.baselineskip div round cvi 1 sub
3415     exch
3416     repeat
3417     pdf.currentrect
3418     dup
3419     pdf.outerbox 0 get
3420     pdf.linkmargin sub
3421     0 exch put
3422     dup dup
3423     1 get
3424     pdf.baselineskip add
3425     1 exch put
3426     dup dup
3427     3 get
3428     pdf.baselineskip add
3429     3 exch put
3430     dup 2 index 2 get 2 exch put
3431     /pdf.currentrect exch def
3432     pdf.breaklink.write
3433     SDict /pdf.pdfmark.good false put
3434     exit
3435 }
3436 { pdf.count 2 sub /pdf.count exch def }
3437 ifelse
3438 }
3439 loop
3440 }

```



```

3441   if
3442   /ANN
3443 }
3444 def
3445 /pdf.breaklink.write
3446 {
3447   counttomark 1 sub
3448   index /_objdef eq
3449   {
3450     counttomark -2 roll
3451     dup wcheck
3452     {
3453       readonly
3454       counttomark 2 roll
3455     }
3456     { pop pop }
3457   } ifelse
3458 }
3459 if
3460 counttomark 1 add copy
3461 pop pdf.currentrect
3462 /ANN pdfmark
3463 }
3464 def

```

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

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

```

3465 /pdf.pdfmark
3466 {
3467   SDict /pdf.pdfmark.good true put
3468   dup /ANN eq
3469   {
3470     pdf.pdfmark.store
3471     pdf.pdfmark.dict
3472     begin
3473       Subtype /Link eq
3474       currentdict /Rect known and
3475       SDict /pdf.outerbox known and
3476       SDict /pdf.baselineskip known and
3477       {
3478         Rect 3 get
3479         pdf.linkmargin 2 mul add
3480         pdf.outerbox pdf.rect.ht add
3481         Rect 1 get sub
3482         pdf.baselineskip div round cvi 0 gt
3483         { pdf.breaklink }
3484       } if
3485     }
3486   } if

```

```

3487         end
3488         SDict /pdf.outerbox undef
3489         SDict /pdf.baselineskip undef
3490         currentdict /pdf.pdfmark.dict undef
3491     }
3492     if
3493     pdf.pdfmark.good
3494     { pdfmark }
3495     { cleartomark }
3496     ifelse
3497 }
3498 def
3499 /pdf.pdfmark.store
3500 {
3501     /pdf.pdfmark.dict 65534 dict def
3502     counttomark 1 add copy
3503     pop
3504     {
3505         dup mark eq
3506         {
3507             pop
3508             exit
3509         }
3510         {
3511             pdf.pdfmark.dict
3512             begin def end
3513         }
3514         ifelse
3515     }
3516     loop
3517 }
3518 def

```

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

```

3519 </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.

<code>_</code>	154	<code>_box_backend_rotate:Nn</code>	235, 283, 340, <u>419</u>
A			
<code>\AtBeginDvi</code>	57	<code>_box_backend_rotate_aux:Nn</code>	235, 283, <u>340</u>
B			
bool commands:			
<code>\bool_gset_false:N</code>	1174, 1193, 1216, 1238, 1254, 1355, 1592, 1628, 2196, 2242	<code>_box_backend_scale:Nnn</code>	252, <u>311</u> , 355, <u>432</u>
<code>\bool_gset_true:N</code>	1172, 1241, 1353, 1607, 2189, 2195	<code>\l_box_backend_sin_fp</code>	<u>283</u>
<code>\bool_if:NTF</code>	67, 675, 1184, 1188, 1204, 1207, 1211, 1222, 1229, 1233, 1245, 1249, 1366, 1371, 1376, 1566, 1611, 1724, 1759, 1869, 1911, 2184, 2199, 2204, 2209	<code>\g_box_clip_path_int</code>	<u>369</u>
<code>\bool_if:nTF</code>	2418, 2684, 2938	C	
<code>\bool_lazy_and:nnTF</code>	867, 937, 3028, 3055	char commands:	
<code>\bool_lazy_or:nnTF</code>	1751, 1904	<code>\char_set_catcode_space:n</code>	154
<code>\bool_new:N</code>	1175, 1242, 1356, 1608, 2169, 2170	clist commands:	
<code>\bool_set_false:N</code>	1734, 1836, 1929, 1993	<code>\clist_map_function:nN</code>	1262, 1386
box commands:		<code>\clist_map_function:nn</code>	1635
<code>\box_dp:N</code>	224, 226, 274, 276, 331, 333, 380, 382, 384, 386, 2221, 2254, 2255, 2280	color internal commands:	
<code>\box_ht:N</code>	226, 276, 333, 384, 386, 1771, 1966, 2226, 2265, 2266, 2282	<code>_color_backend:nnn</code>	1060
<code>\box_if_empty:N</code>	2315	<code>_color_backend_cmyk:w</code>	1061
<code>\box_move_down:nn</code>	2143, 2221	<code>_color_backend_devicen_init:n</code>	<u>906</u>
<code>\box_move_up:nn</code>	2145, 2226	<code>_color_backend_devicen_</code>	
<code>\box_new:N</code>	2028, 2133, 2134	<code>init:nnn</code>	822, <u>906</u>
<code>\box_set_dp:Nn</code>	1691	<code>_color_backend_devicen_init:w</code>	<u>906</u>
<code>\box_set_ht:Nn</code>	1690	<code>_color_backend_fill:n</code>	967, 994, <u>1024</u> , <u>1042</u> , <u>1049</u>
<code>\box_set_wd:Nn</code>	288, 1689	<code>_color_backend_fill_cmyk:n</code>	967, <u>1001</u> , <u>1024</u> , <u>1049</u>
<code>\box_use:N</code>	231, 249, 263, 279, 306, 320, 336, 352, 364, 415, 429, 448, 1306, 1501, 1692, 2174	<code>_color_backend_fill_devicen:nn</code>	993, <u>1015</u> , <u>1041</u> , <u>1111</u>
<code>\box_wd:N</code>	225, 233, 275, 281, 332, 338, 381, 383, 1770, 1965	<code>_color_backend_fill_gray:n</code>	967, <u>1001</u> , <u>1024</u> , <u>1049</u>
box internal commands:		<code>_color_backend_fill_rgb:n</code>	967, <u>1001</u> , <u>1024</u> , <u>1049</u>
<code>_box_backend_clip:N</code>	213, 268, 325, <u>369</u>	<code>_color_backend_fill_separation:nn</code>	993, <u>1001</u> , <u>1041</u> , <u>1111</u>
<code>\l_box_backend_cos_fp</code>	<u>283</u>	<code>\l_color_backend_fill_tl</code>	637, 647, 975, 990
		<code>\c_color_backend_main_stack_int</code>	516
		<code>_color_backend_pickup:N</code>	456, <u>479</u>
		<code>_color_backend_pickup:w</code>	14, 456, <u>479</u>
		<code>_color_backend_reset:</code>	619, 639, 656, 978, 991, <u>1001</u> , 1033, 1058
		<code>_color_backend_rgb:w</code>	1084
		<code>_color_backend_select:n</code>	619, 671
		<code>_color_backend_select:nn</code>	639, 849
		<code>_color_backend_select_cmyk:n</code>	619, 639, <u>656</u>
		<code>_color_backend_select_devicen:nn</code>	670, <u>842</u> , <u>848</u> , <u>959</u>

