

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}{2023-03-30}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2023-03-30}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2023-03-30}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2023-03-30}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2023-03-30}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2023-03-30}{ }
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 `<*luatex | pdftex>`

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

`_kernel_backend_literal_pdf:n`
`_kernel_backend_literal_pdf:x`

This is equivalent to `\special{pdf:}` but the engine can track it. Without the `direct` keyword everything is kept in sync: the transformation matrix is set to the current point automatically. Note that this is still inside the text (BT ...ET block).

```
89 \cs_new_protected:Npn \_kernel_backend_literal_pdf:n #1
90 {
91   <*luatex>
92     \tex_pdfextension:D literal
93   </luatex>
94   <*pdftex>
95     \tex_pdfliteral:D
96   </pdftex>
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   <*luatex>
103     \tex_pdfextension:D literal ~
104   </luatex>
105   <*pdftex>
106     \tex_pdfliteral:D
107   </pdftex>
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   <*luatex>
113     \tex_pdfextension:D save \scan_stop:
114   </luatex>
115   <*pdftex>
116     \tex_pdfsave:D
117   </pdftex>
118 }
119 \cs_new_protected:Npn \_kernel_backend_scope_end:
120 {
121   <*luatex>
122     \tex_pdfextension:D restore \scan_stop:
123   </luatex>
124   <*pdftex>
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 xdvipdfmx (`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 </dvipdfmx | xetex>

```

1.4 dvisvgm backend

151 $\langle *dvisvgm \rangle$

_kernel_backend_literal_svg:n
_kernel_backend_literal_svg:x

Unlike the other backends, the requirements for making SVG files mean that we can't conveniently transform all operations to the current point. That makes life a bit more tricky later as that needs to be accounted for. A new line is added after each call to help to keep the output readable for debugging.

152 \cs_new_protected:Npn _kernel_backend_literal_svg:n #1
153 { _kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }
154 \cs_generate_variant:Nn _kernel_backend_literal_svg:n { x }

(End definition for _kernel_backend_literal_svg:n.)

\g_kernel_backend_scope_int
\l_kernel_backend_scope_int

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

155 \int_new:N \g_kernel_backend_scope_int
156 \int_new:N \l_kernel_backend_scope_int

(End definition for \g_kernel_backend_scope_int and \l_kernel_backend_scope_int.)

_kernel_backend_scope_begin:
_kernel_backend_scope_end:
_kernel_backend_scope_begin:n
_kernel_backend_scope_begin:x
_kernel_backend_scope:n
_kernel_backend_scope:x

In SVG, the need to attach concepts to a scope means we need to be sure we will close all of the open scopes. That is easiest done if we only need an outer “wrapper” **begin/end** pair, and within that we apply operations as a simple scoped statements. To keep down the non-productive groups, we also have a **begin** version that does take an argument.

157 \cs_new_protected:Npn _kernel_backend_scope_begin:
158 {
159 _kernel_backend_literal_svg:n { <g> }
160 \int_set_eq:NN
161 \l_kernel_backend_scope_int
162 \g_kernel_backend_scope_int
163 \group_begin:
164 \int_gset:Nn \g_kernel_backend_scope_int { 1 }
165 }
166 \cs_new_protected:Npn _kernel_backend_scope_end:
167 {
168 \prg_replicate:nn
169 { \g_kernel_backend_scope_int }
170 { _kernel_backend_literal_svg:n { </g> } }
171 \group_end:
172 \int_gset_eq:NN
173 \g_kernel_backend_scope_int
174 \l_kernel_backend_scope_int
175 }
176 \cs_new_protected:Npn _kernel_backend_scope_begin:n #1
177 {
178 _kernel_backend_literal_svg:n { <g ~ #1 > }
179 \int_set_eq:NN
180 \l_kernel_backend_scope_int
181 \g_kernel_backend_scope_int
182 \group_begin:
183 \int_gset:Nn \g_kernel_backend_scope_int { 1 }
184 }
185 \cs_generate_variant:Nn _kernel_backend_scope_begin:n { x }

```

186 \cs_new_protected:Npn \__kernel_backend_scope:n #1
187 {
188   \__kernel_backend_literal_svg:n { <g ~ #1 > }
189   \int_gincr:N \g__kernel_backend_scope_int
190 }
191 \cs_generate_variant:Nn \__kernel_backend_scope:n { x }

(End definition for \__kernel_backend_scope_begin: and others.)

192 </dvisvgm>
193 </package>

```

2 l3backend-box implementation

```

194 <*package>
195 <@@=box>

```

2.1 dvips backend

```

196 <*dvips>

```

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

```

197 \cs_new_protected:Npn \__box_backend_clip:N #1
198 {
199   \__kernel_backend_scope_begin:
200   \__kernel_backend_align_begin:
201   \__kernel_backend_literal_postscript:n { matrix~currentmatrix }
202   \__kernel_backend_literal_postscript:n
203     { Resolution~72~div~VResolution~72~div~scale }
204   \__kernel_backend_literal_postscript:n { DVImag~dup~scale }
205   \__kernel_backend_literal_postscript:x
206     {
207       0 ~
208       \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
209       \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
210       \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
211       rectclip
212     }
213   \__kernel_backend_literal_postscript:n { setmatrix }
214   \__kernel_backend_align_end:
215   \hbox_overlap_right:n { \box_use:N #1 }
216   \__kernel_backend_scope_end:
217   \skip_horizontal:n { \box_wd:N #1 }
218 }

```

(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.

`__box_backend_rotate_aux:Nn`

```

219 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
220 { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
221 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
222 {
223   \__kernel_backend_scope_begin:
224   \__kernel_backend_align_begin:
225   \__kernel_backend_literal_postscript:x
226   {
227     \fp_compare:nNnTF {#2} = \c_zero_fp
228     { 0 }
229     { \fp_eval:n { round ( -(#2) , 5 ) } } ~
230     rotate
231   }
232   \__kernel_backend_align_end:
233   \box_use:N #1
234   \__kernel_backend_scope_end:
235 }

```

(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.

```

236 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
237 {
238   \__kernel_backend_scope_begin:
239   \__kernel_backend_align_begin:
240   \__kernel_backend_literal_postscript:x
241   {
242     \fp_eval:n { round ( #2 , 5 ) } ~
243     \fp_eval:n { round ( #3 , 5 ) } ~
244     scale
245   }
246   \__kernel_backend_align_end:
247   \hbox_overlap_right:n { \box_use:N #1 }
248   \__kernel_backend_scope_end:
249 }

```

(End definition for __box_backend_scale:Nnn.)

250 </dvips>

2.2 LuaTeX and pdfTeX backends

251 <*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.

```

252 \cs_new_protected:Npn \__box_backend_clip:N #1
253 {
254   \__kernel_backend_scope_begin:
255   \__kernel_backend_literal_pdf:x
256   {

```



```

257      0~
258      \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
259      \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
260      \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
261      re~W~n
262    }
263    \hbox_overlap_right:n { \box_use:N #1 }
264    \__kernel_backend_scope_end:
265    \skip_horizontal:n { \box_wd:N #1 }
266  }

```

(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.

```

267 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
268 { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
269 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
270 {
271   \__kernel_backend_scope_begin:
272   \box_set_wd:Nn #1 { 0pt }
273   \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
274   \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
275     { \fp_zero:N \l__box_backend_cos_fp }
276   \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
277   \__kernel_backend_matrix:x
278   {
279     \fp_use:N \l__box_backend_cos_fp \c_space_tl
280     \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp
281       { 0~0 }
282       {
283         \fp_use:N \l__box_backend_sin_fp
284         \c_space_tl
285         \fp_eval:n { -\l__box_backend_sin_fp }
286       }
287     \c_space_tl
288     \fp_use:N \l__box_backend_cos_fp
289   }
290   \box_use:N #1
291   \__kernel_backend_scope_end:
292 }
293 \fp_new:N \l__box_backend_cos_fp
294 \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.

```

295 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
296 {
297   \__kernel_backend_scope_begin:
298   \__kernel_backend_matrix:x

```

```

299     {
300       \fp_eval:n { round ( #2 , 5 ) } ~
301       0~0~
302       \fp_eval:n { round ( #3 , 5 ) }
303     }
304     \hbox_overlap_right:n { \box_use:N #1 }
305     \__kernel_backend_scope_end:
306   }

```

(End definition for __box_backend_scale:Nnn.)

```

307 </luatex | pdftex>

```

2.3 dvipdfmx/X_YTeX backend

```

308 <*dvipdfmx | xetex>

```

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

```

309 \cs_new_protected:Npn \__box_backend_clip:N #1
310 {
311   \__kernel_backend_scope_begin:
312   \__kernel_backend_literal_pdf:x
313   {
314     0~
315     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
316     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
317     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
318     re~W~n
319   }
320   \hbox_overlap_right:n { \box_use:N #1 }
321   \__kernel_backend_scope_end:
322   \skip_horizontal:n { \box_wd:N #1 }
323 }

```

(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.

```

324 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
325 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
326 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
327 {
328   \__kernel_backend_scope_begin:
329   \__kernel_backend_literal:x
330   {
331     x:rotate~
332     \fp_compare:nNnTF {#2} = \c_zero_fp
333     { 0 }
334     { \fp_eval:n { round ( #2 , 5 ) } }
335   }

```

```

336     \box_use:N #1
337     \__kernel_backend_scope_end:
338 }

```

(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.

```

339 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
340 {
341     \__kernel_backend_scope_begin:
342     \__kernel_backend_literal:x
343     {
344         x:scale~
345         \fp_eval:n { round ( #2 , 5 ) } ~
346         \fp_eval:n { round ( #3 , 5 ) }
347     }
348     \hbox_overlap_right:n { \box_use:N #1 }
349     \__kernel_backend_scope_end:
350 }

```

(End definition for `__box_backend_scale:Nnn`.)

```

351 </dviPDFmx | xetex>

```

2.4 dvisvgm backend

```

352 <*dvisvgm>

```

`__box_backend_clip:N`
`\g__kernel_clip_path_int`

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

```

353 \cs_new_protected:Npn \__box_backend_clip:N #1
354 {
355     \int_gincr:N \g__kernel_clip_path_int
356     \__kernel_backend_literal_svg:x
357     { < clipPath-id = " l3cp \int_use:N \g__kernel_clip_path_int " > }
358     \__kernel_backend_literal_svg:x
359     {
360         <
361         path ~ d =
362         "
363         M ~ 0 ~
364         \dim_to_decimal:n { -\box_dp:N #1 } ~
365         L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
366         \dim_to_decimal:n { -\box_dp:N #1 } ~
367         L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
368         \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
369         L ~ 0 ~
370         \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
371         Z

```

```

372         "
373     />
374 }
375 \__kernel_backend_literal_svg:n
376 { < /clipPath > }

```

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

```

377 \__kernel_backend_scope_begin:n
378 {
379     transform =
380     "
381         translate ( { ?x } , { ?y } ) ~
382         scale ( 1 , -1 )
383     "
384 }
385 \__kernel_backend_scope:x
386 {
387     clip-path =
388     "url ( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int ) "
389 }
390 \__kernel_backend_scope:n
391 {
392     transform =
393     "
394         scale ( -1 , 1 ) ~
395         translate ( { ?x } , { ?y } ) ~
396         scale ( -1 , -1 )
397     "
398 }
399 \box_use:N #1
400 \__kernel_backend_scope_end:
401 }
402 \int_new:N \g__kernel_clip_path_int

```

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

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

```

403 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
404 {
405     \__kernel_backend_scope_begin:x
406     {
407         transform =
408         "
409             rotate
410             ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
411         "
412     }
413     \box_use:N #1

```

```

414     \__kernel_backend_scope_end:
415 }

```

(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).

```

416 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
417 {
418     \__kernel_backend_scope_begin:x
419     {
420         transform =
421         "
422             translate ( { ?x } , { ?y } ) ~
423             scale
424             (
425                 \fp_eval:n { round ( -#2 , 5 ) } ,
426                 \fp_eval:n { round ( -#3 , 5 ) }
427             ) ~
428             translate ( { ?x } , { ?y } ) ~
429             scale ( -1 )
430         "
431     }
432     \hbox_overlap_right:n { \box_use:N #1 }
433     \__kernel_backend_scope_end:
434 }

```

(End definition for __box_backend_scale:Nnn.)

```

435 </divisvgn>

```

```

436 </package>

```

3 l3backend-color implementation

```

437 <*package>

```

```

438 <@@=color>

```

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

3.1 The color stack

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

3.1.1 Common code

439 $\langle *luatex \mid pdftex \rangle$

$\backslash l_color_backend_stack_int$ For tracking which stack is in use where multiple stacks are used: currently just pdfTeX/LuaTeX but at some future stage may also cover dvipdfmx/X_YTeX.

440 $\backslash int_new:N \backslash l_color_backend_stack_int$

(End definition for $\backslash l_color_backend_stack_int$.)

441 $\langle /luatex \mid pdftex \rangle$

3.1.2 LuaTeX and pdfTeX

442 $\langle *luatex \mid pdftex \rangle$

$\backslash_kernel_color_backend_stack_init:Nnn$

```
443 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
444 {
445   \int_const:Nn #1
446   {
447      $\langle *luatex \rangle$ 
448     \tex_pdffeedback:D colorstackinit ~
449      $\langle /luatex \rangle$ 
450      $\langle *pdftex \rangle$ 
451     \tex_pdfcolorstackinit:D
452      $\langle /pdftex \rangle$ 
453     \tl_if_blank:nF {#2} { #2 ~ }
454     {#3}
455   }
456 }
```

(End definition for $\backslash_kernel_color_backend_stack_init:Nnn$.)

$\backslash_kernel_color_backend_stack_push:nn$

$\backslash_kernel_color_backend_stack_pop:n$

```
457 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
458 {
459    $\langle *luatex \rangle$ 
460   \tex_pdfextension:D colorstack ~
461    $\langle /luatex \rangle$ 
462    $\langle *pdftex \rangle$ 
463   \tex_pdfcolorstack:D
464    $\langle /pdftex \rangle$ 
465   \int_eval:n {#1} ~ push ~ {#2}
466 }
467 \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
468 {
469    $\langle *luatex \rangle$ 
470   \tex_pdfextension:D colorstack ~
471    $\langle /luatex \rangle$ 
472    $\langle *pdftex \rangle$ 
473   \tex_pdfcolorstack:D
474    $\langle /pdftex \rangle$ 
475   \int_eval:n {#1} ~ pop \scan_stop:
476 }
```

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

477 \langle /luatex | pdftex \rangle

3.2 General color

3.2.1 dvips-style

478 \langle *dvips | dvisvgm \rangle

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

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_named:n
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc
479 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
480 { \__color_backend_select:n { cmyk ~ #1 } }
481 \cs_new_protected:Npn \__color_backend_select_gray:n #1
482 { \__color_backend_select:n { gray ~ #1 } }
483 \cs_new_protected:Npn \__color_backend_select_named:n #1
484 { \__color_backend_select:n { ~ #1 } }
485 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
486 { \__color_backend_select:n { rgb ~ #1 } }
487 \cs_new_protected:Npn \__color_backend_select:n #1
488 {
489   \__kernel_backend_literal:n { color~push~ #1 }
490 }
491 \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
492  $\langle$ /dvips $\rangle$ 
493 }
494 \cs_new_protected:Npn \__color_backend_reset:
495 { \__kernel_backend_literal:n { color~pop } }
```

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

496 \langle /dvips | dvisvgm \rangle

3.2.2 LuaTeX and pdfTeX

497 \langle *luatex | pdftex \rangle

```

\l__color_backend_fill_tl
\l__color_backend_stroke_tl
498 \tl_new:N \l__color_backend_fill_tl
499 \tl_new:N \l__color_backend_stroke_tl
500 \tl_set:Nn \l__color_backend_fill_tl { 0 ~ g }
501 \tl_set:Nn \l__color_backend_stroke_tl { 0 ~ G }
```

(End definition for `\l__color_backend_fill_tl` and `\l__color_backend_stroke_tl`.)

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_select:nn
\__color_backend_reset:
Store the values then pass to the stack.
502 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
503 { \__color_backend_select:nn { #1 ~ k } { #1 ~ K } }
504 \cs_new_protected:Npn \__color_backend_select_gray:n #1
505 { \__color_backend_select:nn { #1 ~ g } { #1 ~ G } }
506 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
507 { \__color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
508 \cs_new_protected:Npn \__color_backend_select:nn #1#2
509 {
```

```

510 \tl_set:Nn \l__color_backend_fill_tl {#1}
511 \tl_set:Nn \l__color_backend_stroke_tl {#2}
512 \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int { #1 ~ #2 }
513 }
514 \cs_new_protected:Npn \__color_backend_reset:
515 { \__kernel_color_backend_stack_pop:n \l__color_backend_stack_int }

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

516 </luatex | pdftex>

```

3.2.3 dvipdfmx/X_YTeX

These backends have the most possible approaches: it recognises both dvips-based color specials and its own format, plus one can include PDF statements directly. Recent releases also have a color stack approach similar to pdfTeX. Of the stack methods, the dedicated the most versatile is the latter as it can cover all of the use cases we have. However, at present this interacts problematically with any color on the original stack. We therefore stick to a single-stack approach here.

```

517 <*dvipdfmx | xetex>

```

```

\__color_backend_select:n Using the single stack is relatively easy as there is only one route.
  \__color_backend_select_cmyk:n 518 \cs_new_protected:Npn \__color_backend_select:n #1
  \__color_backend_select_gray:n 519 { \__kernel_backend_literal:n { pdf : bc ~ [ #1 ] } }
  \__color_backend_select_rgb:n 520 \cs_new_eq:NN \__color_backend_select_cmyk:n \__color_backend_select:n
\__color_backend_reset: 521 \cs_new_eq:NN \__color_backend_select_gray:n \__color_backend_select:n
522 \cs_new_eq:NN \__color_backend_select_rgb:n \__color_backend_select:n
523 \cs_new_protected:Npn \__color_backend_reset:
524 { \__kernel_backend_literal:n { pdf : ec } }

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

```

```

\__color_backend_select_named:n For classical named colors, the only value we should get is Black.
525 \cs_new_protected:Npn \__color_backend_select_named:n #1
526 {
527   \str_if_eq:nnTF {#1} { Black }
528     { \__color_backend_select_gray:n { 0 } }
529     { \msg_error:nnn { color } { unknown-named-color } {#1} }
530 }
531 \msg_new:nnn { color } { unknown-named-color }
532 { Named-color~'#1'~is-not-known. }

(End definition for \__color_backend_select_named:n.)

533 </dvipdfmx | xetex>

```

3.3 Separations

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

```

534 <*dvipdfmx | luatex | pdftex | xetex | dvips>

```

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


```

\g_color_backend_colorant_prop
535 \prop_new:N \g__color_backend_colorant_prop
(End definition for \g_color_backend_colorant_prop.)

\__color_backend_devicen_colorants:n
\__color_backend_devicen_colorants:w
536 \cs_new:Npx \__color_backend_devicen_colorants:n #1
537 {
538   \exp_not:N \tl_if_blank:nF {#1}
539   {
540     \c_space_tl
541     << ~
542     /Colorants ~
543     << ~
544     \exp_not:N \__color_backend_devicen_colorants:w #1 ~
545     \exp_not:N \q_recursion_tail \c_space_tl
546     \exp_not:N \q_recursion_stop
547     >> ~
548   }
549 }
550 }
551 \cs_new:Npn \__color_backend_devicen_colorants:w #1 ~
552 {
553   \quark_if_recursion_tail_stop:n {#1}
554   \prop_if_in:NnT \g_color_backend_colorant_prop {#1}
555   {
556     #1 ~
557     \prop_item:Nn \g_color_backend_colorant_prop {#1} ~
558   }
559   \__color_backend_devicen_colorants:w
560 }

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

561 </dviptfm | luatex | pdftex | xetex | dvips>
562 <*dvips>

\__color_backend_select_separation:nn
\__color_backend_select_devicen:nn
563 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
564 { \__color_backend_select:n { separation ~ #1 ~ #2 } }
565 \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_select_iccbased:nn No support.
566 \cs_new_protected:Npn \__color_backend_select_iccbased:nn #1#2 { }

(End definition for \__color_backend_select_iccbased:nn.)

\__color_backend_separation_init:nnnnn
\__color_backend_separation_init:nxxnn
\__color_backend_separation_init_aux:nnnnnn
\__color_backend_separation_init_/DeviceCMYK:nnn
\__color_backend_separation_init_/DeviceGray:nnn
\__color_backend_separation_init_/DeviceRGB:nnn
\__color_backend_separation_init_Device:Nn
\__color_backend_separation_init:nnn
\__color_backend_separation_init_count:n
\__color_backend_separation_init_count:w
\__color_backend_separation_init:nnnn
\__color_backend_separation_init:w
\__color_backend_separation_init:n
\__color_backend_separation_init:nw
\__color_backend_separation_init_CIELAB:nnn

```

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.