<code>__color_backend_select_gray:n ..</code>	<code>__color_backend_stroke_separation:nn</code>
..... 619 , 639 , 656 993 , 1001 , 1041 , 1111
<code>__color_backend_select_rgb:n ...</code>	<code>\l__color_backend_stroke_tl</code>
..... 619 , 639 , 656 637 , 648 , 977 , 988
<code>__color_backend_select_separation:nn</code>	<code>\g__color_model_int .. 680, 828, 875, 945</code>
..... 670 , 842 , 848 , 959	<code>\c__color_model_range_CIELAB_tl ..</code>
<code>__color_backend_separation_-</code> 783 , 818 , 895 , 902
<code>init:n</code>	<code>color.sc</code>
673 , 851 , 930 , 956	619 , 3138
<code>__color_backend_separation_-</code>	cs commands:
<code>init:nnn</code>	<code>\cs_generate_variant:Nn ..</code>
673	49 , 63 ,
<code>__color_backend_separation_-</code>	66 , 99 , 138 , 143 , 170 , 201 , 207 , 569 ,
<code>init:nnnn</code>	606 , 685 , 1121 , 1316 , 1510 , 1883 ,
673	1940 , 1956 , 2032 , 2069 , 2128 , 2620 ,
<code>__color_backend_separation_-</code>	2648 , 2758 , 2780 , 2815 , 3025 , 3093
<code>init:nnnnn</code>	<code>\cs_gset:Npx ..</code>
673 , 844 , 851	2430 , 2434 , 2943 , 2948
<code>__color_backend_separation_-</code>	<code>\cs_gset_eq:NN</code>
<code>init:nw</code>	663 ,
673	664 , 962 , 1008 , 1009 , 1015 , 1017 , 1019
<code>__color_backend_separation_-</code>	<code>\cs_gset_protected:Npn</code>
<code>init:w</code> 551 , 658 , 665 , 961 , 1003 ,
673	1010 , 1012 , 1014 , 3059 , 3098 , 3107
<code>__color_backend_separation_-</code>	<code>\cs_if_exist:NTF</code>
<code>init_/DeviceCMYK:nnn</code>	27 ,
673	50 , 457 , 480 , 539 , 2311 , 2709 , 2735
<code>__color_backend_separation_-</code>	<code>\cs_if_exist_p:N ..</code>
<code>init_/DeviceGray:nnn</code>	868 , 938 , 3029 , 3056
673	<code>\cs_if_exist_use:NTF</code>
<code>__color_backend_separation_-</code>	38 , 698
<code>init_/DeviceRGB:nnn</code>	<code>\cs_new:Npn</code>
673	707 , 709 , 711 ,
<code>__color_backend_separation_-</code>	713 , 720 , 726 , 728 , 734 , 751 , 758 ,
<code>init_aux:nnnnnn</code>	760 , 950 , 1267 , 1391 , 1639 , 1969 ,
673	1978 , 2022 , 2047 , 2129 , 2131 , 2164 ,
<code>__color_backend_separation_-</code>	2336 , 2436 , 2437 , 2590 , 2621 , 2622 ,
<code>init_CIELAB:nnn</code>	2740 , 2773 , 2816 , 2818 , 2834 , 2951 ,
673 , 844 , 851	2952 , 2962 , 2967 , 2968 , 2973 , 2974
<code>__color_backend_separation_-</code>	<code>\cs_new:Npx</code>
<code>init_CIELAB:nnnnnn</code>	.. 2457 , 2492 , 2649 , 2660 , 2727 , 2864
845	<code>\cs_new_eq:NN</code>
<code>__color_backend_separation_-</code> 46 , 57 , 59 , 672 , 850 , 956 ,
<code>init_count:n</code>	997 , 998 , 1045 , 1046 , 1113 , 1114 ,
673	1120 , 1315 , 1321 , 1322 , 1509 , 1516 ,
<code>__color_backend_separation_-</code>	1701 , 1730 , 1781 , 1782 , 1824 , 1832 ,
<code>init_count:w</code>	1854 , 1925 , 1982 , 1989 , 2021 , 2174
673	<code>\cs_new_protected:Npn</code>
<code>__color_backend_separation_-</code>	47 , 54 ,
<code>init_Device:Nn</code>	61 , 64 , 72 , 78 , 83 , 85 , 89 , 100 , 110 ,
673	119 , 128 , 141 , 144 , 146 , 148 , 168 ,
<code>\g__color_backend_stack_int ...</code>	173 , 182 , 192 , 202 , 213 , 235 , 237 ,
516	252 , 268 , 283 , 285 , 311 , 325 , 340 ,
<code>\l__color_backend_stack_int</code>	342 , 355 , 369 , 419 , 432 , 456 , 474 ,
..... 513 , 541 , 547 , 649 , 653 , 976 , 989	479 , 487 , 517 , 560 , 570 , 582 , 596 ,
<code>__color_backend_stroke:n</code>	607 , 619 , 621 , 623 , 625 , 633 , 639 ,
..... 967 , 996 , 1001	641 , 643 , 645 , 652 , 670 , 686 , 776 ,
<code>__color_backend_stroke_cmyk:n ..</code>	822 , 842 , 843 , 844 , 845 , 848 , 851 ,
..... 967 , 1024 , 1060	880 , 884 , 906 , 967 , 969 , 971 , 973 ,
<code>__color_backend_stroke_cmyk:w</code>	980 , 982 , 984 , 986 , 993 , 995 , 1024 ,
1060	1026 , 1028 , 1030 , 1035 , 1037 , 1039 ,
<code>__color_backend_stroke_devicen:nn</code>	1041 , 1043 , 1049 , 1051 , 1053 , 1055 ,
..... 993 , 1019 , 1041 , 1111	1060 , 1062 , 1073 , 1081 , 1083 , 1085 ,
<code>__color_backend_stroke_gray:n ..</code>	
..... 967 , 1024 , 1060	
<code>__color_backend_stroke_gray_-</code>	
<code>aux:n</code>	
1060	
<code>__color_backend_stroke_rgb:n ...</code>	
..... 967 , 1024 , 1060	
<code>__color_backend_stroke_rgb:w .</code>	
1060	

1111, 1112, 1122, 1127, 1132, 1134,
 1136, 1144, 1152, 1161, 1171, 1173,
 1176, 1178, 1195, 1200, 1218, 1240,
 1243, 1256, 1269, 1274, 1276, 1278,
 1280, 1282, 1284, 1286, 1288, 1293,
 1317, 1319, 1323, 1328, 1333, 1343,
 1352, 1354, 1357, 1359, 1361, 1363,
 1368, 1373, 1378, 1380, 1393, 1398,
 1400, 1402, 1404, 1406, 1408, 1410,
 1412, 1423, 1448, 1460, 1472, 1484,
 1491, 1511, 1517, 1522, 1527, 1538,
 1548, 1558, 1560, 1562, 1564, 1595,
 1597, 1602, 1604, 1606, 1609, 1630,
 1641, 1654, 1656, 1658, 1660, 1662,
 1664, 1666, 1668, 1670, 1678, 1702,
 1716, 1731, 1743, 1748, 1776, 1788,
 1801, 1811, 1826, 1833, 1841, 1852,
 1856, 1859, 1874, 1884, 1919, 1926,
 1932, 1938, 1941, 1948, 1957, 1962,
 1970, 1983, 1990, 1996, 1998, 2000,
 2011, 2030, 2033, 2035, 2039, 2049,
 2070, 2075, 2080, 2085, 2095, 2100,
 2108, 2136, 2141, 2173, 2175, 2180,
 2182, 2187, 2202, 2207, 2244, 2273,
 2292, 2301, 2338, 2345, 2371, 2376,
 2404, 2416, 2428, 2432, 2438, 2440,
 2444, 2468, 2470, 2472, 2483, 2503,
 2513, 2536, 2550, 2560, 2571, 2592,
 2623, 2671, 2682, 2688, 2716, 2750,
 2752, 2759, 2761, 2765, 2775, 2781,
 2786, 2791, 2796, 2798, 2800, 2808,
 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,
 2999, 3006, 3039, 3044, 3061, 3063,
 3069, 3075, 3127, 3129, 3131, 3133
 \cs_new_protected:Npx
 520, 673, 1096, 2699, 2756, 2841
 \cs_set:Npn 152
 \cs_set_eq:NN 2332, 2333
 \cs_set_protected:Npn 459, 482

D

dim commands:

\dim_eval:n 2139, 2374,
 2452, 2453, 2454, 2511, 2546, 2547,
 2548, 2828, 2829, 2830, 2873, 2899
 \dim_max:nn 2252, 2263
 \dim_set:Nn ... 1770, 1771, 1965, 1966
 \dim_to_decimal:n .. 380, 381, 382,
 383, 384, 386, 1520, 1525, 1531,
 1532, 1533, 1534, 1543, 1544, 1545,

1636, 1655, 2016, 2017, 2250, 2261,
 2279, 2280, 2281, 2282, 2286, 2342
 \dim_to_decimal_in_bp:n
 224, 225, 226, 274, 275, 276,
 331, 332, 333, 1140, 1141, 1148,
 1149, 1156, 1157, 1165, 1166, 1167,
 1264, 1268, 1272, 1326, 1331, 1337,
 1338, 1339, 1347, 1348, 1388, 1392,
 1396, 1640, 1707, 1708, 1709, 1710,
 1846, 1847, 1848, 1849, 1898, 1899,
 1900, 1901, 2005, 2006, 2007, 2008
 draw internal commands:
 __draw_align_currentpoint... .. 35
 __draw_backend_add_to_path:n ...
 1517, 1563
 __draw_backend_begin:
 1122, 1317, 1511
 __draw_backend_box_use:Nnnnn ...
 31, 1293, 1491, 1678
 __draw_backend_cap_but:
 1256, 1380, 1630
 __draw_backend_cap_rectangle: ..
 1256, 1380, 1630
 __draw_backend_cap_round:
 1256, 1380, 1630
 __draw_backend_clip: 1176, 1357, 1562
 __draw_backend_closepath:
 1176, 1357, 1562
 __draw_backend_closestroke: ...
 1176, 1357, 1562
 __draw_backend_cm:nnnn 1288, 1301,
 1302, 1303, 1412, 1495, 1670, 1681
 __draw_backend_cm_aux:nnnn .. 1412
 __draw_backend_cm_decompose:nnnnN
 1418, 1447
 __draw_backend_cm_decompose_-
 auxi:nnnnN 1447
 __draw_backend_cm_decompose_-
 auxii:nnnnN 1447
 __draw_backend_cm_decompose_-
 auxiii:nnnnN 1447
 __draw_backend_curveto:nnnnnn ..
 1136, 1323, 1517
 __draw_backend_dash:n
 1256, 1380, 1630
 __draw_backend_dash_aux:nn .. 1630
 __draw_backend_dash_pattern:nn ..
 1256, 1380, 1630
 __draw_backend_discardpath: ...
 1176, 1357, 1562
 __draw_backend_end: 1122, 1317, 1511
 __draw_backend_evenodd_rule: ...
 1171, 1352, 1558
 __draw_backend_fill: 1176, 1357, 1562