```

569 \bool_if:NT \g__kernel_backend_header_bool
570 {
571   \exp_args:Nx \__kernel_backend_first_shipout:n
572   {
573     \exp_not:N \__color_backend_separation_init_aux:nnnnnn
574     { \exp_not:N \int_use:N \g__color_model_int }
575     {#1} {#2} {#3} {#4} {#5}
576   }
577   \prop_gput:Nxx \exp_not:N \g__color_backend_colorant_prop
578   { / \exp_not:N \str_convert_pdfname:n {#1} }
579   {
580     << ~
581     /setcolorspace ~ {} ~
582     >> ~ begin ~
583     color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
584     end
585   }
586 }
587 }
588 \cs_generate_variant:Nn \__color_backend_separation_init:nnnnn { nxx }
589 \cs_new_protected:Npn \__color_backend_separation_init_aux:nnnnnn #1#2#3#4#5#6
590 {
591   \__kernel_backend_literal:e
592   {
593     !
594     TeXDict ~ begin ~
595     /color #1
596     {
597       [ ~
598       /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
599       [ ~ #3 ~ ] ~
600       {
601         \cs_if_exist_use:cF { __color_backend_separation_init_ #3 :nnn }
602         { \__color_backend_separation_init:nnn }
603         {#4} {#5} {#6}
604       }
605       ] ~ setcolorspace
606     } ~ def ~
607     end
608   }
609 }
610 \cs_new:cpn { __color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
611 { \__color_backend_separation_init_Device:Nn 4 {#3} }
612 \cs_new:cpn { __color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
613 { \__color_backend_separation_init_Device:Nn 1 {#3} }
614 \cs_new:cpn { __color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
615 { \__color_backend_separation_init_Device:Nn 2 {#3} }
616 \cs_new:Npn \__color_backend_separation_init_Device:Nn #1#2
617 {
618   #2 ~
619   \prg_replicate:nn {#1}
620   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
621   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
622 }

```

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.

```

623 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3
624 {
625   \exp_args:Ne \__color_backend_separation_init:nnnn
626   { \__color_backend_separation_init_count:n {#2} }
627   {#1} {#2} {#3}
628 }
629 \cs_new:Npn \__color_backend_separation_init_count:n #1
630 { \int_eval:n { 0 \__color_backend_separation_init_count:w #1 ~ \s__color_stop } }
631 \cs_new:Npn \__color_backend_separation_init_count:w #1 ~ #2 \s__color_stop
632 {
633   +1
634   \tl_if_blank:nF {#2}
635   { \__color_backend_separation_init_count:w #2 \s__color_stop }
636 }

```

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.

```

637 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
638 {
639   \__color_backend_separation_init:w #3 ~ \s__color_stop #4 ~ \s__color_stop
640   \prg_replicate:nn {#1}
641   {
642     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
643     \int_eval:n { 3 * #1 } ~ index ~ mul ~
644     2 ~ index ~ add ~
645     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
646   }
647   \int_step_function:nnnN {#1} { -1 } { 1 }
648   \__color_backend_separation_init:n
649   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
650   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
651   \tl_if_blank:nF {#2}
652   { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
653 }
654 \cs_new:Npn \__color_backend_separation_init:w
655 #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
656 {
657   #1 ~ #3 ~ 0 ~
658   \tl_if_blank:nF {#2}
659   { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }

```

```

660 }
661 \cs_new:Npn \__color_backend_separation_init:n #1
662 { \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.

```

663 \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s_color_stop
664 {
665     #2 ~ #3 ~
666     2 ~ index ~ 2 ~ index ~ lt ~
667     { ~ pop ~ exch ~ pop ~ } ~
668     { ~
669         2 ~ index ~ 1 ~ index ~ gt ~
670         { ~ exch ~ pop ~ exch ~ pop ~ } ~
671         { ~ pop ~ pop ~ } ~
672         ifelse ~
673     }
674     ifelse ~
675     #1 ~ 1 ~ roll ~
676     \tl_if_blank:nF {#4}
677     { \__color_backend_separation_init:nw {#1} #4 \s_color_stop }
678 }

```

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.

```

679 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnn #1#2#3
680 {
681     \__color_backend_separation_init:nxxxnn
682     {#2}
683     {
684         /CIEBasedABC ~
685         << ~
686         /RangeABC ~ [ ~ \c__color_model_range_CIELAB_tl \c_space_tl ] ~
687         /DecodeABC ~
688         [ ~
689             { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
690             { ~ 500 ~ div ~ } ~ bind ~
691             { ~ 200 ~ div ~ } ~ bind ~
692         ] ~
693         /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
694         /DecodeLMN ~
695         [ ~
696         { ~
697             dup ~ 6 ~ 29 ~ div ~ ge ~
698             { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
699             { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
700             ifelse ~
701             0.9505 ~ mul ~
702         } ~ bind ~
703         { ~
704             dup ~ 6 ~ 29 ~ div ~ ge ~
705             { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
706             { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
707             ifelse ~

```

```

708         } ~ bind ~
709         { ~
710             dup ~ 6 ~ 29 ~ div ~ ge ~
711             { ~ dup ~ dup ~ mul ~ mul ~ } ~
712             { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
713             ifelse ~
714             1.0890 ~ mul ~
715         } ~ bind
716     ] ~
717     /WhitePoint ~
718     [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
719     >>
720 }
721 { \c__color_model_range_CIELAB_tl }
722 { 100 ~ 0 ~ 0 }
723 {#3}
724 }

```

(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.

```

725 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
726 {
727     \_kernel_backend_literal:e
728     {
729         !
730         TeXDict ~ begin ~
731         /color \int_use:N \g__color_model_int
732         {
733             [ ~
734                 /DeviceN ~
735                 [ ~ #1 ~ ] ~
736                 #2 ~
737                 { ~ #3 ~ } ~
738                 \_color_backend_devicen_colorants:n {#1}
739             ] ~ setcolorspace
740         } ~ def ~
741     end
742 }
743 }

```

(End definition for _color_backend_devicen_init:nnn.)

_color_backend_iccbased_init:nnn No support at present.

```

744 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3 { }

```

(End definition for _color_backend_iccbased_init:nnn.)

```

745 </dvips>
746 <*dvisvgm>

```

_color_backend_select_separation:nn No support at present.

```

\_color_backend_select_devicen:nn
747 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2 { }
748 \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:nnnn`
`_color_backend_separation_init_CIELAB:nnn`

No support at present.

```
749 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }
750 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnnnnn #1#2#3 { }
```

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

`_color_backend_select_iccbased:nn`

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

```
751 \cs_new_protected:Npn \_color_backend_select_iccbased:nn #1#2
752 {
753   \_kernel_backend_literal_svg:x
754   {
755     <style>
756       @color-profile ~
757       \str_if_eq:nnTF {#2} { cmyk }
758       { device-cmyk }
759       { --color \int_use:N \g__color_model_int }
760       \c_space_tl
761       {
762         src: (" #1 ")
763       }
764     </style>
765   }
766 }
```

(End definition for `_color_backend_select_iccbased:nn`.)

```
767 </dvisvgm>
768 <*dviPDFmx | luatex | pdftex | xetex>
```

`_color_backend_select_separation:nn`
`_color_backend_select_devicen:nn`
`_color_backend_select_iccbased:nn`

```
769 <*dviPDFmx | xetex>
770 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
771 { \_kernel_backend_literal:x { pdf : bc ~ \pdf_object_ref:n {#1} ~ [ #2 ] } }
772 </dviPDFmx | xetex>
773 <*luatex | pdftex>
774 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
775 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
776 </luatex | pdftex>
777 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
778 \cs_new_eq:NN \_color_backend_select_iccbased:nn \_color_backend_select_separation:nn
```

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

`_color_backend_init_resource:n`

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

```
779 \cs_new_protected:Npn \_color_backend_init_resource:n #1
780 {
781   <*luatex | pdftex>
782   \bool_lazy_and:nnT
783   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
```

```

784     { \pdfmanagement_if_active_p: }
785     {
786         \use:x
787         {
788             \pdfmanagement_add:nnn
789             { Page / Resources / ColorSpace }
790             { #1 }
791             { \pdf_object_ref_last: }
792         }
793     }
794 </luatex | pdftex>
795 }

```

(End definition for _color_backend_init_resource:n.)

```

\_color_backend_separation_init:nnnnn
\_color_backend_separation_init:nn
\_color_backend_separation_init_CIELAB:nnn

```

Initialising the PDF structures needs two parts: creating an object containing the “real” name of the Separation, then adding a reference to that to each page. We use a separate object for the tint transformation following the model in the PDF reference. The object here for the color needs to be named as that way it’s accessible to dvipdfmx/X_qTeX.

```

796 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5
797 {
798     \pdf_object_unnamed_write:nx { dict }
799     {
800         /FunctionType ~ 2
801         /Domain ~ [0 ~ 1]
802         \tl_if_blank:nF {#3} { /Range ~ [#3] }
803         /C0 ~ [#4] ~
804         /C1 ~ [#5] /N ~ 1
805     }
806     \exp_args:Nx \_color_backend_separation_init:nn
807     { \str_convert_pdfname:n {#1} } {#2}
808     \_color_backend_init_resource:n { color \int_use:N \g__color_model_int }
809 }
810 \cs_new_protected:Npn \_color_backend_separation_init:nn #1#2
811 {
812     \use:x
813     {
814         \pdf_object_new:n { color \int_use:N \g__color_model_int }
815         \pdf_object_write:nnn { color \int_use:N \g__color_model_int } { array }
816         { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
817     }
818     \prop_gput:Nnx \g__color_backend_colorant_prop { /#1 }
819     { \pdf_object_ref_last: }
820 }

```

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

```

821 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
822 {
823     \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
824     {
825         \pdf_object_new:n { __color_illuminant_CIELAB_ #1 }
826         \pdf_object_write:nnx { __color_illuminant_CIELAB_ #1 } { array }
827         {

```

```

828         /Lab ~
829         <<
830         /WhitePoint ~
831         [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
832         /Range ~ [ \c__color_model_range_CIELAB_tl ]
833         >>
834     }
835 }
836 \__color_backend_separation_init:nnnnn
837 {#2}
838 { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
839 { \c__color_model_range_CIELAB_tl }
840 { 100 ~ 0 ~ 0 }
841 {#3}
842 }

```

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

__color_backend_devicen_init:nnn
__color_backend_devicen_init:w

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

```

843 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
844 {
845     \pdf_object_unnamed_write:nx { stream }
846     {
847         {
848             /FunctionType ~ 4 ~
849             /Domain ~
850             [ ~
851                 \prg_replicate:nn
852                 { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
853                 { 0 ~ 1 ~ }
854             ] ~
855             /Range ~
856             [ ~
857                 \str_case:nn {#2}
858                 {
859                     { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
860                     { /DeviceGray } { 0 ~ 1 }
861                     { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
862                 } ~
863             ]
864         }
865         { {#3} }
866     }
867     \use:x
868     {
869         \pdf_object_new:n { color \int_use:N \g__color_model_int }
870         \pdf_object_write:nnn { color \int_use:N \g__color_model_int } { array }
871         {
872             /DeviceN ~
873             [ ~ #1 ~ ] ~
874             #2 ~
875             \pdf_object_ref_last:

```



```

876         \_color_backend_devicen_colorants:n {#1}
877     }
878 }
879 \_color_backend_init_resource:n { color \int_use:N \g_color_model_int }
880 }
881 \cs_new:Npn \_color_backend_devicen_init:w #1 ~ #2 \s_color_stop
882 {
883     + 1
884     \tl_if_blank:nF {#2}
885     { \_color_backend_devicen_init:w #2 \s_color_stop }
886 }

```

(End definition for _color_backend_devicen_init:nnn and _color_backend_devicen_init:w.)

_color_backend_iccbased_init:nnn Lots of data to save here: we only want to do that once per file, so track it by name.

```

887 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3
888 {
889     \pdf_object_if_exist:nF { __color_icc_ #1 }
890     {
891         \pdf_object_new:n { __color_icc_ #1 }
892         \pdf_object_write:nnx { __color_icc_ #1 } { fstream }
893         {
894             {
895                 /N ~ \exp_not:n { #2 } ~
896                 \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
897             }
898             {#1}
899         }
900     }
901     \pdf_object_unnamed_write:nx { array }
902     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
903     \_color_backend_init_resource:n { color \int_use:N \g_color_model_int }
904 }

```

(End definition for _color_backend_iccbased_init:nnn.)

_color_backend_iccbased_device:nnn This is very similar to setting up a color space: the only part we add to the page resources differently.

```

905 \cs_new_protected:Npn \_color_backend_iccbased_device:nnn #1#2#3
906 {
907     \pdf_object_if_exist:nF { __color_icc_ #1 }
908     {
909         \pdf_object_new:n { __color_icc_ #1 }
910         \pdf_object_write:nnn { __color_icc_ #1 } { fstream }
911         {
912             { /N ~ #3 }
913             {#1}
914         }
915     }
916     \pdf_object_unnamed_write:nx { array }
917     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
918     \_color_backend_init_resource:n { Default #2 }
919 }

```

(End definition for _color_backend_iccbased_device:nnn.)

```

920 </dvipdfmx | luatex | pdftex | xetex>

```

3.4 Fill and stroke color

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

921 \langle *dvipdfmx | xetex \rangle

```

    \_color_backend_fill:n
\_color_backend_fill_cmyk:n
\_color_backend_fill_gray:n
\_color_backend_fill_rgb:n
    \_color_backend_stroke:n
        \_color_backend_stroke_cmyk:n
        \_color_backend_stroke_gray:n
        \_color_backend_stroke_rgb:n
922 \cs_new_protected:Npn \_color_backend_fill:n #1
923 { \_kernel_backend_literal:n { pdf : bc ~ fill ~ [ #1 ] } }
924 \cs_new_eq:NN \_color_backend_fill_cmyk:n \_color_backend_fill:n
925 \cs_new_eq:NN \_color_backend_fill_gray:n \_color_backend_fill:n
926 \cs_new_eq:NN \_color_backend_fill_rgb:n \_color_backend_fill:n
927 \cs_new_protected:Npn \_color_backend_stroke:n #1
928 { \_kernel_backend_literal:n { pdf : bc ~ stroke ~ [ #1 ] } }
929 \cs_new_eq:NN \_color_backend_stroke_cmyk:n \_color_backend_stroke:n
930 \cs_new_eq:NN \_color_backend_stroke_gray:n \_color_backend_stroke:n
931 \cs_new_eq:NN \_color_backend_stroke_rgb:n \_color_backend_stroke:n

```

(End definition for _color_backend_fill:n and others.)

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
    \_color_backend_fill_devicen:nn
    \_color_backend_stroke_devicen:nn
932 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
933 {
934     \_kernel_backend_literal:x
935     { pdf : bc ~ fill ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
936 }
937 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
938 {
939     \_kernel_backend_literal:x
940     { pdf : bc ~ stroke ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
941 }
942 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
943 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

(End definition for _color_backend_fill_separation:nn and others.)

```

\_color_backend_fill_reset:
    \_color_backend_stroke_reset:
944 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
945 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:

```

(End definition for _color_backend_fill_reset: and _color_backend_stroke_reset:.)

946 \langle /dvipdfmx | xetex \rangle

947 \langle *luatex | pdftex \rangle

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

```

948 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
949 { \_color_backend_fill:n { #1 ~ k } }

```

```

    \_color_backend_stroke_cmyk:n
    \_color_backend_stroke_gray:n
    \_color_backend_stroke_rgb:n
\_color_backend_stroke:n

```

```

950 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
951 { \__color_backend_fill:n { #1 ~ g } }
952 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
953 { \__color_backend_fill:n { #1 ~ rg } }
954 \cs_new_protected:Npn \__color_backend_fill:n #1
955 {
956   \tl_set:Nn \l__color_backend_fill_tl {#1}
957   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
958   { #1 ~ \l__color_backend_stroke_tl }
959 }
960 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
961 { \__color_backend_stroke:n { #1 ~ K } }
962 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
963 { \__color_backend_stroke:n { #1 ~ G } }
964 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
965 { \__color_backend_stroke:n { #1 ~ RG } }
966 \cs_new_protected:Npn \__color_backend_stroke:n #1
967 {
968   \tl_set:Nn \l__color_backend_stroke_tl {#1}
969   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
970   { \l__color_backend_fill_tl \c_space_tl #1 }
971 }

```

(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
972 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
973 { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
974 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
975 { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
976 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
977 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

\__color_backend_fill_reset:
\__color_backend_stroke_reset:
978 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
979 \cs_new_eq:NN \__color_backend_stroke_reset: \__color_backend_reset:

```

(End definition for __color_backend_fill_reset: and __color_backend_stroke_reset:.)

980 </luatex | pdftex>

981 <*dvips>

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

```

982 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
983 { \__color_backend_fill:n { cmyk ~ #1 } }
984 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
985 { \__color_backend_fill:n { gray ~ #1 } }
986 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
987 { \__color_backend_fill:n { rgb ~ #1 } }
988 \cs_new_protected:Npn \__color_backend_fill:n #1
989 {
990   \__kernel_backend_literal:n { color~push~ #1 }
991 }

```

```

992 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
993   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
994 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
995   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
996 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
997   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }

```

(End definition for __color_backend_fill_cmyk:n and others.)

```

\__color_backend_fill_separation:nn
\__color_backend_stroke_separation:nn
\__color_backend_fill_devicen:nn
\__color_backend_stroke_devicen:nn
998 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
999   { \__color_backend_fill:n { separation ~ #1 ~ #2 } }
1000 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1001   { \__kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1002 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1003 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

\__color_backend_fill_reset:
\__color_backend_stroke_reset:
1004 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1005 \cs_new_protected:Npn \__color_backend_stroke_reset: { }
(End definition for \__color_backend_fill_reset: and \__color_backend_stroke_reset:.)
1006 </dvips>
1007 <*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
\__color_backend_fill_rgb:n
\__color_backend_fill:n
1008 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1009   { \__color_backend_fill:n { cmyk ~ #1 } }
1010 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1011   { \__color_backend_fill:n { gray ~ #1 } }
1012 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1013   { \__color_backend_fill:n { rgb ~ #1 } }
1014 \cs_new_protected:Npn \__color_backend_fill:n #1
1015   {
1016     \__kernel_backend_literal:n { color~push~ #1 }
1017   }

```

(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
\__color_backend_stroke_gray:n
\__color_backend_stroke_gray_aux:n
\__color_backend_stroke_rgb:n
\__color_backend_stroke_rgb:w
\__color_backend:nnn
1018 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1019   { \__color_backend_cmyk:w #1 \s__color_stop }
1020 \cs_new_protected:Npn \__color_backend_stroke_cmyk:w
1021   #1 ~ #2 ~ #3 ~ #4 \s__color_stop
1022   {
1023     \use:x
1024     {
1025       \__color_backend:nnn
1026       { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1027       { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1028       { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }

```

```

1029     }
1030 }
1031 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1032 {
1033   \use:x
1034   {
1035     \__color_backend_stroke_gray_aux:n
1036     { \fp_eval:n { 100 * (#1) } }
1037   }
1038 }
1039 \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1040 { \__color_backend:nnn {#1} {#1} {#1} }
1041 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1042 { \__color_backend_rgb:w #1 \s__color_stop }
1043 \cs_new_protected:Npn \__color_backend_stroke_rgb:w
1044 #1 ~ #2 ~ #3 \s__color_stop
1045 {
1046   \use:x
1047   {
1048     \__color_backend:nnn
1049     { \fp_eval:n { 100 * (#1) } }
1050     { \fp_eval:n { 100 * (#2) } }
1051     { \fp_eval:n { 100 * (#3) } }
1052   }
1053 }
1054 \cs_new_protected:Npx \__color_backend:nnn #1#2#3
1055 {
1056   \__kernel_backend_scope:n
1057   {
1058     stroke =
1059     "
1060     rgb
1061     (
1062       #1 \c_percent_str ,
1063       #2 \c_percent_str ,
1064       #3 \c_percent_str
1065     )
1066     "
1067   }
1068 }

```

(End definition for __color_backend_stroke_cmyk:n and others.)

At present, these are no-ops.

```

\__color_backend_fill_separation:nn
\__color_backend_stroke_separation:nn
\__color_backend_fill_devicen:nn
\__color_backend_stroke_devicen:nn
1069 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1070 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2 { }
1071 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1072 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

\__color_backend_fill_reset:
\__color_backend_stroke_reset:
1073 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1074 \cs_new_protected:Npn \__color_backend_stroke_reset: { }

```

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

`_color_backend_devicen_init:nnn` No support at present.

`_color_backend_iccbased_init:nnn` 1075 `\cs_new_protected:Npn _color_backend_devicen_init:nnn #1#2#3 { }`
1076 `\cs_new_protected:Npn _color_backend_iccbased_init:nnn #1#2#3 { }`

(End definition for `_color_backend_devicen_init:nnn` and `_color_backend_iccbased_init:nnn`.)

1077 `</divisvgn>`

1078 `</package>`

3.5 Font handling integration

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

1079 `<*lua>`

1080 `local l = lpeg`

1081 `local spaces = l.P' '^0`

1082 `local digit16 = l.R('09', 'af', 'AF')`

1083

1084 `local octet = digit16 * digit16 / function(s)`

1085 `return string.format('%.3g ', tonumber(s, 16) / 255)`

1086 `end`

1087

1088 `if luaotfload and luaotfload.set_transparent_colorstack then`

1089 `local htmlcolor = l.Cs(octet * octet * octet * -1 * l.Cc'rg')`

1090 `local color_export = {`

1091 `token.create'tex_endlocalcontrol:D',`

1092 `token.create'tex_hpack:D',`

1093 `token.new(0, 1),`

1094 `token.create'color_export:nnN',`

1095 `token.new(0, 1),`

1096 `'',`

1097 `token.new(0, 2),`

1098 `token.new(0, 1),`

1099 `'backend',`

1100 `token.new(0, 2),`

1101 `token.create'l_tmpa_tl',`

1102 `token.create'exp_after:wN',`

1103 `token.create'__color_select:nn',`

1104 `token.create'l_tmpa_tl',`

1105 `token.new(0, 2),`

1106 `}`

1107 `local group_end = token.create'group_end:'`

1108 `local value = (1 - l.P'}')^0`

1109 `luatexbase.add_to_callback('luaotfload.parse_color', function (value)`

1110 `% Also allow HTML colors to preserve compatibility`

1111 `local html = htmlcolor:match(value)`

1112 `if html then return html end`

1113

1114 `tex.runtoks(function()`

1115 `token.get_next()`

1116 `color_export[6] = value`

```

1117     tex.sprint(-2, color_export)
1118   end)
1119   local list = token.scan_list()
1120   if not list.head or list.head.next
1121     or list.head.subtype ~= node.subtype'pdf_colorstack' then
1122     error'Unexpected backend behavior'
1123   end
1124   local cmd = list.head.data
1125   node.free(list)
1126   return cmd
1127 end, 'l3color')
1128 end
1129 </lua>
1130 <*luatex>
1131 <*package>
1132 \lua_load_module:n {l3backend-luatex}
1133 </package>
1134 </luatex>

```

4 l3backend-draw implementation

```

1135 <*package>
1136 <@=draw>

```

4.1 dvips backend

```

1137 <*dvips>

```

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

```

1138 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
1139 \cs_generate_variant:Nn \_draw_backend_literal:n { x }

```

(End definition for `_draw_backend_literal:n`)

`_draw_backend_begin:` The `ps::[begin]` special here deals with positioning but allows us to continue on to a matching `ps::[end]`: contrast with `ps:`, which positions but where we can't split material between separate calls. The `@beginspecial/@endspecial` pair are from `special.pro` and correct the scale and *y*-axis direction. 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.