<code>__draw_backend_fillstroke:</code>	
.	1176 , 1357 , 1562
<code>__draw_backend_join_bevel:</code>	
.	1256 , 1380 , 1630
<code>__draw_backend_join_miter:</code>	
.	1256 , 1380 , 1630
<code>__draw_backend_join_round:</code>	
.	1256 , 1380 , 1630
<code>__draw_backend_lineto:nn</code>	
.	1136 , 1323 , 1517
<code>__draw_backend_linewidth:n</code>	
.	1256 , 1380 , 1630
<code>__draw_backend_literal:n</code>	
.	1120 , 1125 , 1129 , 1133 , 1135 , 1138 , 1146 , 1154 , 1163 , 1177 , 1180 , 1181 , 1182 , 1183 , 1186 , 1192 , 1202 , 1209 , 1215 , 1220 , 1225 , 1226 , 1227 , 1228 , 1231 , 1237 , 1247 , 1253 , 1258 , 1271 , 1275 , 1277 , 1279 , 1281 , 1283 , 1285 , 1287 , 1290 , 1295 , 1296 , 1297 , 1298 , 1299 , 1300 , 1304 , 1305 , 1307 , 1308 , 1309 , 1310 , 1311 , 1315 , 1325 , 1330 , 1335 , 1345 , 1358 , 1360 , 1362 , 1365 , 1370 , 1375 , 1379 , 1382 , 1395 , 1399 , 1401 , 1403 , 1405 , 1407 , 1409 , 1411 , 1509 , 1569 , 1588 , 1614
<code>__draw_backend_miterlimit:n</code> . . .	
.	1256 , 1380 , 1630
<code>__draw_backend_moveto:nn</code>	
.	1136 , 1323 , 1517
<code>__draw_backend_nonzero_rule:</code> . . .	
.	1171 , 1352 , 1558
<code>__draw_backend_path:n</code>	1562
<code>__draw_backend_rectangle:nmmn</code> . .	
.	1136 , 1323 , 1517
<code>__draw_backend_scope:n</code> 1559 , 1561 , 1581 , 1621 , 1643 , 1655 , 1657 , 1659 , 1661 , 1663 , 1665 , 1667 , 1669 , 1672	
<code>__draw_backend_scope_begin:</code> . . .	
.	1132 , 1318 , 1321
<code>__draw_backend_scope_end:</code>	
.	1132 , 1320 , 1321
<code>__draw_backend_stroke:</code>	
.	1176 , 1357 , 1562
<code>\g__draw_clip_path_int</code>	
.	1568 , 1571 , 1584 , 1613 , 1616 , 1624
<code>\g__draw_draw_clip_bool</code>	1176 , 1562
<code>\g__draw_draw_eor_bool</code>	
.	1171 , 1188 , 1204 , 1211 , 1222 , 1233 , 1249 , 1352 , 1366 , 1371 , 1376
<code>\g__draw_draw_path_int</code>	1562
<code>\g__draw_draw_path_tl</code>	
.	1517 , 1573 , 1589 , 1591 , 1618 , 1627
<code>\g__draw_path_int</code>	1577 , 1594
E	
<code>\errmessage</code>	38
<code>\evensidemargin</code>	2219
exp commands:	
<code>\exp_after:wN</code>	159 , 465 , 1976
<code>\exp_args:Ne</code>	722 , 2373 , 2898
<code>\exp_args:Nf</code>	1261 , 1385 , 2138
<code>\exp_args:NNf</code>	236 , 284 , 341
<code>\exp_args:Nnx</code>	2125 , 2811
<code>\exp_args:NV</code>	461
<code>\exp_args:Nx</code>	677 , 1794 , 1815 , 2082 , 2097 , 2215 , 2777 , 2984 , 3041
<code>\exp_last_unbraced:Nx</code>	470 , 484
<code>\exp_not:N</code>	522 , 523 , 531 , 533 , 679 , 680 , 2459 , 2461 , 2464 , 2494 , 2496 , 2499 , 2651 , 2653 , 2656 , 2662 , 2664 , 2667 , 2704 , 2705 , 2711 , 2712 , 2731 , 2736 , 2845 , 2853 , 2869
<code>\exp_not:n</code>	48 , 97 , 108 , 136 , 2073 , 2078 , 2367 , 2606 , 2607 , 2621 , 2622 , 2634 , 2635 , 2789 , 2794 , 2805 , 2878
<code>\ExplBackendFileDate</code>	1
F	
file commands:	
<code>\file_compare_timestamp:nNnTF</code> .	1803
<code>\file_parse_full_name:nNNN</code>	1790 , 1813
<code>\fmtversion</code>	52
fp commands:	
<code>\fp_compare:nNnTF</code>	
.	243 , 290 , 296 , 348 , 1428 , 1441 , 1486
<code>\fp_eval:n</code>	236 , 245 , 258 , 259 , 284 , 301 , 316 , 318 , 341 , 350 , 361 , 362 , 426 , 441 , 442 , 1068 , 1069 , 1070 , 1078 , 1091 , 1092 , 1093 , 1430 , 1435 , 1436 , 1443 , 1453 , 1454 , 1455 , 1456 , 1465 , 1466 , 1467 , 1468 , 1477 , 1478 , 1479 , 1480 , 2364 , 2533 , 2892 , 2985 , 2995 , 3002 , 3042 , 3066 , 3073 , 3134
<code>\fp_new:N</code>	309 , 310
<code>\fp_set:Nn</code>	289 , 292
<code>\fp_use:N</code>	295 , 299 , 304
<code>\fp_zero:N</code>	291
<code>\c_zero_fp</code> 243 , 290 , 296 , 348 , 1428 , 1441	
G	
graphics commands:	
<code>\graphics_bb_restore:nTF</code> .	1745 , 1959
<code>\graphics_bb_save:n</code>	1774 , 1967
<code>\l_graphics_decodearray_tl</code>	
.	1722 , 1723 , 1733 , 1753 , 1757 , 1758 , 1835 , 1867 , 1868 , 1906 , 1909 , 1910 , 1928 , 1992
<code>\graphics_extract_bb:n</code>	
.	1830 , 1837 , 1987 , 1994

\l_graphics_interpolate_bool . . .	__graphics_backend_include:nn 1996
. 1724 , 1734 , 1752 , 1759 ,	__graphics_backend_include_-
1836 , 1869 , 1905 , 1911 , 1929 , 1993	auxi:nn 1841
\l_graphics_llx_dim	__graphics_backend_include_-
. 1707 , 1846 , 1898 , 2005	auxii:nnn 1841
\l_graphics_lly_dim	__graphics_backend_include_-
. 1708 , 1847 , 1899 , 2006	auxiii:nnn 1841
\l_graphics_name_tl 1808	__graphics_backend_include_-
\l_graphics_page_int	bitmap_quote:w 1970 , 2011
. 1718 , 1738 , 1739 , 1763 ,	__graphics_backend_include_-
1764 , 1828 , 1865 , 1866 , 1892 , 1893 ,	eps:n 1702 , 1783 , 1841 , 1996
1921 , 1934 , 1935 , 1974 , 1975 , 1985	__graphics_backend_include_-
\l_graphics_pagebox_tl	jpg:n 1776 , 1841 , 2011
. 51 , 1719 , 1737 ,	__graphics_backend_include_-
1765 , 1766 , 1829 , 1863 , 1864 , 1894 ,	pdf:n 1776 , 1815 , 1841 , 1970 , 1996
1896 , 1922 , 1943 , 1944 , 1976 , 1986	__graphics_backend_include_pdf_-
\graphics_read_bb:n . 1701 , 1824 , 1982	quote:w 1973 , 1978
\l_graphics_urx_dim	__graphics_backend_include_-
. 1709 , 1770 , 1848 , 1900 , 1965 , 2007	png:n 1776 , 1841 , 2011
\l_graphics_ury_dim . . 1710 , 1771 ,	\l_graphics_backend_name_str . 1783
1849 , 1901 , 1966 , 2008 , 2016 , 2017	\l_graphics_graphics_attr_tl . . .
graphics internal commands: 1715 , 1720 ,
\l_graphics_backend_dir_str . 1783	1727 , 1735 , 1745 , 1772 , 1774 , 1779
\l_graphics_backend_ext_str . 1783	\l_graphics_internal_box
__graphics_backend_getbb_auxi:n 1768 , 1770 , 1771 , 1964 , 1965 , 1966
. 1716	\g_graphics_track_int
__graphics_backend_getbb_- 1840 , 1886 , 1887
auxi:nN 1919	group commands:
__graphics_backend_getbb_-	\group_begin: 151 , 179 , 198
auxii:n 1716	\group_end: 164 , 187
__graphics_backend_getbb_-	\group_insert_after:N 631 , 650 , 661 ,
auxii:nnN 1919	978 , 991 , 1006 , 1033 , 1058 , 3053 , 3090
__graphics_backend_getbb_-	
auxiii:nNnn 1919	H
__graphics_backend_getbb_-	hbox commands:
auxiv:nnNnn 1919	\hbox:n 2144 , 2147 ,
__graphics_backend_getbb_-	2222 , 2228 , 2381 , 2388 , 2906 , 2917
auxv:nNnn 1919	\hbox_overlap_right:n 231 ,
__graphics_backend_getbb_-	263 , 279 , 320 , 336 , 364 , 448 , 1306 , 1501
auxvi:nNnn 1960 , 1962	\hbox_set:Nn 1768 , 1964 , 2214 , 2246
__graphics_backend_getbb_eps:n .	\hbox_set:Nw 2197
. 1701 , 1783 , 1824 , 1982	\hbox_set_end: 2212
__graphics_backend_getbb_eps:nm	\hbox_unpack:N 2333
. 1783	hook commands:
__graphics_backend_getbb_eps:nn	\hook_gput_code:nnn 55
. 1794 , 1801	
__graphics_backend_getbb_jpg:n .	I
. 1716 , 1824 , 1919 , 1983	int commands:
__graphics_backend_getbb_-	\int_compare:nNnTF 516 ,
pagebox:w 1919 , 1976	558 , 656 , 959 , 1001 , 1738 , 1763 ,
__graphics_backend_getbb_pdf:n .	1865 , 1892 , 1934 , 1974 , 2305 , 2406 ,
. 1716 , 1809 , 1824 , 1919 , 1990	2702 , 2730 , 2843 , 2850 , 2866 , 3096
__graphics_backend_getbb_png:n .	\int_const:Nn 157 , 163 , 523 ,
. 1716 , 1824 , 1919 , 1983	549 , 584 , 1772 , 1887 , 2042 , 2580 , 2768