```

1140 \cs_new_protected:Npn \_draw_backend_begin:
1141 {
1142   \_kernel_backend_literal:n { ps::[begin] }
1143   \_draw_backend_literal:n { @beginspecial }
1144 }
1145 \cs_new_protected:Npn \_draw_backend_end:
1146 {
1147   \_draw_backend_literal:n { @endspecial }
1148   \_kernel_backend_literal:n { ps::[end] }
1149 }

```

(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
`_draw_backend_scope_end:` the graphic state has to be sent to the stack.

```

1150 \cs_new_protected:Npn \_draw_backend_scope_begin:
1151   { \_draw_backend_literal:n { save } }
1152 \cs_new_protected:Npn \_draw_backend_scope_end:
1153   { \_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
`_draw_backend_lineto:nn` the need to convert to bp. Notice that x-type expansion is included here to ensure that
`_draw_backend_rectangle:nnnn` any variable values are forced to literals before any possible caching. There is no native
`_draw_backend_curveto:nnnnnn` rectangular path command (without also clipping, filling or stroking), so that task is
done using a small amount of PostScript.

```

1154 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1155   {
1156     \_draw_backend_literal:x
1157     {
1158       \dim_to_decimal_in_bp:n {#1} ~
1159       \dim_to_decimal_in_bp:n {#2} ~ moveto
1160     }
1161   }
1162 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1163   {
1164     \_draw_backend_literal:x
1165     {
1166       \dim_to_decimal_in_bp:n {#1} ~
1167       \dim_to_decimal_in_bp:n {#2} ~ lineto
1168     }
1169   }
1170 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1171   {
1172     \_draw_backend_literal:x
1173     {
1174       \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1175       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1176       moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1177     }
1178   }
1179 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1180   {
1181     \_draw_backend_literal:x
1182     {
1183       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1184       \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1185       \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1186       curveto
1187     }
1188   }

```

(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: 1189 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
\g__draw_draw_eor_bool 1190 { \bool_gset_true:N \g__draw_draw_eor_bool }
1191 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1192 { \bool_gset_false:N \g__draw_draw_eor_bool }
1193 \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_stroke: 1194 \cs_new_protected:Npn \_draw_backend_closepath:
\_draw_backend_fill: 1195 { \_draw_backend_literal:n { closepath } }
\_draw_backend_closestroke: 1196 \cs_new_protected:Npn \_draw_backend_stroke:
\_draw_backend_fillstroke: 1197 {
\_draw_backend_clip: 1198 \_draw_backend_literal:n { gsave }
1199 \_draw_backend_literal:n { color.sc }
1200 \_draw_backend_literal:n { stroke }
1201 \_draw_backend_literal:n { grestore }
1202 \bool_if:NT \g__draw_draw_clip_bool
1203 {
1204 \_draw_backend_literal:x
1205 {
1206 \bool_if:NT \g__draw_draw_eor_bool { eo }
1207 clip
1208 }
1209 }
1210 \_draw_backend_literal:n { newpath }
1211 \bool_gset_false:N \g__draw_draw_clip_bool
1212 }
1213 \cs_new_protected:Npn \_draw_backend_closestroke:
1214 {
1215 \_draw_backend_closepath:
1216 \_draw_backend_stroke:
1217 }
1218 \cs_new_protected:Npn \_draw_backend_fill:
1219 {
1220 \_draw_backend_literal:x
1221 {
1222 \bool_if:NT \g__draw_draw_eor_bool { eo }
1223 fill
1224 }
1225 \bool_if:NT \g__draw_draw_clip_bool
1226 {
1227 \_draw_backend_literal:x
1228 {
1229 \bool_if:NT \g__draw_draw_eor_bool { eo }
1230 clip
1231 }

```

```

1232     }
1233     \__draw_backend_literal:n { newpath }
1234     \bool_gset_false:N \g__draw_draw_clip_bool
1235   }
1236   \cs_new_protected:Npn \__draw_backend_fillstroke:
1237   {
1238     \__draw_backend_literal:x
1239     {
1240       \bool_if:NT \g__draw_draw_eor_bool { eo }
1241       fill
1242     }
1243     \__draw_backend_literal:n { gsave }
1244     \__draw_backend_literal:n { color.sc }
1245     \__draw_backend_literal:n { stroke }
1246     \__draw_backend_literal:n { grestore }
1247     \bool_if:NT \g__draw_draw_clip_bool
1248     {
1249       \__draw_backend_literal:x
1250       {
1251         \bool_if:NT \g__draw_draw_eor_bool { eo }
1252         clip
1253       }
1254     }
1255     \__draw_backend_literal:n { newpath }
1256     \bool_gset_false:N \g__draw_draw_clip_bool
1257   }
1258   \cs_new_protected:Npn \__draw_backend_clip:
1259   { \bool_gset_true:N \g__draw_draw_clip_bool }
1260   \bool_new:N \g__draw_draw_clip_bool
1261   \cs_new_protected:Npn \__draw_backend_discardpath:
1262   {
1263     \bool_if:NT \g__draw_draw_clip_bool
1264     {
1265       \__draw_backend_literal:x
1266       {
1267         \bool_if:NT \g__draw_draw_eor_bool { eo }
1268         clip
1269       }
1270     }
1271     \__draw_backend_literal:n { newpath }
1272     \bool_gset_false:N \g__draw_draw_clip_bool
1273   }

```

(End definition for __draw_backend_closepath: and others.)

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1274 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1275 {
1276   \__draw_backend_literal:x
1277   {
1278     [
1279       \exp_args:Nf \use:n
1280       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1281     ] ~

```

```

1282         \dim_to_decimal_in_bp:n {#2} ~ setdash
1283     }
1284 }
1285 \cs_new:Npn \__draw_backend_dash:n #1
1286 { ~ \dim_to_decimal_in_bp:n {#1} }
1287 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1288 {
1289     \__draw_backend_literal:x
1290     { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1291 }
1292 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1293 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1294 \cs_new_protected:Npn \__draw_backend_cap_but:
1295 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1296 \cs_new_protected:Npn \__draw_backend_cap_round:
1297 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1298 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1299 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1300 \cs_new_protected:Npn \__draw_backend_join_miter:
1301 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1302 \cs_new_protected:Npn \__draw_backend_join_round:
1303 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1304 \cs_new_protected:Npn \__draw_backend_join_bevel:
1305 { \__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/X_YTeX). Thus we take the shortest path available and simply dump the matrix as given.

```

1306 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1307 {
1308     \__draw_backend_literal:n
1309     { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1310 }

```

(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 T_EX 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).

```

1311 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1312 {
1313     \__draw_backend_literal:n { @endspecial }

```

```

1314 \__draw_backend_literal:n { [end] }
1315 \__draw_backend_literal:n { [begin] }
1316 \__draw_backend_literal:n { save }
1317 \__draw_backend_literal:n { currentpoint }
1318 \__draw_backend_literal:n { currentpoint~translate }
1319 \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1320 \__draw_backend_cm:nnnn { #2 } { #3 } { #4 } { #5 }
1321 \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1322 \__draw_backend_literal:n { neg~exch~neg~exch~translate }
1323 \__draw_backend_literal:n { [end] }
1324 \hbox_overlap_right:n { \box_use:N #1 }
1325 \__draw_backend_literal:n { [begin] }
1326 \__draw_backend_literal:n { restore }
1327 \__draw_backend_literal:n { [end] }
1328 \__draw_backend_literal:n { [begin] }
1329 \__draw_backend_literal:n { @beginspecial }
1330 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1331 </dvips>

```

4.2 LuaTeX, pdfTeX, dvipdfmx and XeTeX

LuaTeX, pdfTeX, dvipdfmx and XeTeX directly produce PDF output and understand a shared set of specials for drawing commands.

```

1332 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

```

\__draw_backend_literal:n Pass data through using a dedicated interface.
\__draw_backend_literal:x
1333 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1334 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for __draw_backend_literal:n.)

```

\__draw_backend_begin: No special requirements here, so simply set up a drawing scope.
\__draw_backend_end:
1335 \cs_new_protected:Npn \__draw_backend_begin:
1336 { \__draw_backend_scope_begin: }
1337 \cs_new_protected:Npn \__draw_backend_end:
1338 { \__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:
1339 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1340 \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
\__draw_backend_lineto:nn
  \_draw_backend_curveto:nnnnnn
  \_draw_backend_rectangle:nnnn

```

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

```

1341 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1342 {
1343   \__draw_backend_literal:x
1344   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1345 }
1346 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1347 {
1348   \__draw_backend_literal:x
1349   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1350 }
1351 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1352 {
1353   \__draw_backend_literal:x
1354   {
1355     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1356     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1357     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1358     c
1359   }
1360 }
1361 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1362 {
1363   \__draw_backend_literal:x
1364   {
1365     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1366     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1367     re
1368   }
1369 }

```

(End definition for __draw_backend_moveto:nn and others.)

```

\_draw_backend_evenodd_rule:
\_draw_backend_nonzero_rule:
\g__draw_draw_eor_bool

```

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

```

1370 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1371 { \bool_gset_true:N \g__draw_draw_eor_bool }
1372 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1373 { \bool_gset_false:N \g__draw_draw_eor_bool }
1374 \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:
  \__draw_backend_stroke:
\__draw_backend_closestroke:
  \__draw_backend_fill:
\__draw_backend_fillstroke:
  \__draw_backend_clip:
\__draw_backend_discardpath:

```

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

```

1375 \cs_new_protected:Npn \__draw_backend_closepath:
1376 { \__draw_backend_literal:n { h } }
1377 \cs_new_protected:Npn \__draw_backend_stroke:
1378 { \__draw_backend_literal:n { S } }
1379 \cs_new_protected:Npn \__draw_backend_closestroke:
1380 { \__draw_backend_literal:n { s } }
1381 \cs_new_protected:Npn \__draw_backend_fill:
1382 {
1383   \__draw_backend_literal:x

```

```

1384     { f \bool_if:NT \g__draw_draw_eor_bool * }
1385   }
1386   \cs_new_protected:Npn \__draw_backend_fillstroke:
1387   {
1388     \__draw_backend_literal:x
1389     { B \bool_if:NT \g__draw_draw_eor_bool * }
1390   }
1391   \cs_new_protected:Npn \__draw_backend_clip:
1392   {
1393     \__draw_backend_literal:x
1394     { W \bool_if:NT \g__draw_draw_eor_bool * }
1395   }
1396   \cs_new_protected:Npn \__draw_backend_discardpath:
1397   { \__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:
1398 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1399 {
1400   \__draw_backend_literal:x
1401   {
1402     [
1403       \exp_args:Nf \use:n
1404       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1405     ] ~
1406     \dim_to_decimal_in_bp:n {#2} ~ d
1407   }
1408 }
1409 \cs_new:Npn \__draw_backend_dash:n #1
1410 { ~ \dim_to_decimal_in_bp:n {#1} }
1411 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1412 {
1413   \__draw_backend_literal:x
1414   { \dim_to_decimal_in_bp:n {#1} ~ w }
1415 }
1416 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1417 { \__draw_backend_literal:x { #1 ~ M } }
1418 \cs_new_protected:Npn \__draw_backend_cap_butt:
1419 { \__draw_backend_literal:n { 0 ~ J } }
1420 \cs_new_protected:Npn \__draw_backend_cap_round:
1421 { \__draw_backend_literal:n { 1 ~ J } }
1422 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1423 { \__draw_backend_literal:n { 2 ~ J } }
1424 \cs_new_protected:Npn \__draw_backend_join_miter:
1425 { \__draw_backend_literal:n { 0 ~ j } }
1426 \cs_new_protected:Npn \__draw_backend_join_round:
1427 { \__draw_backend_literal:n { 1 ~ j } }
1428 \cs_new_protected:Npn \__draw_backend_join_bevel:
1429 { \__draw_backend_literal:n { 2 ~ j } }

```

(End definition for __draw_backend_dash_pattern:nn and others.)

__draw_backend_cm: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/XYTeX`, we can to decompose the matrix into rotations and a scaling, then use those operations as they are handled by the backend. (There is backend support for matrix operations in `dvipdfmx/XYTeX`, but as a matched pair so not suitable for the “stand alone” transformation set up here.) The specials used here are from `xdvipdfmx` originally: they are well-tested, but probably equivalent to the `pdf`: versions!

```

1430 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1431 {
1432   <*luatex | pdftex>
1433     \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1434   </luatex | pdftex>
1435   <*dvipdfmx | xetex>
1436     \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1437     \__draw_backend_cm_aux:nnnn
1438   </dvipdfmx | xetex>
1439 }
1440 <*dvipdfmx | xetex>
1441 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1442 {
1443   \__kernel_backend_literal:x
1444   {
1445     x:rotate~
1446     \fp_compare:nNnTF {#1} = \c_zero_fp
1447       { 0 }
1448       { \fp_eval:n { round ( -#1 , 5 ) } }
1449   }
1450   \__kernel_backend_literal:x
1451   {
1452     x:scale~
1453     \fp_eval:n { round ( #2 , 5 ) } ~
1454     \fp_eval:n { round ( #3 , 5 ) }
1455   }
1456   \__kernel_backend_literal:x
1457   {
1458     x:rotate~
1459     \fp_compare:nNnTF {#4} = \c_zero_fp
1460       { 0 }
1461       { \fp_eval:n { round ( -#4 , 5 ) } }
1462   }
1463 }
1464 </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.

```

1465 <*dvipdfmx | xetex>
1466 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1467 {
1468   \use:x
1469   {
1470     \__draw_backend_cm_decompose_auxi:nnnnN
1471     { \fp_eval:n { (#1 + #4) / 2 } }
1472     { \fp_eval:n { (#1 - #4) / 2 } }
1473     { \fp_eval:n { (#3 + #2) / 2 } }
1474     { \fp_eval:n { (#3 - #2) / 2 } }
1475   }
1476   #5
1477 }
1478 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1479 {
1480   \use:x
1481   {
1482     \__draw_backend_cm_decompose_auxii:nnnnN
1483     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1484     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1485     { \fp_eval:n { atand ( #3 , #2 ) } }
1486     { \fp_eval:n { atand ( #4 , #1 ) } }
1487   }
1488   #5
1489 }
1490 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1491 {
1492   \use:x
1493   {
1494     \__draw_backend_cm_decompose_auxiii:nnnnN
1495     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1496     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1497     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1498     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1499   }

```



```

1500         #5
1501     }
1502     \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1503     {
1504         \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1505         { #5 {#1} {#2} {#3} {#4} }
1506         { #5 {#1} {#3} {#2} {#4} }
1507     }
1508     </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.

```

1509     \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1510     {
1511         \__kernel_backend_scope_begin:
1512         < *luatex | pdftex >
1513         \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1514         < /luatex | pdftex >
1515         < *dvipdfmx | xetex >
1516         \__kernel_backend_literal:n
1517         { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1518         < /dvipdfmx | xetex >
1519         \hbox_overlap_right:n { \box_use:N #1 }
1520         < *dvipdfmx | xetex >
1521         \__kernel_backend_literal:n { pdf:etrans }
1522         < /dvipdfmx | xetex >
1523         \__kernel_backend_scope_end:
1524     }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1525     < /dvipdfmx | luatex | pdftex | xetex >

```

4.3 dvisvgm backend

```

1526     < *dvisvgm >

```

The same as the more general literal call.

__draw_backend_literal:n
__draw_backend_literal:x

```

1527     \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1528     \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for __draw_backend_literal:n.)

__draw_backend_scope_begin:
__draw_backend_scope_end:

Use the backend-level scope mechanisms.

```

1529     \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1530     \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_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

```

1531 \cs_new_protected:Npn __draw_backend_begin:
1532 {
1533   __kernel_backend_scope_begin:
1534   __kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1535 }
1536 \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_backend_path_tl`

```

1537 \cs_new_protected:Npn __draw_backend_moveto:nn #1#2
1538 {
1539   __draw_backend_add_to_path:n
1540   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1541 }
1542 \cs_new_protected:Npn __draw_backend_lineto:nn #1#2
1543 {
1544   __draw_backend_add_to_path:n
1545   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1546 }
1547 \cs_new_protected:Npn __draw_backend_rectangle:nnnn #1#2#3#4
1548 {
1549   __draw_backend_add_to_path:n
1550   {
1551     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1552     h ~ \dim_to_decimal:n {#3} ~
1553     v ~ \dim_to_decimal:n {#4} ~
1554     h ~ \dim_to_decimal:n { -#3 } ~
1555     Z
1556   }
1557 }
1558 \cs_new_protected:Npn __draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1559 {
1560   __draw_backend_add_to_path:n
1561   {
1562     C ~
1563     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1564     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1565     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1566   }
1567 }
1568 \cs_new_protected:Npn __draw_backend_add_to_path:n #1
1569 {
1570   \tl_gset:Nx \g__draw_backend_path_tl
1571   {
1572     \g__draw_backend_path_tl
1573     \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1574     #1

```

```

1575     }
1576   }
1577   \tl_new:N \g__draw_backend_path_tl

```

(End definition for _draw_backend_moveto:nn and others.)

```

\_draw_backend_evenodd_rule:
\_draw_backend_nonzero_rule:

```

The fill rules here have to be handled as scopes.

```

1578 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1579   { \_kernel_backend_scope:n { fill-rule="evenodd" } }
1580 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1581   { \_kernel_backend_scope:n { fill-rule="nonzero" } }

```

(End definition for _draw_backend_evenodd_rule: and _draw_backend_nonzero_rule:.)

```

\_draw_backend_path:n
\_draw_backend_closepath:
\_draw_backend_stroke:
\_draw_backend_closestroke:
\_draw_backend_fill:
\_draw_backend_fillstroke:
\_draw_backend_clip:
\_draw_backend_discardpath:
\g__draw_draw_clip_bool
\g__draw_draw_path_int

```

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

```

1582 \cs_new_protected:Npn \_draw_backend_closepath:
1583   { \_draw_backend_add_to_path:n { Z } }
1584 \cs_new_protected:Npn \_draw_backend_path:n #1
1585   {
1586     \bool_if:NTF \g__draw_draw_clip_bool
1587     {
1588       \int_gincr:N \g__kernel_clip_path_int
1589       \_draw_backend_literal:x
1590       {
1591         < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1592         { ?nl }
1593         <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1594         < /clipPath > { ? nl }
1595         <
1596         use~xlink:href =
1597         "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1598         #1
1599         />
1600       }
1601       \_kernel_backend_scope:x
1602       {
1603         clip-path =
1604         "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1605       }
1606     }
1607     {
1608       \_draw_backend_literal:x
1609       { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1610     }
1611     \tl_gclear:N \g__draw_backend_path_tl
1612     \bool_gset_false:N \g__draw_draw_clip_bool
1613   }
1614 \int_new:N \g__draw_backend_path_int
1615 \cs_new_protected:Npn \_draw_backend_stroke:
1616   { \_draw_backend_path:n { style="fill:none" } }

```

```

1617 \cs_new_protected:Npn \__draw_backend_closestroke:
1618 {
1619   \__draw_backend_closepath:
1620   \__draw_backend_stroke:
1621 }
1622 \cs_new_protected:Npn \__draw_backend_fill:
1623 { \__draw_backend_path:n { style="stroke:none" } }
1624 \cs_new_protected:Npn \__draw_backend_fillstroke:
1625 { \__draw_backend_path:n { } }
1626 \cs_new_protected:Npn \__draw_backend_clip:
1627 { \bool_gset_true:N \g__draw_draw_clip_bool }
1628 \bool_new:N \g__draw_draw_clip_bool
1629 \cs_new_protected:Npn \__draw_backend_discardpath:
1630 {
1631   \bool_if:NT \g__draw_draw_clip_bool
1632   {
1633     \int_gincr:N \g__kernel_clip_path_int
1634     \__draw_backend_literal:x
1635     {
1636       < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1637       { ?nl }
1638       <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1639       < /clipPath >
1640     }
1641     \__kernel_backend_scope:x
1642     {
1643       clip-path =
1644       "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1645     }
1646   }
1647   \tl_gclear:N \g__draw_path_tl
1648   \bool_gset_false:N \g__draw_draw_clip_bool
1649 }

```

(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:
1650 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1651 {
1652   \use:x
1653   {
1654     \__draw_backend_dash_aux:nn
1655     { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1656     { \dim_to_decimal:n {#2} }
1657   }
1658 }
1659 \cs_new:Npn \__draw_backend_dash:n #1
1660 { , \dim_to_decimal_in_bp:n {#1} }
1661 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1662 {
1663   \__kernel_backend_scope:x
1664   {
1665     stroke-dasharray =

```

```

1666         "
1667         \tl_if_empty:nTF {#1}
1668         { none }
1669         { \use_none:n #1 }
1670     " ~
1671     stroke-offset=" #2 "
1672 }
1673 }
1674 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1675 { \__kernel_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1676 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1677 { \__kernel_backend_scope:x { stroke-miterlimit=" #1 " } }
1678 \cs_new_protected:Npn \__draw_backend_cap_but:
1679 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1680 \cs_new_protected:Npn \__draw_backend_cap_round:
1681 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1682 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1683 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1684 \cs_new_protected:Npn \__draw_backend_join_miter:
1685 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1686 \cs_new_protected:Npn \__draw_backend_join_round:
1687 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1688 \cs_new_protected:Npn \__draw_backend_join_bevel:
1689 { \__kernel_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.

```

1690 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1691 {
1692     \__kernel_backend_scope:n
1693     {
1694         transform =
1695         " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1696     }
1697 }

```

(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.

```

1698 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1699 {
1700     \__kernel_backend_scope_begin:
1701     \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1702     \__kernel_backend_literal_svg:n
1703     {
1704         < g~
1705         stroke="none"~
1706         transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1707     }
1708 }
1709 \box_set_wd:Nn #1 { Opt }

```

```

1710     \box_set_ht:Nn #1 { Opt }
1711     \box_set_dp:Nn #1 { Opt }
1712     \box_use:N #1
1713     \__kernel_backend_literal_svg:n { </g> }
1714     \__kernel_backend_scope_end:
1715 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1716 </dvisvgm>

```

```

1717 </package>

```

5 l3backend-graphics implementation

```

1718 <*package>
1719 <@@=graphics>

```

__graphics_backend_loaded:n To deal with file load ordering. Plain users are on their own.

```

1720 \cs_new_protected:Npn \__graphics_backend_loaded:n #1
1721 {
1722     \cs_if_exist:NTF \hook_gput_code:nnn
1723     {
1724         \hook_gput_code:nnn
1725         { file / l3graphics.sty / after }
1726         { backend }
1727         {#1}
1728     }
1729     {#1}
1730 }

```

(End definition for __graphics_backend_loaded:n.)