<code>\int_eval:n</code> 61 , 75 , 76 , 80 , 217 , 218 , 220 , 221 , 229 , 241 , 256 , 1120 , 2408 , 2420
. 565 , 575 , 604 , 615 , 718 , 727 , 740 , 742 , 746 , 759 , 2430 , 2434 , 2680 , 2705 , 2712 , 2725 , 2935 , 2943 , 2948	
<code>\int_gincr:N</code>	205 , 371 , 522 , 1568 , 1613 , 1886 , 2041 , 2110 , 2154 , 2231 , 2767 , 2810 , 2823 , 2845
<code>\int_gset:Nn</code>	180 , 199 , 2294
<code>\int_gset_eq:NN</code>	188 , 2155 , 2232 , 2824
<code>\int_if_exist:NTF</code>	1876
<code>\int_if_odd:nTF</code>	2217
<code>\int_new:N</code>	171 , 172 , 418 , 513 , 519 , 1594 , 1840 , 2037 , 2135 , 2166 , 2168 , 2763 , 2820 , 2836
<code>\int_set:Nn</code>	541
<code>\int_set_eq:NN</code> ...	176 , 195 , 547 , 2306
<code>\int_step_function:nnnN</code>	744
<code>\int_use:N</code>	373 , 404 , 531 , 542 , 680 , 828 , 875 , 945 , 1571 , 1577 , 1584 , 1616 , 1624 , 1739 , 1764 , 1779 , 1866 , 1879 , 1891 , 1893 , 1975 , 2048 , 2113 , 2126 , 2130 , 2158 , 2165 , 2236 , 2337 , 2591 , 2601 , 2774 , 2812 , 2817 , 2827 , 2835 , 2853 , 2869
<code>\int_value:w</code> 2459 , 2494 , 2651 , 2662 , 2680
<code>\int_zero:N</code> ...	1718 , 1828 , 1921 , 1985
K	
kernel internal commands:	
<code>__kernel_backend_align_begin: ..</code> 72 , 216 , 240 , 255
<code>__kernel_backend_align_end: ...</code> 72 , 230 , 248 , 262
<code>__kernel_backend_first_shipout:n</code> 50 , 69 , 526 , 677
<code>\g__kernel_backend_header_bool ..</code> 67 , 675
<code>__kernel_backend_literal:n</code> 46 , 62 , 65 , 70 , 74 , 81 , 84 , 86 , 142 , 145 , 147 , 149 , 169 , 345 , 358 , 528 , 553 , 554 , 562 , 572 , 627 , 634 , 660 , 666 , 688 , 824 , 1005 , 1011 , 1013 , 1032 , 1057 , 1124 , 1130 , 1425 , 1432 , 1438 , 1498 , 1503 , 1704 , 1843 , 1878 , 1888 , 2002 , 2013 , 2757 , 2873 , 2935 , 2939 , 2944 , 2949
<code>__kernel_backend_literal_page:n</code> 100 , 144 , 2751 , 2753 , 2954 , 2956
<code>__kernel_backend_literal_pdf:n .</code>	.. 89 , 141 , 271 , 328 , 1315 , 3105 , 3120
<code>__kernel_backend_literal_-</code>	
<code>postscript:n</code>	
 61 , 75 , 76 , 80 , 217 , 218 , 220 , 221 , 229 , 241 , 256 , 1120 , 2408 , 2420
<code>__kernel_backend_literal_svg:n .</code> 168 , 175 , 186 , 194 , 204 , 372 , 374 , 391 , 1509 , 1682 , 1693
<code>__kernel_backend_matrix:n</code> 128 , 293 , 314 , 1415
<code>__kernel_backend_postscript:n ..</code> 64 , 629 , 1036 , 1038 , 1040 , 1044 , 2031 , 2087 , 2102 , 2144 , 2150 , 2190 , 2222 , 2229 , 2233 , 2247 , 2275 , 2319 , 2326 , 2332 , 2340 , 2347 , 2381 , 2388 , 3008
<code>__kernel_backend_scope:n</code> 173 , 401 , 406 , 1098 , 1514 , 3134
<code>__kernel_backend_scope_begin: ..</code> 83 , 110 , 146 , 173 , 215 , 239 , 254 , 270 , 287 , 313 , 327 , 344 , 357 , 1321 , 1493 , 1513 , 1680
<code>__kernel_backend_scope_begin:n .</code> 173 , 393 , 421 , 434
<code>__kernel_backend_scope_end: ...</code>	. 83 , 110 , 146 , 173 , 232 , 250 , 264 , 280 , 307 , 321 , 337 , 353 , 365 , 416 , 430 , 449 , 551 , 1322 , 1505 , 1516 , 1694
<code>\g__kernel_backend_scope_int ...</code>	171 , 178 , 180 , 185 , 189 , 197 , 199 , 205
<code>\l__kernel_backend_scope_int ...</code> 171 , 177 , 190 , 196
<code>__kernel_color_backend_stack_-</code>	
<code>init:Nnn</code>	516 , 582 , 3032
<code>__kernel_color_backend_stack_-</code>	
<code>pop:n</code>	558 , 596 , 653 , 3062
<code>__kernel_color_backend_stack_-</code>	
<code>push:nn</code> 558 , 596 , 649 , 976 , 989 , 3051 , 3088
<code>__kernel_dependency_version_-</code>	
<code>check:Nn</code>	1
<code>__kernel_dependency_version_-</code>	
<code>check:nn</code>	27 , 29
<code>__kernel_kern:n</code> 2149 , 2151 , 2380 , 2384 , 2387 , 2391 , 2905 , 2913 , 2916 , 2932
<code>\c__kernel_sys_dvipdfmx_version_-</code>	
<code>int</code>	151 , 516 , 558 , 656 , 959 , 1001 , 2843 , 2850 , 2866 , 3096
M	
<code>\MessageBreak</code>	40
mode commands:	
<code>\mode_if_horizontal:TF</code> ...	2296 , 2303
<code>\mode_if_math:TF</code>	2194
O	
<code>\oddsidemargin</code>	2218

opacity internal commands:

[__opacity_backend:nn](#) [3127](#)
[__opacity_backend:nnn](#) [2982](#)
[__opacity_backend_fill:n](#)
 [2982](#), [3063](#), [3127](#)
[__opacity_backend_fill_stroke:nn](#)
 [3065](#), [3071](#), [3075](#), [3093](#), [3107](#)
[\l__opacity_backend_fill_tl](#)
 .. [3037](#), [3046](#), [3072](#), [3080](#), [3100](#), [3112](#)
[__opacity_backend_fillstroke:nn](#)
 [3063](#)
[__opacity_backend_reset:](#) [3039](#), [3090](#)
[__opacity_backend_select:n](#)
 [2982](#), [3039](#), [3096](#), [3127](#)
[__opacity_backend_select_aux:n](#) .
 [2982](#), [3039](#), [3078](#), [3098](#), [3110](#)
[\c__opacity_backend_stack_int](#) ...
 [3028](#), [3051](#), [3062](#), [3088](#)
[__opacity_backend_stroke:n](#)
 [2982](#), [3063](#), [3127](#)
[\l__opacity_backend_stroke_tl](#) ...
 .. [3037](#), [3047](#), [3067](#), [3081](#), [3101](#), [3113](#)

P

pdf commands:

[\pdf_object_if_exist:nTF](#) [886](#)
[\pdf_object_new:nn](#) [888](#)
[\pdf_object_ref:n](#) [901](#)
[\pdf_object_ref_last:](#)
 [865](#), [876](#), [935](#), [946](#)
[\pdf_object_unnamed_write:nn](#) ...
 [853](#), [882](#), [908](#)
[\pdf_object_write:nn](#) [889](#)

pdf internal commands:

[__pdf_backend:n](#) [2756](#),
[2760](#), [2762](#), [2788](#), [2793](#), [2802](#), [2825](#),
[2847](#), [2863](#), [2876](#), [2908](#), [2909](#), [2919](#)
[__pdf_backend_annotation:nnnn](#) ..
 [2136](#), [2444](#), [2821](#)
[__pdf_backend_annotation_-](#)
 aux:nnnn [2138](#), [2141](#)
[\g__pdf_backend_annotation_int](#) ..
 .. [2135](#), [2155](#), [2165](#), [2820](#), [2824](#), [2835](#)
[__pdf_backend_annotation_last:](#) .
 [2164](#), [2457](#), [2834](#)
[__pdf_backend_bdc:nn](#)
 [2438](#), [2750](#), [2953](#), [2975](#)
[__pdf_backend_catalog_gput:nn](#) ..
 [2033](#), [2550](#), [2759](#), [2959](#)
[__pdf_backend_compress_objects:n](#)
 [2404](#), [2671](#), [2934](#), [2969](#)
[__pdf_backend_compresslevel:n](#) ..
 [2404](#), [2671](#), [2934](#), [2969](#)

[\l__pdf_backend_content_box](#) [2133](#),
[2197](#), [2221](#), [2224](#), [2226](#), [2255](#), [2266](#)
[__pdf_backend_destination:nn](#) ...
 [2345](#), [2513](#), [2874](#)
[__pdf_backend_destination:nnnn](#) .
 [2345](#), [2513](#), [2874](#)
[__pdf_backend_destination_-](#)
 aux:nnnn [2345](#), [2874](#)
[__pdf_backend_emc:](#)
 [2438](#), [2750](#), [2953](#), [2975](#)
[__pdf_backend_info_gput:nn](#)
 [2033](#), [2550](#), [2759](#), [2959](#)
[__pdf_backend_link:nw](#) [2175](#)
[__pdf_backend_link_aux:nw](#) ... [2175](#)
[__pdf_backend_link_begin:n](#) .. [2837](#)
[__pdf_backend_link_begin:nnnw](#) [2468](#)
[__pdf_backend_link_begin:nw](#) ...
 [2177](#), [2181](#), [2182](#)
[__pdf_backend_link_begin_aux:nw](#)
 [2185](#), [2187](#)
[__pdf_backend_link_begin_-](#)
 goto:nnw [2175](#), [2468](#), [2837](#)
[__pdf_backend_link_begin_-](#)
 user:nnw [2175](#), [2468](#), [2837](#)
[\g__pdf_backend_link_bool](#)
 [2170](#), [2184](#), [2189](#), [2204](#), [2242](#)
[\g__pdf_backend_link_dict_tl](#) ...
 [2167](#), [2192](#), [2237](#)
[__pdf_backend_link_end:](#)
 [2175](#), [2468](#), [2837](#)
[__pdf_backend_link_end_aux:](#) . [2175](#)
[\g__pdf_backend_link_int](#)
 [2166](#), [2232](#),
[2236](#), [2337](#), [2836](#), [2845](#), [2853](#), [2869](#)
[__pdf_backend_link_last:](#)
 [2336](#), [2492](#), [2864](#)
[__pdf_backend_link_margin:n](#) ...
 [2338](#), [2503](#), [2872](#)
[\g__pdf_backend_link_math_bool](#) ..
 [2169](#), [2195](#), [2196](#), [2199](#), [2209](#)
[__pdf_backend_link_minima:](#) .. [2175](#)
[__pdf_backend_link_outerbox:n](#) [2175](#)
[\g__pdf_backend_link_sf_int](#)
 [2168](#), [2294](#), [2305](#), [2306](#)
[__pdf_backend_link_sf_restore:](#) [2175](#)
[__pdf_backend_link_sf_save:](#) . [2175](#)
[\l__pdf_backend_model_box](#) . [2134](#),
[2214](#), [2246](#), [2254](#), [2265](#), [2280](#), [2282](#)
[__pdf_backend_objcompresslevel:n](#)
 [2671](#)
[\g__pdf_backend_object_int](#)
 [2037](#), [2041](#), [2044](#),
[2110](#), [2113](#), [2126](#), [2130](#), [2154](#), [2155](#),