5.1 dvips backend

```

1731 <*dvips>

```

\l_graphics_search_ext_seq

```

1732 \__graphics_backend_loaded:n
1733 { \seq_set_from_clist:Nn \l_graphics_search_ext_seq { .eps , .ps } }

```

(End definition for \l_graphics_search_ext_seq. This variable is documented on page ??.)

__graphics_backend_getbb_eps:n Simply use the generic function.

```

\__graphics_backend_getbb_ps:n
1734 \__graphics_backend_loaded:n
1735 {
1736     \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1737     \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
1738 }

```

(End definition for __graphics_backend_getbb_eps:n and __graphics_backend_getbb_ps:n.)

`_graphics_backend_include_eps:n`
`_graphics_backend_include_ps:n`

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

```

1739 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1740 {
1741   \_kernel_backend_literal:x
1742   {
1743     PSfile = #1 \c_space_tl
1744     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1745     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1746     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1747     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1748   }
1749 }
1750 \cs_new_eq:NN \_graphics_backend_include_ps:n \_graphics_backend_include_eps:n

```

(End definition for `_graphics_backend_include_eps:n` and `_graphics_backend_include_ps:n`.)

`_graphics_backend_get_pagecount:n`

```

1751 \_graphics_backend_loaded:n
1752 { \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n }

```

(End definition for `_graphics_backend_get_pagecount:n`.)

```

1753 </dvips>

```

5.2 LuaTeX and pdfTeX backends

```

1754 <*luatex | pdftex>

```

`\l_graphics_search_ext_seq`

```

1755 \_graphics_backend_loaded:n
1756 {
1757   \seq_set_from_clist:Nn
1758   \l_graphics_search_ext_seq
1759   { .pdf , .eps , .ps , .png , .jpg , .jpeg }
1760 }

```

(End definition for `\l_graphics_search_ext_seq`. This variable is documented on page ??.)

`\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.

```

1761 \tl_new:N \l__graphics_graphics_attr_tl

```

(End definition for `\l__graphics_graphics_attr_tl`.)

`_graphics_backend_getbb_jpg:n`
`_graphics_backend_getbb_jpeg:n`
`_graphics_backend_getbb_pdf:n`
`_graphics_backend_getbb_png:n`
`_graphics_backend_getbb_auxi:n`
`_graphics_backend_getbb_auxiii:n`
`_graphics_backend_dequote:w`

Getting the bounding box here requires us to box up the graphic and measure it. To deal with the difference in feature support in bitmap and vector graphics but keeping the common parts, there is a little work to do in terms of auxiliaries. The key here is to notice that we need two forms of the attributes: a “short” set to allow us to track for caching, and the full form to pass to the primitive.

```

1762 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1763 {
1764   \int_zero:N \l__graphics_page_int

```

```

1765 \tl_clear:N \l__graphics_pagebox_tl
1766 \tl_set:Nx \l__graphics_graphics_attr_tl
1767 {
1768   \tl_if_empty:NF \l__graphics_decodearray_str
1769   { :D \l__graphics_decodearray_str }
1770   \bool_if:NT \l__graphics_interpolate_bool
1771   { :I }
1772 }
1773 \tl_clear:N \l__graphics_graphics_attr_tl
1774 \__graphics_backend_getbb_auxi:n {#1}
1775 }
1776 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1777 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1778 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1779 {
1780   \tl_clear:N \l__graphics_decodearray_str
1781   \bool_set_false:N \l__graphics_interpolate_bool
1782   \tl_set:Nx \l__graphics_graphics_attr_tl
1783   {
1784     : \l__graphics_pagebox_tl
1785     \int_compare:nNnT \l__graphics_page_int > 1
1786     { :P \int_use:N \l__graphics_page_int }
1787   }
1788   \__graphics_backend_getbb_auxi:n {#1}
1789 }
1790 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1791 {
1792   \__graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1793   { \__graphics_backend_getbb_auxii:n {#1} }
1794 }

```

Measuring the graphic is done by boxing up: for PDF graphics we could use `\tex_pdfximagebbox:D`, but if doesn't work for other types. As the box always starts at (0,0) there is no need to worry about the lower-left position. Quotes need to be *removed* as LuaTeX does not like them here.

```

1795 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1796 {
1797   \exp_args:Ne \__graphics_backend_getbb_auxiii:n
1798   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1799   \int_const:cn { c__graphics_ #1 \l__graphics_graphics_attr_tl _int }
1800   { \tex_the:D \tex_pdflastximage:D }
1801   \__graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1802 }
1803 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1804 {
1805   \tex_immediate:D \tex_pdfximage:D
1806   \bool_lazy_or:nnT
1807   { \l__graphics_interpolate_bool }
1808   { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1809   {
1810     attr ~
1811     {
1812       \tl_if_empty:NF \l__graphics_decodearray_str
1813       { /Decode~[ \l__graphics_decodearray_str ] }

```



```

1814         \bool_if:NT \l__graphics_interpolate_bool
1815         { /Interpolate~true }
1816     }
1817 }
1818 \int_compare:nNnT \l__graphics_page_int > 0
1819 { page ~ \int_use:N \l__graphics_page_int }
1820 \tl_if_empty:NF \l__graphics_pagebox_tl
1821 { \l__graphics_pagebox_tl }
1822 {#1}
1823 \hbox_set:Nn \l__graphics_internal_box
1824 { \tex_pdfrefximage:D \tex_pdflastximage:D }
1825 \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1826 \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1827 }
1828 \cs_new:Npn \__graphics_backend_dequote:w #1 " #2 " #3 \s__graphics_stop {#2}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

__graphics_backend_include_jpg:n
__graphics_backend_include_jpeg:n
__graphics_backend_include_pdf:n
__graphics_backend_include_png:n

Images are already loaded for the measurement part of the code, so inclusion is straightforward, with only any attributes to worry about. The latter carry through from determination of the bounding box.

```

1829 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1830 {
1831     \tex_pdfrefximage:D
1832     \int_use:c { c__graphics_ #1 \l__graphics_graphics_attr_tl _int }
1833 }
1834 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1835 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1836 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

(End definition for __graphics_backend_include_jpg:n and others.)

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

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

```

1837 \sys_if_shell:T
1838 {
1839     \str_new:N \l__graphics_backend_dir_str
1840     \str_new:N \l__graphics_backend_name_str
1841     \str_new:N \l__graphics_backend_ext_str
1842     \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1843     {
1844         \file_parse_full_name:nNNN {#1}
1845         \l__graphics_backend_dir_str
1846         \l__graphics_backend_name_str
1847         \l__graphics_backend_ext_str
1848         \exp_args:Nx \__graphics_backend_getbb_eps:nn
1849         {
1850             \exp_args:Ne \__kernel_file_name_quote:n
1851             {
1852                 \l__graphics_backend_name_str
1853                 - \str_tail:N \l__graphics_backend_ext_str
1854                 -converted-to.pdf
1855             }

```

```

1856     }
1857     {#1}
1858   }
1859   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_backend_getbb_eps:n
1860   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1861   {
1862     \file_compare_timestamp:nNnT {#2} > {#1}
1863     {
1864       \sys_shell_now:n
1865       { repstopdf ~ #2 ~ #1 }
1866     }
1867     \tl_set:Nn \l__graphics_final_name_str {#1}
1868     \__graphics_backend_getbb_pdf:n {#1}
1869   }
1870   \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1871   {
1872     \file_parse_full_name:nNNN {#1}
1873     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1874     \exp_args:Nx \__graphics_backend_include_pdf:n
1875     {
1876       \exp_args:Ne \__kernel_file_name_quote:n
1877       {
1878         \l__graphics_backend_name_str
1879         - \str_tail:N \l__graphics_backend_ext_str
1880         -converted-to.pdf
1881       }
1882     }
1883   }
1884   \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1885 }

```

(End definition for __graphics_backend_getbb_eps:n and others.)

__graphics_backend_get_pagecount:n Simply load and store.

```

1886 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1887 {
1888   \tex_pdfximage:D {#1}
1889   \int_const:cn { c__graphics_ #1 _pages_int }
1890   { \int_use:N \tex_pdflastximagepages:D }
1891 }

```

(End definition for __graphics_backend_get_pagecount:n.)

1892 </luatex | pdftex>

5.3 dvipdfmx backend

1893 <*dvipdfmx | xetex>

\l_graphics_search_ext_seq

```

1894 \__graphics_backend_loaded:n
1895 {
1896   \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1897   { .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }
1898 }

```

(End definition for \l_graphics_search_ext_seq. This variable is documented on page ??.)

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

```

\graphics_backend_getbb_eps:n
\graphics_backend_getbb_ps:n
\graphics_backend_getbb_jpg:n
\graphics_backend_getbb_jpeg:n
\graphics_backend_getbb_pdf:n
\graphics_backend_getbb_png:n
\graphics_backend_getbb_bmp:n
1899 \__graphics_backend_loaded:n
1900 {
1901   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1902   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
1903 }
1904 <*dvipdfmx>
1905 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1906 {
1907   \int_zero:N \l__graphics_page_int
1908   \tl_clear:N \l__graphics_pagebox_tl
1909   \__graphics_extract_bb:n {#1}
1910 }
1911 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1912 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1913 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
1914 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1915 {
1916   \tl_clear:N \l__graphics_decodearray_str
1917   \bool_set_false:N \l__graphics_interpolate_bool
1918   \__graphics_extract_bb:n {#1}
1919 }
1920 </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.

```
1921 \int_new:N \g__graphics_track_int
```

(End definition for \g__graphics_track_int.)

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

```

\graphics_backend_include_eps:n
\graphics_backend_include_ps:n
\graphics_backend_include_jpg:n
\graphics_backend_include_jpseg:n
\graphics_backend_include_pdf:n
\graphics_backend_include_png:n
\graphics_backend_include_bmp:n
\graphics_backend_include_auxi:nn
\graphics_backend_include_auxii:nnn
\graphics_backend_include_auxiii:nnnn
1922 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1923 {
1924   \__kernel_backend_literal:x
1925   {
1926     PSfile = #1 \c_space_tl
1927     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1928     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1929     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1930     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1931   }
1932 }
1933 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1934 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1935 { \__graphics_backend_include_auxi:nn {#1} { image } }
1936 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1937 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
1938 \cs_new_eq:NN \__graphics_backend_include_bmp:n \__graphics_backend_include_jpg:n
1939 <*dvipdfmx>

```

```

1940 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1941 { \__graphics_backend_include_auxi:nn {#1} { epdf } }
1942 </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.

```

1943 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1944 {
1945   \__graphics_backend_include_auxii:xnn
1946   {
1947     \tl_if_empty:NF \l__graphics_pagebox_tl
1948     { : \l__graphics_pagebox_tl }
1949     \int_compare:nNnT \l__graphics_page_int > 1
1950     { :P \int_use:N \l__graphics_page_int }
1951     \tl_if_empty:NF \l__graphics_decodearray_str
1952     { :D \l__graphics_decodearray_str }
1953     \bool_if:NT \l__graphics_interpolate_bool
1954     { :I }
1955   }
1956   {#1} {#2}
1957 }
1958 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1959 {
1960   \int_if_exist:cTF { c__graphics_ #2#1 _int }
1961   {
1962     \__kernel_backend_literal:x
1963     { pdf:usexobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
1964   }
1965   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1966 }
1967 \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!

```

1968 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1969 {
1970   \int_gincr:N \g__graphics_track_int
1971   \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
1972   \__kernel_backend_literal:x
1973   {
1974     pdf:#3~
1975     @graphic \int_use:c { c__graphics_ #1#2 _int } ~
1976     \int_compare:nNnT \l__graphics_page_int > 1
1977     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
1978     \tl_if_empty:NF \l__graphics_pagebox_tl
1979     {
1980       pagebox ~ \l__graphics_pagebox_tl \c_space_tl
1981       bbox ~
1982       \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1983       \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1984       \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl

```

```

1985         \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
1986     }
1987     (#1)
1988     \bool_lazy_or:nnT
1989     { \l__graphics_interpolate_bool }
1990     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1991     {
1992         <<
1993         \tl_if_empty:NF \l__graphics_decodearray_str
1994         { /Decode~[ \l__graphics_decodearray_str ] }
1995         \bool_if:NT \l__graphics_interpolate_bool
1996         { /Interpolate~true> }
1997     }
1998     }
1999 }
2000 }

```

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

`__graphics_backend_get_pagecount:n`

```

2001 <*\dviptdmx>
2002 \__graphics_backend_loaded:n
2003 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }
2004 </dviptdmx>

```

(End definition for `__graphics_backend_get_pagecount:n`.)

```
2005 </dviptdmx | xetex>
```

5.4 X_YT_EX backend

```
2006 <*\xetex>
```

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

```

2007 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2008 {
2009     \int_zero:N \l__graphics_page_int
2010     \tl_clear:N \l__graphics_pagebox_tl
2011     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2012 }
2013 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2014 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
2015 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2016 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2017 {
2018     \tl_clear:N \l__graphics_decodearray_str
2019     \bool_set_false:N \l__graphics_interpolate_bool
2020     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2021 }
2022 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2023 {
2024     \int_compare:nNnTF \l__graphics_page_int > 1

```

```

2025     { \__graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2026     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2027   }
2028   \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2029   { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2030   \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2031   \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2032   {
2033     \tl_if_empty:NTF \l__graphics_pagebox_tl
2034     { \__graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2035     { \__graphics_backend_getbb_auxv:nNnn }
2036     {#1} #2 {#3} {#4}
2037   }
2038   \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2039   {
2040     \use:x
2041     {
2042       \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2043       {
2044         #5
2045         \tl_if_blank:nF {#1}
2046         { \c_space_tl \__graphics_backend_getbb_pagebox:w #1 }
2047       }
2048     }
2049   }
2050   \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2051   \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2052   {
2053     \__graphics_bb_restore:nF {#1#3}
2054     { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2055   }
2056   \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2057   {
2058     \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
2059     \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2060     \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2061     \__graphics_bb_save:n {#1#3}
2062   }
2063   \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

For PDF graphics, properly supporting the pagebox concept in X_YTeX is best done using the \tex_XeTeXpdfnfile: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.

```

2064   \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2065   {
2066     \tex_XeTeXpdfnfile:D #1 ~
2067     \int_compare:nNnT \l__graphics_page_int > 0
2068     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2069     \exp_after:wN \__graphics_backend_getbb_pagebox:w \l__graphics_pagebox_tl
2070   }

```

(End definition for __graphics_backend_include_pdf:n.)

_graphics_backend_get_pagecount:n

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

```
2071 \cs_new_protected:Npn \_graphics_backend_get_pagecount:n #1
2072 {
2073   \int_const:cn { c__graphics_ #1 _pages_int }
2074   {
2075     \int_max:nn
2076     { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2077     { 1 }
2078   }
2079 }
```

(End definition for _graphics_backend_get_pagecount:n.)

```
2080 </xetex>
```

5.5 dvisvgm backend

```
2081 <*dvisvgm>
```

\l_graphics_search_ext_seq

```
2082 \_graphics_backend_loaded:n
2083 {
2084   \seq_set_from_clist:Nn
2085   \l_graphics_search_ext_seq
2086   { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }
2087 }
```

(End definition for \l_graphics_search_ext_seq. This variable is documented on page ??.)

_graphics_backend_getbb_svg:n
_graphics_backend_getbb_svg_auxi:nNn
_graphics_backend_getbb_svg_auxii:w
_graphics_backend_getbb_svg_auxiii:Nw
_graphics_backend_getbb_svg_auxiv:Nw
_graphics_backend_getbb_svg_auxv:Nw
_graphics_backend_getbb_svg_auxvi:Nn
_graphics_backend_getbb_svg_auxvii:w

This is relatively similar to reading bounding boxes for .eps files. Life is though made more tricky as we cannot pick a single line for the data. So we have to loop until we collect up both height and width. To do that, we can use a marker value. We also have to allow for the default units of the lengths: they are big points and may be omitted.

```
2088 \cs_new_protected:Npn \_graphics_backend_getbb_svg:n #1
2089 {
2090   \_graphics_bb_restore:nF {#1}
2091   {
2092     \ior_open:Nn \l__graphics_internal_ior {#1}
2093     \ior_if_eof:NTF \l__graphics_internal_ior
2094     { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2095     {
2096       \dim_zero:N \l__graphics_llx_dim
2097       \dim_zero:N \l__graphics_lly_dim
2098       \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2099       \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2100       \ior_str_map_inline:Nn \l__graphics_internal_ior
2101       {
2102         \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2103         {
2104           \_graphics_backend_getbb_svg_auxi:nNn
2105           { width } \l__graphics_urx_dim {#1}
2106         }
2107         \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2108         {
2109           \_graphics_backend_getbb_svg_auxi:nNn
```

```

2110         { height } \l__graphics_ury_dim {##1}
2111     }
2112     \bool_lazy_and:nnF
2113     { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2114     { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2115     { \ior_map_break: }
2116 }
2117 \__graphics_bb_save:n {#1}
2118 }
2119 \ior_close:N \l__graphics_internal_ior
2120 }
2121 }
2122 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2123 {
2124     \use:x
2125     {
2126         \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxii:w
2127             ###1 \tl_to_str:n {#1} = ###2 \tl_to_str:n {#1} = ###3
2128         \s__graphics_stop
2129     }
2130     {
2131         \tl_if_blank:nF {##2}
2132         {
2133             \peek_remove_spaces:n
2134             {
2135                 \peek_meaning:NTF ' % '
2136                 { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2137                 {
2138                     \peek_meaning:NTF " % "
2139                     { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2140                     { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2141                 }
2142             }
2143             ##2 \s__graphics_stop
2144         }
2145     }
2146     \use:x
2147     {
2148         \__graphics_backend_getbb_svg_auxii:w #3
2149         \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2150         \s__graphics_stop
2151     }
2152 }
2153 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2154 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop
2155 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2156 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2157 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2158 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2159 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2160 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2161 {
2162     \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2163     \l__graphics_internal_dim #2 bp \scan_stop:

```



```

2164 \dim_set_eq:NN #1 \l__graphics_internal_dim
2165 }
2166 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvii:w #1 \scan_stop: { }

```

(End definition for __graphics_backend_getbb_svg:n and others.)

_graphics_backend_getbb_eps:n Simply use the generic function.

```

\_graphics_backend_getbb_ps:n
2167 \__graphics_backend_loaded:n
2168 {
2169 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2170 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
2171 }

```

(End definition for __graphics_backend_getbb_eps:n and __graphics_backend_getbb_ps:n.)

_graphics_backend_getbb_png:n These can be included by extracting the bounding box data.

```

\_graphics_backend_getbb_jpg:n
\_graphics_backend_getbb_jpeg:n
2172 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2173 {
2174 \int_zero:N \l__graphics_page_int
2175 \tl_clear:N \l__graphics_pagebox_tl
2176 \__graphics_extract_bb:n {#1}
2177 }
2178 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2179 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n

```

(End definition for __graphics_backend_getbb_png:n, __graphics_backend_getbb_jpg:n, and __graphics_backend_getbb_jpeg:n.)

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

```

2180 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2181 {
2182 \tl_clear:N \l__graphics_decodearray_str
2183 \bool_set_false:N \l__graphics_interpolate_bool
2184 \__graphics_extract_bb:n {#1}
2185 }

```

(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_ps:n
\_graphics_backend_include_pdf:n
\_graphics_backend_include:nn
2186 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2187 { \__graphics_backend_include:nn { PSfile } {#1} }
2188 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
2189 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2190 { \__graphics_backend_include:nn { pdffile } {#1} }
2191 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
2192 {
2193 \__kernel_backend_literal:x
2194 {
2195 #1 = #2 \c_space_tl
2196 llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2197 lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2198 urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2199 ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2200 }
2201 }

```

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

```

\__graphics_backend_include_svg:n
\__graphics_backend_include_png:n
\__graphics_backend_include_jpg:n
\__graphics_backend_include_jpeg:n
\__graphics_backend_include_dequote:w

2202 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2203 {
2204   \box_move_up:nn { \l__graphics_ury_dim }
2205   {
2206     \hbox:n
2207     {
2208       \__kernel_backend_literal:x
2209       {
2210         dvisvgm:img~
2211         \dim_to_decimal:n { \l__graphics_urx_dim } ~
2212         \dim_to_decimal:n { \l__graphics_ury_dim } ~
2213         \__graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2214       }
2215     }
2216   }
2217 }
2218 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_svg:n
2219 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_svg:n
2220 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_svg:n
2221 \cs_new:Npn \__graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2222 {#2}

```

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

```

\__graphics_backend_get_pagecount:n

2223 \__graphics_backend_loaded:n
2224 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }

(End definition for \__graphics_backend_get_pagecount:n.)

2225 </dvisvgm>
2226 </package>

```

6 l3backend-pdf implementation

```

2227 <*package>
2228 <@@=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
```

```
2229 \box_new:N \l__pdf_internal_box
```

(End definition for \l__pdf_internal_box.)

6.2 dvips backend

```
2230 <*dvips>
```

```
\__pdf_backend_pdfmark:n
```

```
\__pdf_backend_pdfmark:x
```

Used often enough it should be a separate function.

```
2231 \cs_new_protected:Npn \__pdf_backend_pdfmark:n #1
2232 { \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2233 \cs_generate_variant:Nn \__pdf_backend_pdfmark:n { x }
```

(End definition for __pdf_backend_pdfmark:n.)

6.2.1 Catalogue entries

```
\_pdf_backend_catalog_gput:nn
```

```
\__pdf_backend_info_gput:nn
```

```
2234 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2235 { \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2236 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2237 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }
```

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

6.2.2 Objects

```
\g__pdf_backend_object_int
```

For tracking objects.

```
2238 \int_new:N \g__pdf_backend_object_int
```

(End definition for \g__pdf_backend_object_int.)

```
\__pdf_backend_object_new:n
```

```
\__pdf_backend_object_ref:n
```

```
2239 \cs_new_protected:Npn \__pdf_backend_object_new:n #1
2240 {
2241   \int_gincr:N \g__pdf_backend_object_int
2242   \int_const:cn
2243   { c__pdf_object_ \tl_to_str:n {#1} _int }
2244   { \g__pdf_backend_object_int }
2245 }
2246 \cs_new:Npn \__pdf_backend_object_ref:n #1
2247 { { pdf.obj \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } } }
```

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

```
\_pdf_backend_object_write:nnn
```

```
\_pdf_backend_object_write:nnx
```

```
\_pdf_backend_object_write_aux:nnn
```

```
\_pdf_backend_object_write_array:nn
```

```
\_pdf_backend_object_write_dict:nn
```

```
\_pdf_backend_object_write_fstream:nn
```

```
\_pdf_backend_object_write_stream:nn
```

```
\_pdf_backend_object_write_stream:nnn
```

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

```
2248 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
2249 {
2250   \__pdf_backend_object_write_aux:nnn
2251   { \__pdf_backend_object_ref:n {#1} }
2252   {#2} {#3}
2253 }
2254 \cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nnx }
```

```

2255 \cs_new_protected:Npn \__pdf_backend_object_write_aux:nnn #1#2#3
2256 {
2257   \__pdf_backend_pdfmark:x
2258   {
2259     /objdef ~ #1
2260     /type
2261     \str_case:nn {#2}
2262     {
2263       { array } { /array }
2264       { dict } { /dict }
2265       { fstream } { /stream }
2266       { stream } { /stream }
2267     }
2268     /OBJ
2269   }
2270   \use:c { \__pdf_backend_object_write_ #2 :nn } {#1} {#3}
2271 }
2272 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2273 {
2274   \__pdf_backend_pdfmark:x
2275   { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2276 }
2277 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2278 {
2279   \__pdf_backend_pdfmark:x
2280   { #1 << \exp_not:n {#2} >> /PUT }
2281 }
2282 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2283 {
2284   \exp_args:Nx
2285   \__pdf_backend_object_write_fstream:nnn {#1} #2
2286 }
2287 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2288 {
2289   \__kernel_backend_postscript:n
2290   {
2291     SDict ~ begin ~
2292     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2293     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2294     end
2295   }
2296 }
2297 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2298 {
2299   \exp_args:Nx
2300   \__pdf_backend_object_write_stream:nnn {#1} #2
2301 }
2302 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2303 {
2304   \__kernel_backend_postscript:n
2305   {
2306     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2307     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2308   }

```

2309 }

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

`_pdf_backend_object_now:nn` No anonymous objects, so things are done manually.

`_pdf_backend_object_now:nx`

```

2310 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2311 {
2312   \int_gincr:N \g__pdf_backend_object_int
2313   \_pdf_backend_object_write_aux:nnn
2314   { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2315   {#1} {#2}
2316 }
2317 \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.

```

2318 \cs_new:Npn \_pdf_backend_object_last:
2319 { { 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.

```

2320 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2321 { { 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.

```

2322 \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.

```

2323 \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.

```

2324 \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.

```
2325 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2326 {
2327   \exp_args:Nf \_pdf_backend_annotation_aux:nnnn
2328   { \dim_eval:n {#1} } {#2} {#3} {#4}
2329 }
2330 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2331 {
2332   \box_move_down:nn {#3}
2333   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2334   \box_move_up:nn {#2}
2335   {
2336     \hbox:n
2337     {
2338       \_kernel_kern:n {#1}
2339       \_kernel_backend_postscript:n { pdf.save.ur }
2340       \_kernel_kern:n { -#1 }
2341     }
2342   }
2343   \int_gincr:N \g__pdf_backend_object_int
2344   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2345   \_pdf_backend_pdfmark:x
2346   {
2347     /objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2348     pdf.rect
2349     #4 ~
2350     /ANN
2351   }
2352 }
```

(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.

```
2353 \cs_new:Npn \_pdf_backend_annotation_last:
2354 { { 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.

```
2355 \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.

```
2356 \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.

```
2357 \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.

2358 `\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.

2359 `\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.

2360 `\tl_new:N \l__pdf_breaklink_pdfmark_tl`

2361 `\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.

2362 `\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.

2363 `\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`

2364 `\cs_new_protected:Npn __pdf_backend_link_begin_goto:nnw #1#2`

2365 `{`

2366 `__pdf_backend_link_begin:nw`

2367 `{ #1 /Subtype /Link /Action << /S /GoTo /D (#2) >> }`

2368 `}`

2369 `\cs_new_protected:Npn __pdf_backend_link_begin_user:nnw #1#2`

2370 `{ __pdf_backend_link_begin:nw {#1#2} }`

2371 `\cs_new_protected:Npn __pdf_backend_link_begin:nw #1`

2372 `{`

```

2373 \bool_if:NF \g__pdf_backend_link_bool
2374 { \__pdf_backend_link_begin_aux:nw {#1} }
2375 }

```

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.

```

2376 \cs_new_protected:Npn \__pdf_backend_link_begin_aux:nw #1
2377 {
2378   \bool_gset_true:N \g__pdf_backend_link_bool
2379   \__kernel_backend_postscript:n
2380   { /pdf.link.dict ( #1 ) def }
2381   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2382   \__pdf_backend_link_sf_save:
2383   \mode_if_math:TF
2384     { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2385     { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2386   \hbox_set:Nw \l__pdf_backend_content_box
2387   \__pdf_backend_link_sf_restore:
2388   \bool_if:NT \g__pdf_backend_link_math_bool
2389     { \c_math_toggle_token }
2390 }
2391 \cs_new_protected:Npn \__pdf_backend_link_end:
2392 {
2393   \bool_if:NT \g__pdf_backend_link_bool
2394     { \__pdf_backend_link_end_aux: }
2395 }
2396 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2397 {
2398   \bool_if:NT \g__pdf_backend_link_math_bool
2399     { \c_math_toggle_token }
2400   \__pdf_backend_link_sf_save:
2401   \hbox_set_end:
2402   \__pdf_backend_link_minima:
2403   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2404   \exp_args:Nx \__pdf_backend_link_outerbox:n
2405   {
2406     \int_if_odd:nTF { \value { page } }
2407       { \oddsidemargin }
2408       { \evensidemargin }
2409   }
2410   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2411     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2412   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2413   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2414   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2415   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2416     {
2417       \hbox:n
2418         { \__kernel_backend_postscript:n { pdf.save.linkur } }
2419     }
2420   \int_gincr:N \g__pdf_backend_object_int
2421   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2422   \__kernel_backend_postscript:x
2423   {

```



```

2424     mark
2425     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2426     \g__pdf_backend_link_dict_tl \c_space_tl
2427     pdf.rect
2428     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2429   }
2430   \__pdf_backend_link_sf_restore:
2431   \bool_gset_false:N \g__pdf_backend_link_bool
2432 }
2433 \cs_new_protected:Npn \__pdf_backend_link_minima:
2434 {
2435   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2436   \__kernel_backend_postscript:x
2437   {
2438     /pdf.linkdp.pad ~
2439     \dim_to_decimal:n
2440     {
2441       \dim_max:nn
2442       {
2443         \box_dp:N \l__pdf_backend_model_box
2444         - \box_dp:N \l__pdf_backend_content_box
2445       }
2446       { Opt }
2447     } ~
2448     pdf.pt.dvi ~ def
2449   /pdf.linkht.pad ~
2450   \dim_to_decimal:n
2451   {
2452     \dim_max:nn
2453     {
2454       \box_ht:N \l__pdf_backend_model_box
2455       - \box_ht:N \l__pdf_backend_content_box
2456     }
2457     { Opt }
2458   } ~
2459   pdf.pt.dvi ~ def
2460 }
2461 }
2462 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2463 {
2464   \__kernel_backend_postscript:x
2465   {
2466     /pdf.outerbox
2467     [
2468       \dim_to_decimal:n {#1} ~
2469       \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2470       \dim_to_decimal:n { #1 + \textwidth } ~
2471       \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2472     ]
2473     [ exch { pdf.pt.dvi } forall ] def
2474   /pdf.baselineskip ~
2475   \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2476   { pdf.pt.dvi ~ def }
2477   { pop ~ pop }

```

```

2478         ifelse
2479     }
2480 }
2481 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2482 {
2483     \int_gset:Nn \g__pdf_backend_link_sf_int
2484     {
2485         \mode_if_horizontal:TF
2486         { \tex_spacefactor:D }
2487         { 0 }
2488     }
2489 }
2490 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2491 {
2492     \mode_if_horizontal:T
2493     {
2494         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2495         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2496     }
2497 }

```

(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.

```

2498 \use_none:n
2499 {
2500     \cs_if_exist:NT \@makecol@hook
2501     {
2502         \tl_put_right:Nn \@makecol@hook
2503         {
2504             \box_if_empty:NF \@cclv
2505             {
2506                 \vbox_set:Nn \@cclv
2507                 {
2508                     \__kernel_backend_postscript:n
2509                     {
2510                         pdf.globaldict /pdf.brokenlink.rect ~ known
2511                         { pdf.bordertracking.continue }
2512                         if
2513                     }
2514                     \vbox_unpack_drop:N \@cclv
2515                     \__kernel_backend_postscript:n
2516                     { pdf.bordertracking.endpage }
2517                 }
2518             }
2519         }
2520         \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2521         \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2522         \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2523     }
2524 }

```

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

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

```
2525 \cs_new:Npn \_pdf_backend_link_last:
2526 { { 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.

```
2527 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2528 {
2529   \__kernel_backend_postscript:x
2530   {
2531     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2532   }
2533 }
```

(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. `fitr` without rule spec doesn't work, so it falls back to `/Fit` here.

```
2534 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2535 {
2536   \__kernel_backend_postscript:n { pdf.dest.anchor }
2537   \_pdf_backend_pdfmark:x
2538   {
2539     /View
2540     [
2541       \str_case:nnF {#2}
2542       {
2543         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2544         { fit } { /Fit }
2545         { fitb } { /FitB }
2546         { fitbh } { /FitBH ~ pdf.dest.y }
2547         { fitbv } { /FitBV ~ pdf.dest.x }
2548         { fith } { /FitH ~ pdf.dest.y }
2549         { fitv } { /FitV ~ pdf.dest.x }
2550         { fitr } { /Fit }
2551       }
2552       {
2553         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2554       }
2555     ]
2556     /Dest ( \exp_not:n {#1} ) cvn
2557     /DEST
2558   }
2559 }
2560 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2561 {
2562   \exp_args:Ne \_pdf_backend_destination_aux:nnnn
2563   { \dim_eval:n {#2} } {#1} {#3} {#4}
2564 }
```

```

2565 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2566 {
2567   \vbox_to_zero:n
2568   {
2569     \__kernel_kern:n {#4}
2570     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2571     \tex_vss:D
2572   }
2573   \__kernel_kern:n {#1}
2574   \vbox_to_zero:n
2575   {
2576     \__kernel_kern:n { -#3 }
2577     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2578     \tex_vss:D
2579   }
2580   \__kernel_kern:n { -#1 }
2581   \__pdf_backend_pdfmark:n
2582   {
2583     /View
2584     [
2585       /FitR ~
2586       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2587       pdf.urx ~ pdf.ury ~ pdf.dest2device
2588     ]
2589     /Dest ( #2 ) cvn
2590     /DEST
2591   }
2592 }

```

(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
2593 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2594 {
2595   \int_compare:nNnT {#1} = 0
2596   {
2597     \__kernel_backend_literal_postscript:n
2598     {
2599       /setdistillerparams ~ where
2600       { pop << /CompressPages ~ false >> setdistillerparams }
2601       if
2602     }
2603   }
2604 }
2605 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2606 {
2607   \bool_if:nF {#1}
2608   {
2609     \__kernel_backend_literal_postscript:n
2610     {
2611       /setdistillerparams ~ where

```

```

2612         { pop << /CompressStreams ~ false >> setdistillerparams }
2613     if
2614     }
2615 }
2616 }

```

(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`

```

2617 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2618 {
2619     \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2620 }
2621 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2622 {
2623     \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2624 }

```

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

`_pdf_backend_version_major:` Data not available!

```

2625 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2626 \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:
2627 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2628 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2629 \cs_new_protected:Npn \_pdf_backend_emc:
2630 { \_pdf_backend_pdfmark:n { /EMC } }

```

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

2631 `</dvips>`

6.3 LuaTeX and pdfTeX backend

2632 `<*luatex | pdftex>`

6.3.1 Annotations

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

```

2633 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2634 {
2635     <*luatex>
2636     \tex_pdfextension:D annot ~
2637     </luatex>
2638     <*pdftex>
2639     \tex_pdfannot:D
2640     </pdftex>
2641     width ~ \dim_eval:n {#1} ~
2642     height ~ \dim_eval:n {#2} ~
2643     depth ~ \dim_eval:n {#3} ~

```

```

2644     {#4}
2645 }

```

(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.

```

2646 \cs_new:Npx \_pdf_backend_annotation_last:
2647 {
2648     \exp_not:N \int_value:w
2649 <*luatex>
2650     \exp_not:N \tex_pdffeedback:D lastannot ~
2651 </luatex>
2652 <*pdftex>
2653     \exp_not:N \tex_pdflastannot:D
2654 </pdftex>
2655     \c_space_tl 0 ~ R
2656 }

```

(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
    \_pdf_backend_link_begin:nnnw
\_pdf_backend_link_end:
2657 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2658 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2659 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2660 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2661 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2662 {
2663 <*luatex>
2664     \tex_pdfextension:D startlink ~
2665 </luatex>
2666 <*pdftex>
2667     \tex_pdfstartlink:D
2668 </pdftex>
2669     attr {#1}
2670     #2 {#3}
2671 }
2672 \cs_new_protected:Npn \_pdf_backend_link_end:
2673 {
2674 <*luatex>
2675     \tex_pdfextension:D endlink \scan_stop:
2676 </luatex>
2677 <*pdftex>
2678     \tex_pdfendlink:D
2679 </pdftex>
2680 }

```

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

`_pdf_backend_link_last:` Formatted for direct use.

```

2681 \cs_new:Npx \_pdf_backend_link_last:
2682 {
2683     \exp_not:N \int_value:w
2684 <*luatex>

```

```

2685 \exp_not:N \tex_pdffeedback:D lastlink ~
2686 </luatex>
2687 <*pdfTeX>
2688 \exp_not:N \tex_pdflastlink:D
2689 </pdfTeX>
2690 \c_space_tl 0 ~ R
2691 }

```

(End definition for _pdf_backend_link_last:.)

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

```

2692 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2693 {
2694 <luatex>
2695 \tex_pdfvariable:D linkmargin
2696 </luatex>
2697 <*pdfTeX>
2698 \tex_pdflinkmargin:D
2699 </pdfTeX>
2700 \dim_eval:n {#1} \scan_stop:
2701 }

```

(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

```

2702 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2703 {
2704 <luatex>
2705 \tex_pdfextension:D dest ~
2706 </luatex>
2707 <*pdfTeX>
2708 \tex_pdfdest:D
2709 </pdfTeX>
2710 name {#1}
2711 \str_case:nnF {#2}
2712 {
2713 { xyz } { xyz }
2714 { fit } { fit }
2715 { fitb } { fitb }
2716 { fitbh } { fitbh }
2717 { fitbv } { fitbv }
2718 { fith } { fith }
2719 { fitv } { fitv }
2720 { fitr } { fitr }
2721 }
2722 { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2723 \scan_stop:
2724 }
2725 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2726 {
2727 <luatex>
2728 \tex_pdfextension:D dest ~

```

```

2729 </luatex>
2730 <*pdftex>
2731     \tex_pdfdest:D
2732 </pdftex>
2733     name {#1}
2734     fitr ~
2735         width \dim_eval:n {#2} ~
2736         height \dim_eval:n {#3} ~
2737         depth \dim_eval:n {#4} \scan_stop:
2738 }

```

(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
2739 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2740 {
2741 <*luatex>
2742     \tex_pdfextension:D catalog
2743 </luatex>
2744 <*pdftex>
2745     \tex_pdfcatalog:D
2746 </pdftex>
2747     { / #1 ~ #2 }
2748 }
2749 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2750 {
2751 <*luatex>
2752     \tex_pdfextension:D info
2753 </luatex>
2754 <*pdftex>
2755     \tex_pdfinfo:D
2756 </pdftex>
2757     { / #1 ~ #2 }
2758 }

```

(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.
2759 \prop_new:N \g__pdf_backend_object_prop

```

(End definition for \g__pdf_backend_object_prop.)

```

\_pdf_backend_object_new:n Declaring objects means reserving at the PDF level plus starting tracking.
\_pdf_backend_object_ref:n
2760 \cs_new_protected:Npn \_pdf_backend_object_new:n #1
2761 {
2762 <*luatex>
2763     \tex_pdfextension:D obj ~
2764 </luatex>
2765 <*pdftex>
2766     \tex_pdfobj:D

```



```

2767 </pdfTex>
2768     reserveobjnum ~
2769     \int_const:cn
2770     { c__pdf_object_ \tl_to_str:n {#1} _int }
2771 <*luatex>
2772     { \tex_pdffeedback:D lastobj }
2773 </luatex>
2774 <*pdfTex>
2775     { \tex_pdflastobj:D }
2776 </pdfTex>
2777 }
2778 \cs_new:Npn \__pdf_backend_object_ref:n #1
2779 { \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

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

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

```

\__pdf_backend_object_write:nnx 2780 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
\__pdf_backend_object_write:nn 2781 {
  \__pdf_exp_not_i:nn 2782 <*luatex>
  \__pdf_exp_not_ii:nn 2783     \tex_immediate:D \tex_pdfextension:D obj ~
2784 </luatex>
2785 <*pdfTex>
2786     \tex_immediate:D \tex_pdfobj:D
2787 </pdfTex>
2788     useobjnum ~
2789     \int_use:c
2790     { c__pdf_object_ \tl_to_str:n {#1} _int }
2791     \__pdf_backend_object_write:nn {#2} {#3}
2792 }
2793 \cs_new:Npn \__pdf_backend_object_write:nn #1#2
2794 {
2795     \str_case:nn {#1}
2796     {
2797         { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2798         { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2799         { fstream }
2800         {
2801             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2802             file ~ { \__pdf_exp_not_ii:nn #2 }
2803         }
2804         { stream }
2805         {
2806             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2807             { \__pdf_exp_not_ii:nn #2 }
2808         }
2809     }
2810 }
2811 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nnx }
2812 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2813 \cs_new:Npn \__pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

```

(End definition for __pdf_backend_object_write:nnn and others.)

`_pdf_backend_object_now:nn` Much like writing, but direct creation.

`_pdf_backend_object_now:nx`

```

2814 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2815 {
2816   \*luatex
2817   \tex_immediate:D \tex_pdfextension:D obj ~
2818   \*pdftex
2819   \tex_immediate:D \tex_pdfobj:D
2820   \*pdftex
2821   \_pdf_backend_object_write:nn {#1} {#2}
2822 }
2823 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }
2824
(End definition for \_pdf_backend_object_now:nn.)

```

`_pdf_backend_object_last:` Much like annotation.