2158, 2231, 2232, 2763, 2767, 2770, 2810, 2812, 2817, 2823, 2824, 2827	pdf.baselineskip 2175, 3465
__pdf_backend_object_last:	pdf.bordertracking 3223
2129, 2649, 2816, 2961	pdf.bordertracking.begin 3223
__pdf_backend_object_new:nn	pdf.bordertracking.continue 3223
2039, 2571, 2765, 2961	pdf.bordertracking.end 3223
__pdf_backend_object_now:nn	pdf.bordertracking.endpage 3223
2108, 2623, 2808, 2961	pdf.breaklink 3361
\g_pdf_backend_object_prop	pdf.breaklink.write 3361
2037, 2045, 2056, 2066, 2570, 2588, 2604, 2763, 2771, 2778	pdf.brokenlink.dict 3223
__pdf_backend_object_ref:n 2039, 2053, 2067, 2571, 2765, 2784, 2961	pdf.brokenlink.rect 3223
__pdf_backend_object_write:nn	pdf.brokenlink.skip 3223
2049, 2592, 2775, 2961	pdf.count 3361
__pdf_backend_object_write:nnn 2775	pdf.currentrect 3361
__pdf_backend_object_write_- array:nn 2049, 2775	pdf.cvs 3145
__pdf_backend_object_write_- dict:nn 2049, 2775	pdf.dest.anchor 3188
__pdf_backend_object_write_- fstream:nn 2049, 2775	pdf.dest.point 3188
__pdf_backend_object_write_- fstream:nnn 2083, 2085	pdf.dest.x 3188
__pdf_backend_object_write_- stream:nn 2049, 2775	pdf.dest.y 3188
__pdf_backend_object_write_- stream:nnn 2049	pdf.dest2device 3188
__pdf_backend_object_write_- stream:nnnn 2775	pdf.dev.x 3188
__pdf_backend_pageobject_ref:n	pdf.dev.y 3188
2131, 2660, 2818, 2961	pdf.dvi.pt 3145
__pdf_backend_pdfmark:n	pdf.globaldict 3142
2030, 2034, 2036, 2051, 2072, 2077, 2111, 2156, 2348, 2392, 2439, 2441	pdf.leftboundary 3223
__pdf_backend_version_major:	pdf.link.dict 2175
2430, 2436, 2727, 2943, 2944, 2951, 2973	pdf.linkdp.pad 2175, 3149
__pdf_backend_version_major_- gset:n 2428, 2699, 2941, 2971	pdf.linkht.pad 2175, 3149
__pdf_backend_version_minor:	pdf.linkmargin 3149
2434, 2436, 2727, 2948, 2949, 2951, 2973	pdf.llx 2175, 3152
__pdf_backend_version_minor_- gset:n 2428, 2699, 2941, 2971	pdf.lly 2175, 3152
\l_pdf_breaklink_pdfmark_tl	pdf.originx 3223
2171, 2239, 2331	pdf.originy 3223
__pdf_breaklink_postscript:n	pdf.outerbox 2175, 3465
2173, 2223, 2225, 2332	pdf.pdfmark 3465
__pdf_breaklink_usebox:N	pdf.pdfmark.dict 3465
2174, 2224, 2333	pdf.pdfmark.good 3465
__pdf_exp_not_i:nn . 2592, 2638, 2643	pdf.pt.dvi 3145
__pdf_exp_not_ii:nn 2592, 2639, 2644	pdf.rect 3152
\l_pdf_internal_box 2028	pdf.rect.ht 3145
	pdf.rightboundary 3223
	pdf.save.linkll 3152
	pdf.save.linkur 3152
	pdf.save.ll 3152
	pdf.save.ur 3152
	pdf.tmpa 3188
	pdf.tmpb 3188
	pdf.tmpc 3188
	pdf.tmpd 3188
	pdf.urx 3152
	pdf.ury 2175, 3152
	pdfmanagement commands:
	\pdfmanagement_add:nnn
	873, 943, 3034, 3048, 3082, 3085, 3102, 3114, 3117

tl commands:	
\c_space_tl	295, 300, 303, 532, 783, 990, 1553, 1706, 1707, 1708, 1709, 1845, 1846, 1847, 1848, 1893, 1896, 1898, 1899, 1900, 1901, 1973, 1975, 2004, 2005, 2006, 2007, 2237, 2466, 2501, 2658, 2669, 2827, 2854
\tl_clear:N	1719, 1727, 1733, 1829, 1835, 1922, 1928, 1986, 1992
\tl_gclear:N	1591, 1627
\tl_gset:Nn	1550, 2192
\tl_if_blank:nTF	533, 592, 731, 748, 755, 773, 857, 953
\tl_if_empty:NTF	1553, 1722, 1757, 1765, 1863, 1867, 1894, 1909, 1943
\tl_if_empty:nTF	1647
\tl_if_empty_p:N	1753, 1906
\tl_if_head_is_space:nTF	461
\tl_new:N	637, 638, 1557, 1715, 2167, 2171, 3037, 3038
\tl_put_right:Nn	2313
\tl_set:Nn	463, 475, 491, 494, 497, 501, 504, 647, 648, 975, 988, 1720, 1735, 1808, 2172, 2331, 3046, 3047, 3080, 3081, 3100, 3101, 3112, 3113
\tl_to_str:n	2043, 2048, 2581, 2591, 2602, 2769, 2774
\tl_use:N	815, 894
token commands:	
\c_math_toggle_token	2200, 2210
U	
use commands:	
\use:N	43, 2065, 2125, 2783, 2811
\use:n	59, 465, 501, 524, 871, 941, 1065, 1075, 1088, 1261, 1385, 1450, 1462, 1474, 1632, 1950
\use_none:n	1647, 1649, 2309
V	
\value	2217
vbox commands:	
\vbox_set:Nn	2317
\vbox_to_zero:n	2378, 2385, 2903, 2914
\vbox_unpack_drop:N	2325