```

2825 \cs_new:Npx \_pdf_backend_object_last:
2826 {
2827   \exp_not:N \int_value:w
2828   \*luatex
2829   \exp_not:N \tex_pdffeedback:D lastobj ~
2830   \*pdftex
2831   \exp_not:N \tex_pdflastobj:D
2832   \*pdftex
2833   \c_space_tl 0 ~ R
2834 }
2835
(End definition for \_pdf_backend_object_last:.)

```

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

```

2836 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2837 {
2838   \exp_not:N \int_value:w
2839   \*luatex
2840   \exp_not:N \tex_pdffeedback:D pageref
2841   \*pdftex
2842   \exp_not:N \tex_pdfpageref:D
2843   \*pdftex
2844   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2845 }
2846
(End definition for \_pdf_backend_pageobject_ref:n.)

```

6.3.4 Structure

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

`_pdf_backend_compress_objects:n`

`_pdf_backend_objcompresslevel:n`

```

2847 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2848 {
2849   \tex_global:D
2850   \*luatex
2851   \tex_pdfvariable:D compresslevel
2852   \*pdftex

```

```

2853 <*pdftex>
2854     \tex_pdfcompresslevel:D
2855 </pdftex>
2856     \int_value:w \int_eval:n {#1} \scan_stop:
2857 }
2858 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2859 {
2860     \bool_if:nTF {#1}
2861     { \__pdf_backend_objcompresslevel:n { 2 } }
2862     { \__pdf_backend_objcompresslevel:n { 0 } }
2863 }
2864 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2865 {
2866     \tex_global:D
2867 <*luatex>
2868     \tex_pdfvariable:D objcompresslevel
2869 </luatex>
2870 <*pdftex>
2871     \tex_pdfobjcompresslevel:D
2872 </pdftex>
2873     #1 \scan_stop:
2874 }

(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.

```

2875 \cs_new_protected:Npx \__pdf_backend_version_major_gset:n #1
2876 {
2877 <*luatex>
2878     \int_compare:nNnT \tex_luatexversion:D > { 106 }
2879     {
2880         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2881         \exp_not:N \int_eval:n {#1} \scan_stop:
2882     }
2883 </luatex>
2884 <*pdftex>
2885     \cs_if_exist:NT \tex_pdfmajorversion:D
2886     {
2887         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2888         \exp_not:N \int_eval:n {#1} \scan_stop:
2889     }
2890 </pdftex>
2891 }
2892 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2893 {
2894     \tex_global:D
2895 <*luatex>
2896     \tex_pdfvariable:D minorversion
2897 </luatex>
2898 <*pdftex>
2899     \tex_pdfminorversion:D
2900 </pdftex>
2901     \int_eval:n {#1} \scan_stop:
2902 }

```

(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:
2903 \cs_new:Npx \_pdf_backend_version_major:
2904 {
2905   <*luatex>
2906     \int_compare:nNnTF \tex luatexversion:D > { 106 }
2907       { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2908       { 1 }
2909   </luatex>
2910   <*pdftex>
2911     \cs_if_exist:NTF \tex_pdfmajorversion:D
2912       { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2913       { 1 }
2914   </pdftex>
2915 }
2916 \cs_new:Npn \_pdf_backend_version_minor:
2917 {
2918   \tex_the:D
2919   <*luatex>
2920     \tex_pdfvariable:D minorversion
2921   </luatex>
2922   <*pdftex>
2923     \tex_pdfminorversion:D
2924   </pdftex>
2925 }

```

(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:
2926 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2927 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2928 \cs_new_protected:Npn \_pdf_backend_emc:
2929 { \__kernel_backend_literal_page:n { EMC } }

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

2930 </luatex | pdftex>

```

6.4 dvipdfmx backend

```

2931 <*dvipdfmx | xetex>

```

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

```

\__pdf_backend:x
2932 \cs_new_protected:Npx \_pdf_backend:n #1
2933 { \__kernel_backend_literal:n { pdf: #1 } }
2934 \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
2935 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2936 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2937 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2938 { \__pdf_backend:n { docinfo << /#1 ~ #2 >> } }

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

```

6.4.2 Objects

For tracking objects to allow finalisation.

```

\g__pdf_backend_object_int
\g__pdf_backend_object_prop
2939 \int_new:N \g__pdf_backend_object_int
2940 \prop_new:N \g__pdf_backend_object_prop

(End definition for \g__pdf_backend_object_int and \g__pdf_backend_object_prop.)

```

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

```

\__pdf_backend_object_new:n
\__pdf_backend_object_ref:n
2941 \cs_new_protected:Npn \__pdf_backend_object_new:n #1
2942 {
2943   \int_gincr:N \g__pdf_backend_object_int
2944   \int_const:cn
2945     { c__pdf_object_ \tl_to_str:n {#1} _int }
2946     { \g__pdf_backend_object_int }
2947 }
2948 \cs_new:Npn \__pdf_backend_object_ref:n #1
2949 { @pdf.obj \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } }

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

```

This is where we choose the actual type.

```

\__pdf_backend_object_write:nnn
\__pdf_backend_object_write:nnx
2950 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2951 {
2952   \use:c { __pdf_backend_object_write_ #2 :nn }
2953   { \__pdf_backend_object_ref:n {#1} } {#3}
2954 }
2955 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nnx }
2956 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2957 {
2958   \__pdf_backend:x
2959     { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2960 }
2961 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2962 {
2963   \__pdf_backend:x
2964     { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2965 }
2966 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2967 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2968 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2969 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2970 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2971 {
2972   \__pdf_backend:x

```

```

2973     {
2974         #1 stream ~ #2 ~
2975         ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2976     }
2977 }

```

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

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

```

\__pdf_backend_object_now:nx
2978 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2979 {
2980     \int_gincr:N \g__pdf_backend_object_int
2981     \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2982     { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2983     {#2}
2984 }
2985 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for `__pdf_backend_object_now:nn`.)

`__pdf_backend_object_last:`

```

2986 \cs_new:Npn \__pdf_backend_object_last:
2987 { @pdf.obj \int_use:N \g__pdf_backend_object_int }

```

(End definition for `__pdf_backend_object_last:.`)

`_pdf_backend_pageobject_ref:n` Page references are easy in dvipdfmx/X_YTeX.

```

2988 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2989 { @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.

```

2990 \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.

```

2991 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2992 {
2993     \int_gincr:N \g__pdf_backend_object_int
2994     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2995     \_pdf_backend:x
2996     {
2997         ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
2998         width ~ \dim_eval:n {#1} ~
2999         height ~ \dim_eval:n {#2} ~
3000         depth ~ \dim_eval:n {#3} ~
3001         << /Type /Annot #4 >>
3002     }
3003 }

```

(End definition for `_pdf_backend_annotation:nnnn`.)

_pdf_backend_annotation_last:

```
3004 \cs_new:Npn \_pdf_backend_annotation_last:
3005 { @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.

```
3006 \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:

```
3007 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
3008 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
3009 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
3010 { \_pdf_backend_link_begin:n {#1#2} }
3011 \cs_new_protected:Npx \_pdf_backend_link_begin:n #1
3012 {
3013   \exp_not:N \int_gincr:N \exp_not:N \g__pdf_backend_link_int
3014   \_pdf_backend:x
3015   {
3016     bann ~
3017     @pdf.lnk
3018     \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
3019     \c_space_tl
3020     <<
3021     /Type /Annot
3022     #1
3023     >>
3024   }
3025 }
3026 \cs_new_protected:Npn \_pdf_backend_link_end:
3027 { \_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.

```
3028 \cs_new:Npn \_pdf_backend_link_last:
3029 { @pdf.lnk \int_use:N \g__pdf_backend_link_int }
```

(End definition for _pdf_backend_link_last:.)

_pdf_backend_link_margin:n Pass to dvipdfmx.

```
3030 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
3031 { \_kernel_backend_literal:x { dvipdfmx:config-g~ \dim_eval:n {#1} } }
```

(End definition for _pdf_backend_link_margin:n.)

_pdf_backend_destination:nn

_pdf_backend_destination:nnnn

_pdf_backend_destination_aux:nnnn

Here, we need to turn the zoom into a scale. The method for FitR is from Alexander Grahn: the idea is to avoid needing to do any calculations in 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.

```
3032 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
3033 {
3034   \_pdf_backend:x
```

```

3035 {
3036   dest ~ ( \exp_not:n {#1} )
3037   [
3038     @thispage
3039     \str_case:nnF {#2}
3040     {
3041       { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
3042       { fit } { /Fit }
3043       { fitb } { /FitB }
3044       { fitbh } { /FitBH }
3045       { fitbv } { /FitBV ~ @xpos }
3046       { fith } { /FitH ~ @ypos }
3047       { fitv } { /FitV ~ @xpos }
3048       { fitr } { /Fit }
3049     }
3050     { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
3051   ]
3052 }
3053 }
3054 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
3055 {
3056   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
3057   { \dim_eval:n {#2} } {#1} {#3} {#4}
3058 }
3059 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
3060 {
3061   \vbox_to_zero:n
3062   {
3063     \__kernel_kern:n {#4}
3064     \hbox:n
3065     {
3066       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
3067       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
3068     }
3069     \tex_vss:D
3070   }
3071   \__kernel_kern:n {#1}
3072   \vbox_to_zero:n
3073   {
3074     \__kernel_kern:n { -#3 }
3075     \hbox:n
3076     {
3077       \__pdf_backend:n
3078       {
3079         dest ~ (#2)
3080         [
3081           @thispage
3082           /FitR ~
3083           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
3084           @xpos ~ @ypos
3085         ]
3086       }
3087     }
3088     \tex_vss:D

```



```

3089     }
3090     \__kernel_kern:n { -#1 }
3091 }

```

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

6.4.4 Structure

_pdf_backend_compresslevel:n
_pdf_backend_compress_objects:n

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

```

3092 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
3093 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
3094 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
3095 {
3096   \bool_if:nF {#1}
3097   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
3098 }

```

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

_pdf_backend_version_major_gset:n
_pdf_backend_version_minor_gset:n

We start with the assumption that the default is active.

```

3099 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
3100 {
3101   \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
3102   \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
3103 }
3104 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
3105 {
3106   \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
3107   \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
3108 }

```

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

_pdf_backend_version_major:
_pdf_backend_version_minor:

We start with the assumption that the default is active.

```

3109 \cs_new:Npn \__pdf_backend_version_major: { 1 }
3110 \cs_new:Npn \__pdf_backend_version_minor: { 5 }

```

(End definition for __pdf_backend_version_major: and __pdf_backend_version_minor:.)

6.4.5 Marked content

__pdf_backend_bdc:nn
_pdf_backend_emc:

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

```

3111 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
3112 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
3113 \cs_new_protected:Npn \__pdf_backend_emc:
3114 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

3115 </dvipdfmx | xetex>

```

6.5 dvisvgm backend

3116 $\langle *dvisvgm \rangle$

6.5.1 Annotations

$\backslash_pdf_backend_annotation:nnnn$

3117 $\backslash cs_new_protected:Npn \backslash_pdf_backend_annotation:nnnn \#1\#2\#3\#4 \{ \}$

(End definition for $\backslash_pdf_backend_annotation:nnnn$.)

$\backslash_pdf_backend_annotation_last:$

3118 $\backslash cs_new:Npn \backslash_pdf_backend_annotation_last: \{ \}$

(End definition for $\backslash_pdf_backend_annotation_last:$.)

$\backslash_pdf_backend_link_begin_goto:nnw$

$\backslash_pdf_backend_link_begin_user:nnw$

$\backslash_pdf_backend_link_begin:nnnw$

$\backslash_pdf_backend_link_end:$

3119 $\backslash cs_new_protected:Npn \backslash_pdf_backend_link_begin_goto:nnw \#1\#2 \{ \}$

3120 $\backslash cs_new_protected:Npn \backslash_pdf_backend_link_begin_user:nnw \#1\#2 \{ \}$

3121 $\backslash cs_new_protected:Npn \backslash_pdf_backend_link_begin:nnnw \#1\#2\#3 \{ \}$

3122 $\backslash cs_new_protected:Npn \backslash_pdf_backend_link_end: \{ \}$

(End definition for $\backslash_pdf_backend_link_begin_goto:nnw$ and others.)

$\backslash_pdf_backend_link_last:$

3123 $\backslash cs_new:Npx \backslash_pdf_backend_link_last: \{ \}$

(End definition for $\backslash_pdf_backend_link_last:$.)

$\backslash_pdf_backend_link_margin:n$

A simple task: pass the data to the primitive.

3124 $\backslash cs_new_protected:Npn \backslash_pdf_backend_link_margin:n \#1 \{ \}$

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

$\backslash_pdf_backend_destination:nn$

$\backslash_pdf_backend_destination:nnnn$

3125 $\backslash cs_new_protected:Npn \backslash_pdf_backend_destination:nn \#1\#2 \{ \}$

3126 $\backslash cs_new_protected:Npn \backslash_pdf_backend_destination:nnnn \#1\#2\#3\#4 \{ \}$

(End definition for $\backslash_pdf_backend_destination:nn$ and $\backslash_pdf_backend_destination:nnnn$.)

6.5.2 Catalogue entries

$\backslash_pdf_backend_catalog_gput:nn$

No-op.

$\backslash_pdf_backend_info_gput:nn$

3127 $\backslash cs_new_protected:Npn \backslash_pdf_backend_catalog_gput:nn \#1\#2 \{ \}$

3128 $\backslash cs_new_protected:Npn \backslash_pdf_backend_info_gput:nn \#1\#2 \{ \}$

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

6.5.3 Objects

All no-ops here.

```

\__pdf_backend_object_new:n
\__pdf_backend_object_ref:n
    \__pdf_backend_object_write:nnn
    \__pdf_backend_object_write:nx
\__pdf_backend_object_now:nn
\__pdf_backend_object_now:nx
\__pdf_backend_object_last:
    \__pdf_backend_pageobject_ref:n
3129 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1 { }
3130 \cs_new:Npn \__pdf_backend_object_ref:n #1 { }
3131 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3 { }
3132 \cs_new_protected:Npn \__pdf_backend_object_write:nnx #1#2#3 { }
3133 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2 { }
3134 \cs_new_protected:Npn \__pdf_backend_object_now:nx #1#2 { }
3135 \cs_new:Npn \__pdf_backend_object_last: { }
3136 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1 { }

```

(End definition for __pdf_backend_object_new:n and others.)

6.5.4 Structure

These are all no-ops.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
3137 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
3138 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1 { }

```

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

Data not available!

```

\__pdf_backend_version_major_gset:n
\__pdf_backend_version_minor_gset:n
3139 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
3140 \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.)

Data not available!

```

\__pdf_backend_version_major:
\__pdf_backend_version_minor:
3141 \cs_new:Npn \__pdf_backend_version_major: { -1 }
3142 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

```

(End definition for __pdf_backend_version_major: and __pdf_backend_version_minor:.)

More no-ops.

```

\__pdf_backend_bdc:nn
\__pdf_backend_emc:
3143 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
3144 \cs_new_protected:Npn \__pdf_backend_emc: { }

```

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

3145 \langle /dvisvgm \rangle

6.6 PDF Page size (media box)

For setting the media box, the split between backends is somewhat different to other areas, thus we approach this separately. The code here assumes a recent L^AT_EX 2_ε: that is ensured at the level above.

3146 \langle *dvipdfmx | dvips \rangle

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

```

3147 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
3148 {
3149   \_kernel_backend_first_shipout:n
3150   {
3151     \_kernel_backend_literal:e
3152     {
3153       <*dvipdfmx>
3154         pdf:pagesize ~
3155         width ~ \dim_eval:n {#1} ~
3156         height ~ \dim_eval:n {#2}
3157       </dvipdfmx>
3158       <*dvips>
3159         papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
3160       </dvips>
3161     }
3162   }
3163 }

(End definition for \_pdf_backend_pagesize_gset:nn.)

3164 </dvipdfmx | dvips>
3165 <*luatex | pdftex | xetex>

```

`_pdf_backend_pagesize_gset:nn` Pass to the primitives.

```

3166 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
3167 {
3168   \dim_gset:Nn \tex_pagewidth:D {#1}
3169   \dim_gset:Nn \tex_pageheight:D {#2}
3170 }

(End definition for \_pdf_backend_pagesize_gset:nn.)

3171 </luatex | pdftex | xetex>
3172 <*dvisvgm>

```

`_pdf_backend_pagesize_gset:nn` A no-op.

```

3173 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2 { }

(End definition for \_pdf_backend_pagesize_gset:nn.)

3174 </dvisvgm>
3175 </package>

```

7 l3backend-opacity implementation

```

3176 <*package>
3177 <@@=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.

3178 $\langle *dvips \rangle$

$\backslash_opacity_backend_select:n$
 $\backslash_opacity_backend_select_aux:n$
 $\backslash_opacity_backend_fill:n$
 $\backslash_opacity_backend_stroke:n$
 $\backslash_opacity_backend:nnn$
 $\backslash_opacity_backend:xnn$

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.

```

3179 \cs_new_protected:Npn \__opacity_backend_select:n #1
3180 {
3181   \exp_args:Nx \__opacity_backend_select_aux:n
3182   { \fp_eval:n { min(max(0,#1),1) } }
3183 }
3184 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3185 {
3186   \__opacity_backend:nnn {#1} { fill } { ca }
3187   \__opacity_backend:nnn {#1} { stroke } { CA }
3188 }
3189 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3190 {
3191   \__opacity_backend:xnn
3192   { \fp_eval:n { min(max(0,#1),1) } }
3193   { fill }
3194   { ca }
3195 }
3196 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3197 {
3198   \__opacity_backend:xnn
3199   { \fp_eval:n { min(max(0,#1),1) } }
3200   { stroke }
3201   { CA }
3202 }
3203 \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3204 {
3205   \__kernel_backend_postscript:n
3206   {
3207     product ~ (Ghostscript) ~ search
3208     {
3209       pop ~ pop ~ pop ~
3210       #1 ~ .set #2 constantalpha
3211     }
3212     {
3213       pop ~
3214       mark ~
3215       /#3 ~ #1
3216       /SetTransparency ~
3217       pdfmark
3218     }
3219     ifelse
3220   }
3221 }
3222 \cs_generate_variant:Nn \__opacity_backend:nnn { x }

```

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

3223 $\langle /dvips \rangle$

3224 $\langle *dvipdfmx | luatex | pdftex | xetex \rangle$

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

```

3225 \bool_lazy_and:nnT
3226 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3227 { \pdfmanagement_if_active_p:}
3228 {
3229  $\langle *luatex | pdftex \rangle$ 
3230   \__kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3231   { page ~ direct } { /opacity 1 ~ gs }
3232  $\langle /luatex | pdftex \rangle$ 
3233   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3234   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3235 }

```

(End definition for $\backslash c_opacity_backend_stack_int$.)

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

```

\l_opacity_backend_stroke_tl
3236 \tl_new:N \l__opacity_backend_fill_tl
3237 \tl_new:N \l__opacity_backend_stroke_tl

```

(End definition for $\backslash l_opacity_backend_fill_tl$ and $\backslash l_opacity_backend_stroke_tl$.)

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

```

\__opacity_backend_select_aux:n
\__opacity_backend_reset:
3238 \cs_new_protected:Npn \__opacity_backend_select:n #1
3239 {
3240   \exp_args:Nx \__opacity_backend_select_aux:n
3241   { \fp_eval:n { min(max(0,#1),1) } }
3242 }
3243 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3244 {
3245   \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3246   \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3247   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3248   { opacity #1 }
3249   { << /ca ~ #1 /CA ~ #1 >> }
3250  $\langle *dvipdfmx | xetex \rangle$ 
3251   \__kernel_backend_literal_pdf:n
3252  $\langle /dvipdfmx | xetex \rangle$ 
3253  $\langle *luatex | pdftex \rangle$ 
3254   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3255  $\langle /luatex | pdftex \rangle$ 
3256   { /opacity #1 ~ gs }
3257   \group_insert_after:N \__opacity_backend_reset:
3258 }
3259 \bool_lazy_and:nnF
3260 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3261 { \pdfmanagement_if_active_p:}
3262 {
3263   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1 { }
3264 }
3265 \cs_new_protected:Npn \__opacity_backend_reset:
3266 {
3267  $\langle *dvipdfmx | xetex \rangle$ 

```

```

3268   \__kernel_backend_literal_pdf:n
3269   { /opacity1 ~ gs }
3270 </dviptdmx | xetex>
3271 <*luatex | pdftex>
3272   \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int
3273 </luatex | pdftex>
3274 }

```

(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
3275 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3276 {
3277   \__opacity_backend_fill_stroke:xx
3278   { \fp_eval:n { min(max(0,#1),1) } }
3279   \l__opacity_backend_stroke_tl
3280 }
3281 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3282 {
3283   \__opacity_backend_fill_stroke:xx
3284   \l__opacity_backend_fill_tl
3285   { \fp_eval:n { min(max(0,#1),1) } }
3286 }
3287 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3288 {
3289   \str_if_eq:nnTF {#1} {#2}
3290   { \__opacity_backend_select_aux:n {#1} }
3291   {
3292     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3293     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3294     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3295     { opacity.fill #1 }
3296     { << /ca ~ #1 >> }
3297     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3298     { opacity.stroke #1 }
3299     { << /CA ~ #2 >> }
3300 <*dviptdmx | xetex>
3301   \__kernel_backend_literal_pdf:n
3302 </dviptdmx | xetex>
3303 <*luatex | pdftex>
3304   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3305 </luatex | pdftex>
3306   { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3307   \group_insert_after:N \__opacity_backend_reset:
3308 }
3309 }
3310 \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.)

```

3311 </dviptdmx | luatex | pdftex | xetex>
3312 <*dvisvgm>

```

```

\__opacity_backend_select:n
\__opacity_backend_fill:n
\__opacity_backend_stroke:n
\__opacity_backend:nn
3313 \cs_new_protected:Npn \__opacity_backend_select:n #1
3314 { \__opacity_backend:nn {#1} { } }
3315 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3316 { \__opacity_backend:nn {#1} { fill- } }
3317 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3318 { \__opacity_backend:nn { {#1} } { stroke- } }
3319 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3320 { \__kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

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

3321 \</dvisvgm>
3322 \</package>

```

7.1 Font handling integration

In LuaTeX we want to use these functions also for transparent fonts to avoid interference between both uses of transparency.

```

3323 \*lua>

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

3324 local pdfmanagement_active do
3325   local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3326   local cmd = pdfmanagement_if_active_p.cmdname
3327   if cmd == 'undefined_cs' then
3328     pdfmanagement_active = false
3329   else
3330     token.put_next(pdfmanagement_if_active_p)
3331     pdfmanagement_active = token.scan_int() ~= 0
3332   end
3333 end

3334

3335 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3336   luaotfload.set_transparent_colorstack(token.create'c__opacity_backend_stack_int'.index)
3337

3338   local transparent_register = {
3339     token.create'pdfmanagement_add:nnn',
3340     token.new(0, 1),
3341     'Page/Resources/ExtGState',
3342     token.new(0, 2),
3343     token.new(0, 1),
3344     '',
3345     token.new(0, 2),
3346     token.new(0, 1),
3347     '<</ca ',
3348     '',
3349     '/CA ',
3350     '',
3351     '>>',
3352     token.new(0, 2),
3353   }
3354   luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)

```



```

3355     value = (octet * -1):match(value)
3356     if not value then
3357         tex.error'Invalid transparency value'
3358         return
3359     end
3360     value = value:sub(1, -2)
3361     local result = 'opacity' .. value
3362     tex.runtoks(function()
3363         transparent_register[6], transparent_register[10], transparent_register[12] = result,
3364         tex.sprint(-2, transparent_register)
3365     end)
3366     return '/' .. result .. ' gs'
3367 end, 'l3opacity')
3368 end
3369  $\langle$ /lua $\rangle$ 

```

8 l3backend-header implementation

```

3370  $\langle$ *dvips & header $\rangle$ 

```

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

```

3371 /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.

```

3372 TeXDict begin
3373 /TeXcolorseparation { setcolor } def
3374 end

```

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

pdf.globaldict A small global dictionary for backend use.

```

3375 true setglobal
3376 /pdf.globaldict 4 dict def
3377 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.

```

3378 /pdf.cvs { 65534 string cvs } def
3379 /pdf.dvi.pt { 72.27 mul Resolution div } def
3380 /pdf.pt.dvi { 72.27 div Resolution mul } def
3381 /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.

```

3382 /pdf.linkmargin { 1 pdf.pt.dvi } def
3383 /pdf.linkdp.pad { 0 } def
3384 /pdf.linkht.pad { 0 } def

```

(End definition for `pdf.linkmargin`, `pdf.linkdp.pad`, and `pdf.linkht.pad`. These functions are documented 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 3385 /pdf.rect
pdf.save.linkur 3386 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.llx         3387 /pdf.save.ll
pdf.lly         3388 {
pdf.urx         3389     currentpoint
pdf.ury         3390     /pdf.lly exch def
                3391     /pdf.llx exch def
                3392 }
                3393 def
pdf.save.ur     3394 /pdf.save.ur
                3395 {
                3396     currentpoint
                3397     /pdf.ury exch def
                3398     /pdf.urx exch def
                3399 }
                3400 def
pdf.save.linkll 3401 /pdf.save.linkll
                3402 {
                3403     currentpoint
                3404     pdf.linkmargin add
                3405     pdf.linkdp.pad add
                3406     /pdf.lly exch def
                3407     pdf.linkmargin sub
                3408     /pdf.llx exch def
                3409 }
                3410 def
pdf.save.linkur 3411 /pdf.save.linkur
                3412 {
                3413     currentpoint
                3414     pdf.linkmargin sub
                3415     pdf.linkht.pad sub
                3416     /pdf.ury exch def
                3417     pdf.linkmargin add
                3418     /pdf.urx exch def
                3419 }
                3420 def
```

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

`pdf.dest.anchor` For finding the anchor point of a destination link. We make the use case a separate
`pdf.dest.x` function as it comes up a lot, and as this makes it easier to adjust if we need additional
`pdf.dest.y` effects. We also need a more complex approach to convert a co-ordinate pair correctly
`pdf.dest.point` when defining a rectangle: this can otherwise be out when using a landscape page.
`pdf.dest2device` (Thanks to Alexander Grahn for the approach here.)

```
pdf.dev.x       3421 /pdf.dest.anchor
pdf.dev.y       3422 {
pdf.tmpa        3423     currentpoint exch
pdf.tmpb        3424     pdf.dvi.pt 72 add
pdf.tmpc
pdf.tmpd
```

```

3425     /pdf.dest.x exch def
3426     pdf.dvi.pt
3427     vsize 72 sub exch sub
3428     /pdf.dest.y exch def
3429   }
3430   def
3431 /pdf.dest.point
3432   { pdf.dest.x pdf.dest.y } def
3433 /pdf.dest2device
3434   {
3435     /pdf.dest.y exch def
3436     /pdf.dest.x exch def
3437     matrix currentmatrix
3438     matrix defaultmatrix
3439     matrix invertmatrix
3440     matrix concatmatrix
3441     cvx exec
3442     /pdf.dev.y exch def
3443     /pdf.dev.x exch def
3444     /pdf.tmpd exch def
3445     /pdf.tmpc exch def
3446     /pdf.tmpb exch def
3447     /pdf.tmpa exch def
3448     pdf.dest.x pdf.tmpa mul
3449     pdf.dest.y pdf.tmpc mul add
3450     pdf.dev.x add
3451     pdf.dest.x pdf.tmpb mul
3452     pdf.dest.y pdf.tmpd mul add
3453     pdf.dev.y add
3454   }
3455   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	3456 /pdf.bordertracking false def
pdf.brokenlink.skip	3457 /pdf.bordertracking.begin
pdf.brokenlink.dict	3458 {
pdf.bordertracking.endpage	3459 SDict /pdf.bordertracking true put
pdf.bordertracking.continue	3460 SDict /pdf.leftboundary undef
pdf.originx	3461 SDict /pdf.rightboundary undef
pdf.originy	3462 /a where
	3463 {
	3464 /a
	3465 {
	3466 currentpoint pop
	3467 SDict /pdf.rightboundary known dup
	3468 {
	3469 SDict /pdf.rightboundary get 2 index lt
	3470 { not }
	3471 if

```

3472     }
3473     if
3474     { pop }
3475     { SDict exch /pdf.rightboundary exch put }
3476     ifelse
3477     moveto
3478     currentpoint pop
3479     SDict /pdf.leftboundary known dup
3480     {
3481         SDict /pdf.leftboundary get 2 index gt
3482         { not }
3483         if
3484     }
3485     if
3486     { pop }
3487     { SDict exch /pdf.leftboundary exch put }
3488     ifelse
3489 }
3490 put
3491 }
3492 if
3493 }
3494 def
3495 /pdf.bordertracking.end
3496 {
3497     /a where { /a { moveto } put } if
3498     /x where { /x { 0 exch rmoveto } put } if
3499     SDict /pdf.leftboundary known
3500     { pdf.outerbox 0 pdf.leftboundary put }
3501     if
3502     SDict /pdf.rightboundary known
3503     { pdf.outerbox 2 pdf.rightboundary put }
3504     if
3505     SDict /pdf.bordertracking false put
3506 }
3507 def
3508 /pdf.bordertracking.endpage
3509 {
3510     pdf.bordertracking
3511     {
3512         pdf.bordertracking.end
3513         true setglobal
3514         pdf.globaldict
3515         /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3516         pdf.globaldict
3517         /pdf.brokenlink.skip pdf.baselineskip put
3518         pdf.globaldict
3519         /pdf.brokenlink.dict
3520         pdf.link.dict pdf.cvs put
3521         false setglobal
3522         mark pdf.link.dict cvx exec /Rect
3523         [
3524             pdf.llx
3525             pdf.lly

```

```

3526         pdf.outerbox 2 get pdf.linkmargin add
3527         currentpoint exch pop
3528         pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3529     ]
3530     /ANN pdf.pdfmark
3531 }
3532 if
3533 }
3534 def
3535 /pdf.bordertracking.continue
3536 {
3537     /pdf.link.dict pdf.globaldict
3538     /pdf.brokenlink.dict get def
3539     /pdf.outerbox pdf.globaldict
3540     /pdf.brokenlink.rect get def
3541     /pdf.baselineskip pdf.globaldict
3542     /pdf.brokenlink.skip get def
3543     pdf.globaldict dup dup
3544     /pdf.brokenlink.dict undef
3545     /pdf.brokenlink.skip undef
3546     /pdf.brokenlink.rect undef
3547     currentpoint
3548     /pdf.originy exch def
3549     /pdf.originx exch def
3550     /a where
3551     {
3552         /a
3553         {
3554             moveto
3555             SDict
3556             begin
3557                 currentpoint pdf.originy ne exch
3558                 pdf.originx ne or
3559                 {
3560                     pdf.save.linkll
3561                     /pdf.lly
3562                     pdf.lly pdf.outerbox 1 get sub def
3563                     pdf.bordertracking.begin
3564                 }
3565                 if
3566                 end
3567             }
3568             put
3569         }
3570     if
3571     /x where
3572     {
3573         /x
3574         {
3575             0 exch rmoveto
3576             SDict
3577             begin
3578                 currentpoint
3579                 pdf.originy ne exch pdf.originx ne or

```

```

3580         {
3581             pdf.save.linkll
3582             /pdf.lly
3583             pdf.lly pdf.outerbox 1 get sub def
3584             pdf.bordertracking.begin
3585         }
3586         if
3587         end
3588     }
3589     put
3590 }
3591 if
3592 }
3593 def

```

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

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

```

3594 /pdf.breaklink
3595 {
3596     pop
3597     counttomark 2 mod 0 eq
3598     {
3599         counttomark /pdf.count exch def
3600         {
3601             pdf.count 0 eq { exit } if
3602             counttomark 2 roll
3603             1 index /Rect eq
3604             {
3605                 dup 4 array copy
3606                 dup dup
3607                 1 get
3608                 pdf.outerbox pdf.rect.ht
3609                 pdf.linkmargin 2 mul add sub
3610                 3 exch put
3611                 dup
3612                 pdf.outerbox 2 get
3613                 pdf.linkmargin add
3614                 2 exch put
3615                 dup dup
3616                 3 get
3617                 pdf.outerbox pdf.rect.ht
3618                 pdf.linkmargin 2 mul add add
3619                 1 exch put
3620             } /pdf.currentrect exch def
3621             pdf.breaklink.write
3622             {
3623                 pdf.currentrect
3624                 dup
3625                 pdf.outerbox 0 get

```

```

3626         pdf.linkmargin sub
3627         0 exch put
3628     dup
3629         pdf.outerbox 2 get
3630         pdf.linkmargin add
3631         2 exch put
3632     dup dup
3633         1 get
3634         pdf.baselineskip add
3635         1 exch put
3636     dup dup
3637         3 get
3638         pdf.baselineskip add
3639         3 exch put
3640     /pdf.currentrect exch def
3641     pdf.breaklink.write
3642 }
3643     1 index 3 get
3644     pdf.linkmargin 2 mul add
3645     pdf.outerbox pdf.rect.ht add
3646     2 index 1 get sub
3647     pdf.baselineskip div round cvi 1 sub
3648     exch
3649     repeat
3650     pdf.currentrect
3651     dup
3652         pdf.outerbox 0 get
3653         pdf.linkmargin sub
3654         0 exch put
3655     dup dup
3656         1 get
3657         pdf.baselineskip add
3658         1 exch put
3659     dup dup
3660         3 get
3661         pdf.baselineskip add
3662         3 exch put
3663     dup 2 index 2 get 2 exch put
3664     /pdf.currentrect exch def
3665     pdf.breaklink.write
3666     SDict /pdf.pdfmark.good false put
3667     exit
3668 }
3669 { pdf.count 2 sub /pdf.count exch def }
3670 ifelse
3671 }
3672 loop
3673 }
3674 if
3675 /ANN
3676 }
3677 def
3678 /pdf.breaklink.write
3679 {

```

```

3680     counttomark 1 sub
3681     index /_objdef eq
3682     {
3683         counttomark -2 roll
3684         dup wcheck
3685         {
3686             readonly
3687             counttomark 2 roll
3688         }
3689         { pop pop }
3690         ifelse
3691     }
3692     if
3693     counttomark 1 add copy
3694     pop pdf.currentrect
3695     /ANN pdfmark
3696 }
3697 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.

```

3698 /pdf.pdfmark
3699 {
3700     SDict /pdf.pdfmark.good true put
3701     dup /ANN eq
3702     {
3703         pdf.pdfmark.store
3704         pdf.pdfmark.dict
3705         begin
3706             Subtype /Link eq
3707             currentdict /Rect known and
3708             SDict /pdf.outerbox known and
3709             SDict /pdf.baselineskip known and
3710             {
3711                 Rect 3 get
3712                 pdf.linkmargin 2 mul add
3713                 pdf.outerbox pdf.rect.ht add
3714                 Rect 1 get sub
3715                 pdf.baselineskip div round cvi 0 gt
3716                 { pdf.breaklink }
3717                 if
3718             }
3719             if
3720             end
3721             SDict /pdf.outerbox undef
3722             SDict /pdf.baselineskip undef
3723             currentdict /pdf.pdfmark.dict undef
3724         }
3725     if

```



```

3726     pdf.pdfmark.good
3727     { pdfmark }
3728     { cleartomark }
3729     ifelse
3730   }
3731   def
3732 /pdf.pdfmark.store
3733 {
3734   /pdf.pdfmark.dict 65534 dict def
3735   counttomark 1 add copy
3736   pop
3737   {
3738     dup mark eq
3739     {
3740       pop
3741       exit
3742     }
3743     {
3744       pdf.pdfmark.dict
3745       begin def end
3746     }
3747     ifelse
3748   }
3749   loop
3750 }
3751 def

```

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

```

3752 </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.

A	
<code>\AtBeginDvi</code>	57
B	
bool commands:	
<code>\bool_gset_false:N</code>	1192, 1211, 1234, 1256, 1272, 1373, 1612, 1648, 2385, 2431
<code>\bool_gset_true:N</code>	1190, 1259, 1371, 1627, 2378, 2384
<code>\bool_if:NTF</code>	67, 569, 1202, 1206, 1222, 1225, 1229, 1240, 1247, 1251, 1263, 1267, 1384, 1389, 1394, 1586, 1631, 1770, 1814, 1953, 1995, 2373, 2388, 2393, 2398
<code>\bool_if:nTF</code>	2607, 2860, 3096
<code>\bool_lazy_and:nnTF</code>	782, 2112, 3225, 3259
<code>\bool_lazy_or:nnTF</code>	1806, 1988
<code>\bool_new:N</code>	1193, 1260, 1374, 1628, 2358, 2359
<code>\bool_set_false:N</code>	1781, 1917, 2019, 2183
box commands:	
<code>\box_dp:N</code>	208, 210, 258, 260, 315, 317, 364, 366, 368, 370, 2410, 2443, 2444, 2469
<code>\box_ht:N</code>	210, 260, 317, 368, 370, 1826, 2060, 2415, 2454, 2455, 2471
<code>\box_if_empty:N</code>	2504
<code>\box_move_down:nn</code>	2332, 2410
<code>\box_move_up:nn</code>	2204, 2334, 2415
<code>\box_new:N</code>	2229, 2322, 2323
<code>\box_set_dp:Nn</code>	1711
<code>\box_set_ht:Nn</code>	1710
<code>\box_set_wd:Nn</code>	272, 1709
<code>\box_use:N</code>	215, 233, 247, 263, 290, 304, 320, 336, 348, 399, 413, 432, 1324, 1519, 1712, 2363
<code>\box_wd:N</code>	209, 217, 259, 265, 316, 322, 365, 367, 1825, 2059
box internal commands:	
<code>__box_backend_clip:N</code>	197, 197, 252, 252, 309, 309, 353, 353
<code>\l_box_backend_cos_fp</code>	267
<code>__box_backend_rotate:Nn</code>	219, 219, 267, 267, 324, 324, 403, 403
<code>__box_backend_rotate_aux:Nn</code>	219, 220, 221, 267, 268, 269, 324, 325, 326
<code>__box_backend_scale:Nnn</code>	236, 236, 295, 295, 339, 339, 416, 416
<code>\l_box_backend_sin_fp</code>	267
C	
clist commands:	
<code>\clist_map_function:nN</code>	1280, 1404, 1655
color internal commands:	
<code>__color_backend:nnn</code>	1018, 1025, 1040, 1048, 1054
<code>__color_backend_cmyk:w</code>	1019
<code>\g_color_backend_colorant_prop</code>	535, 554, 557, 577, 818
<code>__color_backend_devicen_colorants:n</code>	536, 536, 738, 876
<code>__color_backend_devicen_colorants:w</code>	536, 544, 551, 559
<code>__color_backend_devicen_init:nnn</code>	725, 725, 843, 843, 1075, 1075
<code>__color_backend_devicen_init:w</code>	843, 852, 881, 885
<code>__color_backend_fill:n</code>	922, 922, 924, 925, 926, 948, 949, 951, 953, 954, 973, 982, 983, 985, 987, 988, 999, 1008, 1009, 1011, 1013, 1014
<code>__color_backend_fill_cmyk:n</code>	922, 924, 948, 948, 982, 982, 1008, 1008
<code>__color_backend_fill_devicen:nn</code>	932, 942, 972, 976, 998, 1002, 1069, 1071
<code>__color_backend_fill_gray:n</code>	922, 925, 948, 950, 982, 984, 1008, 1010
<code>__color_backend_fill_reset:</code>	944, 944, 978, 978, 1004, 1004, 1073, 1073
<code>__color_backend_fill_rgb:n</code>	922, 926, 948, 952, 982, 986, 1008, 1012
<code>__color_backend_fill_separation:nn</code>	932, 932, 942, 972, 972, 976, 998, 998, 1002, 1069, 1069, 1071
<code>\l_color_backend_fill_tl</code>	498, 510, 956, 970
<code>__color_backend_iccbased_device:nnn</code>	905, 905

_color_backend_iccbased_- 567, 679, 749, 796, 821
init:nnn	
..... 744, 744, 887, 887, 1075, 1076	
_color_backend_init_resource:n	
..... 779, 779, 808, 879, 903, 918	
_color_backend_reset:	
..... 479, 494, 502, 514,	
518, 523, 944, 945, 978, 979, 1004, 1073	
_color_backend_rgb:w	1042
_color_backend_select:n	
..... 479, 480, 482, 484,	
486, 487, 518, 518, 520, 521, 522, 564	
_color_backend_select:nn	
..... 502, 503, 505, 507, 508, 775	
_color_backend_select_cmyk:n ..	
..... 479, 479, 502, 502, 518, 520	
_color_backend_select_devicen:nn	
..... 563, 565, 747, 748, 769, 777	
_color_backend_select_gray:n ..	
..... 479, 481, 502, 504, 518, 521, 528	
_color_backend_select_iccbased:nn	
..... 566, 566, 751, 751, 769, 778	
_color_backend_select_named:n ..	
..... 479, 483, 525, 525	
_color_backend_select_rgb:n ..	
..... 479, 485, 502, 506, 518, 522	
_color_backend_select_separation:nn	
..... 563, 563, 565,	
747, 747, 748, 769, 770, 774, 777, 778	
_color_backend_separation_-	
init:n	567, 648, 661
_color_backend_separation_-	
init:nn	796, 806, 810
_color_backend_separation_-	
init:nnn	567, 602, 623
_color_backend_separation_-	
init:nnnn	567, 625, 637
_color_backend_separation_-	
init:nnnnn	567,
567, 588, 681, 749, 749, 796, 796, 836	
_color_backend_separation_-	
init:nw	567, 652, 663, 677
_color_backend_separation_-	
init:w	567, 639, 654, 659
_color_backend_separation_-	
init_/DeviceCMYK:nnn	567
_color_backend_separation_-	
init_/DeviceGray:nnn	567
_color_backend_separation_-	
init_/DeviceRGB:nnn	567
_color_backend_separation_-	
init_aux:nnnnnn	567, 573, 589
_color_backend_separation_-	
init_CIELAB:nnn	
_color_backend_separation_-	
init_CIELAB:nnnnnn	750
_color_backend_separation_-	
init_count:n	567, 626, 629
_color_backend_separation_-	
init_count:w	567, 630, 631, 635
_color_backend_separation_-	
init_Device:Nn	
..... 567, 611, 613, 615, 616	
\l_color_backend_stack_int	
..... 440, 512, 515, 957, 969	
_color_backend_stroke:n	
..... 922, 927, 929,	
930, 931, 948, 961, 963, 965, 966, 975	
_color_backend_stroke_cmyk:n ..	
..... 922, 929, 948, 960, 982, 992, 1018, 1018	
_color_backend_stroke_cmyk:w ..	
..... 1018, 1020	
_color_backend_stroke_devicen:nn	
..... 932,	
943, 972, 977, 998, 1003, 1069, 1072	
_color_backend_stroke_gray:n ..	
..... 922, 930, 948, 962, 982, 994, 1018, 1031	
_color_backend_stroke_gray_-	
aux:n	1018, 1035, 1039
_color_backend_stroke_reset: ..	
..... 944,	
945, 978, 979, 1004, 1005, 1073, 1074	
_color_backend_stroke_rgb:n	
..... 922, 931, 948, 964, 982, 996, 1018, 1041	
_color_backend_stroke_rgb:w	
..... 1018, 1043	
_color_backend_stroke_separation:nn	
..... 932, 937, 943, 972, 974,	
977, 998, 1000, 1003, 1069, 1070, 1072	
\l_color_backend_stroke_tl	
..... 498, 511, 958, 968	
\g_color_model_int 574, 583, 731,	
759, 808, 814, 815, 869, 870, 879, 903	
\c_color_model_range_CIELAB_tl ..	
..... 686, 721, 832, 839	
color.sc	479, 3371
cs commands:	
\cs_generate_variant:Nn	
49, 63, 66, 99, 138, 143, 154, 185,	
191, 588, 1139, 1334, 1528, 1967,	
2030, 2050, 2233, 2254, 2317, 2811,	
2824, 2934, 2955, 2985, 3222, 3310	
\cs_gset:Npx ..	2619, 2623, 3101, 3106
\cs_gset_protected:Npn	3263
\cs_if_exist:NTF	
..... 27, 50, 1722, 2500, 2885, 2911	
\cs_if_exist_p:N	783, 3226, 3260

\cs_if_exist_use:NTF	38, 601	1682, 1684, 1686, 1688, 1690, 1698,
\cs_new:Npn		1720, 1739, 1762, 1778, 1790, 1795,
. 551, 610, 612, 614, 616, 623, 629,		1803, 1829, 1842, 1860, 1870, 1886,
631, 637, 654, 661, 663, 881, 1285,		1905, 1914, 1922, 1934, 1940, 1943,
1409, 1659, 1828, 2063, 2221, 2246,		1958, 1968, 2007, 2016, 2022, 2028,
2318, 2320, 2353, 2525, 2625, 2626,		2031, 2038, 2051, 2056, 2064, 2071,
2778, 2793, 2812, 2813, 2916, 2948,		2088, 2122, 2153, 2154, 2156, 2158,
2986, 2988, 3004, 3028, 3109, 3110,		2160, 2166, 2172, 2180, 2186, 2189,
3118, 3130, 3135, 3136, 3141, 3142		2191, 2202, 2231, 2234, 2236, 2239,
\cs_new:Npx		2248, 2255, 2272, 2277, 2282, 2287,
536, 2646, 2681, 2825, 2836, 2903, 3123		2297, 2302, 2310, 2325, 2330, 2362,
\cs_new_eq:NN	46, 57, 59,	2364, 2369, 2371, 2376, 2391, 2396,
520, 521, 522, 565, 748, 777, 778,		2433, 2462, 2481, 2490, 2527, 2534,
924, 925, 926, 929, 930, 931, 942,		2560, 2565, 2593, 2605, 2617, 2621,
943, 944, 945, 976, 977, 978, 979,		2627, 2629, 2633, 2657, 2659, 2661,
1002, 1003, 1004, 1071, 1072, 1073,		2672, 2692, 2702, 2725, 2739, 2749,
1138, 1333, 1339, 1340, 1527, 1529,		2760, 2780, 2814, 2847, 2858, 2864,
1530, 1536, 1736, 1737, 1750, 1752,		2892, 2926, 2928, 2935, 2937, 2941,
1776, 1777, 1834, 1835, 1836, 1859,		2950, 2956, 2961, 2966, 2968, 2970,
1884, 1901, 1902, 1911, 1912, 1913,		2978, 2991, 3007, 3009, 3026, 3030,
1933, 1936, 1937, 1938, 2003, 2013,		3032, 3054, 3059, 3092, 3094, 3099,
2014, 2015, 2169, 2170, 2178, 2179,		3104, 3111, 3113, 3117, 3119, 3120,
2188, 2218, 2219, 2220, 2224, 2363		3121, 3122, 3124, 3125, 3126, 3127,
\cs_new_protected:Npn		3128, 3129, 3131, 3132, 3133, 3134,
.	47, 54, 61, 64, 72,	3137, 3138, 3139, 3140, 3143, 3144,
78, 83, 85, 89, 100, 110, 119, 128,		3147, 3166, 3173, 3179, 3184, 3189,
141, 144, 146, 148, 152, 157, 166,		3196, 3203, 3238, 3243, 3265, 3275,
176, 186, 197, 219, 221, 236, 252,		3281, 3287, 3313, 3315, 3317, 3319
267, 269, 295, 309, 324, 326, 339,		\cs_new_protected:Npx
353, 403, 416, 443, 457, 467, 479,	 567, 1054, 2875, 2932, 3011
481, 483, 485, 487, 494, 502, 504,		\cs_set_eq:NN 2521, 2522
506, 508, 514, 518, 523, 525, 563,		\cs_set_protected:Npn 2126
566, 589, 679, 725, 744, 747, 749,		
750, 751, 770, 774, 779, 796, 810,		
821, 843, 887, 905, 922, 927, 932,		
937, 948, 950, 952, 954, 960, 962,		
964, 966, 972, 974, 982, 984, 986,		
988, 992, 994, 996, 998, 1000, 1005,		
1008, 1010, 1012, 1014, 1018, 1020,		
1031, 1039, 1041, 1043, 1069, 1070,		
1074, 1075, 1076, 1140, 1145, 1150,		
1152, 1154, 1162, 1170, 1179, 1189,		
1191, 1194, 1196, 1213, 1218, 1236,		
1258, 1261, 1274, 1287, 1292, 1294,		
1296, 1298, 1300, 1302, 1304, 1306,		
1311, 1335, 1337, 1341, 1346, 1351,		
1361, 1370, 1372, 1375, 1377, 1379,		
1381, 1386, 1391, 1396, 1398, 1411,		
1416, 1418, 1420, 1422, 1424, 1426,		
1428, 1430, 1441, 1466, 1478, 1490,		
1502, 1509, 1531, 1537, 1542, 1547,		
1558, 1568, 1578, 1580, 1582, 1584,		
1615, 1617, 1622, 1624, 1626, 1629,		
1650, 1661, 1674, 1676, 1678, 1680,		

D

dim commands:

\dim_compare:nNnTF	2102, 2107
\dim_compare_p:nNn	2113, 2114
\dim_eval:n	
. . .	2328, 2563, 2641, 2642, 2643,
	2700, 2735, 2736, 2737, 2998, 2999,
	3000, 3031, 3057, 3155, 3156, 3159
\dim_gset:Nn	3168, 3169
\dim_max:nn	2441, 2452
\dim_set:Nn	
. .	1825, 1826, 2059, 2060, 2098, 2099
\dim_set_eq:NN	2164
\dim_to_decimal:n . .	364, 365, 366,
	367, 368, 370, 1540, 1545, 1551,
	1552, 1553, 1554, 1563, 1564, 1565,
	1656, 1675, 2211, 2212, 2439, 2450,
	2468, 2469, 2470, 2471, 2475, 2531
\dim_to_decimal_in_bp:n	
. . . .	208, 209, 210, 258, 259, 260,
	315, 316, 317, 1158, 1159, 1166,

1167, 1174, 1175, 1183, 1184, 1185,
 1282, 1286, 1290, 1344, 1349, 1355,
 1356, 1357, 1365, 1366, 1406, 1410,
 1414, 1660, 1744, 1745, 1746, 1747,
 1927, 1928, 1929, 1930, 1982, 1983,
 1984, 1985, 2196, 2197, 2198, 2199
 \dim_zero:N 2096, 2097
 \c_max_dim
 .. 2098, 2099, 2102, 2107, 2113, 2114
 draw internal commands:
 __draw_align_currentpoint... .. 35
 __draw_backend_add_to_path:n ...
 1537,
 1539, 1544, 1549, 1560, 1568, 1583
 __draw_backend_begin:
 .. 1140, 1140, 1335, 1335, 1531, 1531
 __draw_backend_box_use:Nnnnn ...
 31, 1311, 1311, 1509, 1509, 1698, 1698
 __draw_backend_cap_but:
 .. 1274, 1294, 1398, 1418, 1650, 1678
 __draw_backend_cap_rectangle: ..
 .. 1274, 1298, 1398, 1422, 1650, 1682
 __draw_backend_cap_round:
 .. 1274, 1296, 1398, 1420, 1650, 1680
 __draw_backend_clip:
 .. 1194, 1258, 1375, 1391, 1582, 1626
 __draw_backend_closepath:
 1194, 1194,
 1215, 1375, 1375, 1582, 1582, 1619
 __draw_backend_closestroke: ...
 .. 1194, 1213, 1375, 1379, 1582, 1617
 __draw_backend_cm:nnnn
 ... 1306, 1306, 1319, 1320, 1321,
 1430, 1430, 1513, 1690, 1690, 1701
 __draw_backend_cm_aux:nnnn
 1430, 1437, 1441
 __draw_backend_cm_decompose:nnnnN
 1436, 1465, 1466
 __draw_backend_cm_decompose_-
 auxi:nnnnN 1465, 1470, 1478
 __draw_backend_cm_decompose_-
 auxii:nnnnN 1465, 1482, 1490
 __draw_backend_cm_decompose_-
 auxiii:nnnnN 1465, 1494, 1502
 __draw_backend_curveto:nnnnnn ..
 .. 1154, 1179, 1341, 1351, 1537, 1558
 __draw_backend_dash:n
 1274, 1280, 1285,
 1398, 1404, 1409, 1650, 1655, 1659
 __draw_backend_dash_aux:nn
 1650, 1654, 1661
 __draw_backend_dash_pattern:nn ..
 .. 1274, 1274, 1398, 1398, 1650, 1650
 __draw_backend_discardpath: ...
 .. 1194, 1261, 1375, 1396, 1582, 1629
 __draw_backend_end:
 .. 1140, 1145, 1335, 1337, 1531, 1536
 __draw_backend_evenodd_rule: ...
 .. 1189, 1189, 1370, 1370, 1578, 1578
 __draw_backend_fill:
 .. 1194, 1218, 1375, 1381, 1582, 1622
 __draw_backend_fillstroke:
 .. 1194, 1236, 1375, 1386, 1582, 1624
 __draw_backend_join_bevel:
 .. 1274, 1304, 1398, 1428, 1650, 1688
 __draw_backend_join_miter:
 .. 1274, 1300, 1398, 1424, 1650, 1684
 __draw_backend_join_round:
 .. 1274, 1302, 1398, 1426, 1650, 1686
 __draw_backend_lineto:nn
 .. 1154, 1162, 1341, 1346, 1537, 1542
 __draw_backend_linewidth:n
 .. 1274, 1287, 1398, 1411, 1650, 1674
 __draw_backend_literal:n
 1138, 1138, 1139, 1143,
 1147, 1151, 1153, 1156, 1164, 1172,
 1181, 1195, 1198, 1199, 1200, 1201,
 1204, 1210, 1220, 1227, 1233, 1238,
 1243, 1244, 1245, 1246, 1249, 1255,
 1265, 1271, 1276, 1289, 1293, 1295,
 1297, 1299, 1301, 1303, 1305, 1308,
 1313, 1314, 1315, 1316, 1317, 1318,
 1322, 1323, 1325, 1326, 1327, 1328,
 1329, 1333, 1333, 1334, 1343, 1348,
 1353, 1363, 1376, 1378, 1380, 1383,
 1388, 1393, 1397, 1400, 1413, 1417,
 1419, 1421, 1423, 1425, 1427, 1429,
 1527, 1527, 1528, 1589, 1608, 1634
 __draw_backend_miterlimit:n ...
 .. 1274, 1292, 1398, 1416, 1650, 1676
 __draw_backend_moveto:nn
 .. 1154, 1154, 1341, 1341, 1537, 1537
 __draw_backend_nonzero_rule: ...
 .. 1189, 1191, 1370, 1372, 1578, 1580
 __draw_backend_path:n
 1582, 1584, 1616, 1623, 1625
 \g__draw_backend_path_int 1597, 1614
 \g__draw_backend_path_tl
 1537, 1593, 1609, 1611, 1638
 __draw_backend_rectangle:nnnn ..
 .. 1154, 1170, 1341, 1361, 1537, 1547
 __draw_backend_scope_begin: 1150,
 1150, 1336, 1339, 1339, 1529, 1529
 __draw_backend_scope_end: 1150,
 1152, 1338, 1339, 1340, 1529, 1530
 __draw_backend_stroke: 1194, 1196,
 1216, 1375, 1377, 1582, 1615, 1620

\g__draw_draw_clip_bool ..	1194 , 1582	graphics internal commands:	
\g__draw_draw_eor_bool		__graphics_backend_dequote:w ...	1762 , 1798 , 1828
... ..	1189 , 1206 , 1222 , 1229 , 1240 , 1251 , 1267 , 1370 , 1384 , 1389 , 1394	\l__graphics_backend_dir_str .	1837
\g__draw_draw_path_int	1582	\l__graphics_backend_ext_str .	1837
\g__draw_path_tl	1647	__graphics_backend_get_pagecount:n	1751 , 1752 , 1886 , 1886 , 2001 , 2003 , 2071 , 2071 , 2223 , 2224
E		__graphics_backend_getbb_auxi:n	1762 , 1774 , 1788 , 1790
\errmessage	38	__graphics_backend_getbb_-	
\evensidemargin	2408	auxi:nN ...	2007 , 2011 , 2020 , 2022
exp commands:		__graphics_backend_getbb_-	
\exp_after:wN	2069	auxii:n	1762 , 1793 , 1795
\exp_args:Ne		__graphics_backend_getbb_-	
... ..	625 , 1797 , 1850 , 1876 , 2562 , 3056	auxii:nnN ..	2007 , 2025 , 2028 , 2030
\exp_args:Nf	1279 , 1403 , 2327	__graphics_backend_getbb_-	
\exp_args:NNf	220 , 268 , 325	auxiii:n	1762 , 1797 , 1803
\exp_args:Nnx	2981	__graphics_backend_getbb_-	
\exp_args:Nx	571 , 806 , 1848 , 1874 , 2284 , 2299 , 2404 , 3181 , 3240	auxiii:nNnn .	2007 , 2026 , 2029 , 2031
\exp_not:N	538 , 544 , 545 , 546 , 573 , 574 , 577 , 578 , 583 , 2648 , 2650 , 2653 , 2683 , 2685 , 2688 , 2827 , 2829 , 2832 , 2838 , 2840 , 2843 , 2880 , 2881 , 2887 , 2888 , 2907 , 2912 , 3013 , 3018	__graphics_backend_getbb_-	
\exp_not:n	48 , 97 , 108 , 136 , 895 , 2275 , 2280 , 2556 , 2797 , 2798 , 2812 , 2813 , 2959 , 2964 , 2975 , 3036	auxiv:nnNnn .	2007 , 2034 , 2038 , 2050
\ExplBackendFileDate	1	__graphics_backend_getbb_-	
F		auxvi:nNnn	2054 , 2056
file commands:		__graphics_backend_getbb_bmp:n .	1899 , 1913 , 2007 , 2015
\file_compare_timestamp:nNnTF .	1862	__graphics_backend_getbb_eps:n .	1734 , 1736 , 1837 , 1842 , 1859 , 1899 , 1901 , 2167 , 2169
\file_parse_full_name:nNNN	1844 , 1872	__graphics_backend_getbb_eps:nmm	1837
\fmtversion	52	__graphics_backend_getbb_eps:nn	1848 , 1860
fp commands:		__graphics_backend_getbb_jpeg:n	1762 , 1776 , 1899 , 1911 , 2007 , 2013 , 2172 , 2178
\fp_compare:nNnTF		__graphics_backend_getbb_jpg:n .	1762 , 1762 , 1776 , 1777 , 1899 , 1905 , 1911 , 1912 , 1913 , 2007 , 2007 , 2013 , 2014 , 2015 , 2172 , 2172 , 2178 , 2179
. .	227 , 274 , 280 , 332 , 1446 , 1459 , 1504	__graphics_backend_getbb_-	
\fp_eval:n .	220 , 229 , 242 , 243 , 268 , 285 , 300 , 302 , 325 , 334 , 345 , 346 , 410 , 425 , 426 , 1026 , 1027 , 1028 , 1036 , 1049 , 1050 , 1051 , 1448 , 1453 , 1454 , 1461 , 1471 , 1472 , 1473 , 1474 , 1483 , 1484 , 1485 , 1486 , 1495 , 1496 , 1497 , 1498 , 2553 , 2722 , 3050 , 3182 , 3192 , 3199 , 3241 , 3278 , 3285 , 3320	pagebox:w ..	2007 , 2046 , 2063 , 2069
\fp_new:N	293 , 294	__graphics_backend_getbb_pdf:n .	1762 , 1778 , 1868 , 1899 , 1914 , 2007 , 2016 , 2180 , 2180
\fp_set:Nn	273 , 276	__graphics_backend_getbb_png:n .	1762 , 1777 , 1899 , 1912 , 2007 , 2014 , 2172 , 2179
\fp_use:N	279 , 283 , 288	__graphics_backend_getbb_ps:n ..	1734 , 1737 , 1837 , 1859 , 1899 , 1902 , 2167 , 2170
\fp_zero:N	275		
\c_zero_fp	227 , 274 , 280 , 332 , 1446 , 1459		
G			
graphics commands:			
\l__graphics_search_ext_seq	1732 , 1755 , 1894 , 2082		

<code>__graphics_backend_getbb_svg:n</code> .	<code>\l__graphics_backend_name_str</code> .
..... 2088, 2088	1837
<code>__graphics_backend_getbb_svg_-</code>	<code>__graphics_bb_restore:nTF</code>
<code>auxi:nNn</code> ... 2088, 2104, 2109, 2122 1792, 2053, 2090
<code>__graphics_backend_getbb_svg_-</code>	<code>__graphics_bb_save:n</code> 1801, 2061, 2117
<code>auxii:w</code> 2088, 2126, 2148, 2153	<code>\l__graphics_decodearray_str</code> ...
<code>__graphics_backend_getbb_svg_-</code> 1768, 1769,
<code>auxiii:Nw</code> 2088, 2136, 2154	1780, 1808, 1812, 1813, 1916, 1951,
<code>__graphics_backend_getbb_svg_-</code>	1952, 1990, 1993, 1994, 2018, 2182
<code>auxiv:Nw</code> 2088, 2139, 2156	<code>__graphics_extract_bb:n</code>
<code>__graphics_backend_getbb_svg_-</code> 1909, 1918, 2176, 2184
<code>auxv:Nw</code> 2088, 2140, 2158	<code>\l__graphics_final_name_str</code> .. 1867
<code>__graphics_backend_getbb_svg_-</code>	<code>__graphics_get_pagecount:n</code>
<code>auxvi:Nn</code> 2088, 2155, 2157, 2159, 2160 1752, 2003, 2224
<code>__graphics_backend_getbb_svg_-</code>	<code>\l__graphics_graphics_attr_tl</code> ...
<code>auxvii:w</code> 2088, 2162, 2166 1761, 1766,
<code>__graphics_backend_include:nn</code> ..	1773, 1782, 1792, 1799, 1801, 1832
..... 2186, 2187, 2190, 2191	<code>\l__graphics_internal_box</code>
<code>__graphics_backend_include_-</code>	.. 1823, 1825, 1826, 2058, 2059, 2060
<code>auxi:nn</code> 1922, 1935, 1941, 1943	<code>\l__graphics_internal_dim</code> 2163, 2164
<code>__graphics_backend_include_-</code>	<code>\l__graphics_internal_ior</code>
<code>auxii:nnn</code> .. 1922, 1945, 1958, 1967 2092, 2093, 2100, 2119
<code>__graphics_backend_include_-</code>	<code>\l__graphics_interpolate_bool</code> ...
<code>auxiii:nnn</code> 1922, 1965, 1968 1770, 1781, 1807, 1814,
<code>__graphics_backend_include_-</code>	1917, 1953, 1989, 1995, 2019, 2183
<code>bmp:n</code> 1922, 1938	<code>\l__graphics_llx_dim</code>
<code>__graphics_backend_include_-</code> 1744, 1927, 1982, 2096, 2196
<code>dequote:w</code> 2202, 2213, 2221	<code>\l__graphics_lly_dim</code>
<code>__graphics_backend_include_-</code> 1745, 1928, 1983, 2097, 2197
<code>eps:n</code> 1739,	<code>\l__graphics_page_int</code>
1739, 1750, 1837, 1870, 1884, 1764, 1785, 1786, 1818,
1922, 1922, 1933, 2186, 2186, 2188	1819, 1907, 1949, 1950, 1976, 1977,
<code>__graphics_backend_include_-</code>	2009, 2024, 2025, 2067, 2068, 2174
<code>jpeg:n</code> . 1829, 1834, 1936, 2202, 2219	<code>\l__graphics_pagebox_tl</code>
<code>__graphics_backend_include_-</code> 54, 1765, 1784,
<code>jpg:n</code> 1829,	1820, 1821, 1908, 1947, 1948, 1978,
1829, 1834, 1835, 1836, 1922,	1980, 2010, 2033, 2034, 2069, 2175
1934, 1936, 1937, 1938, 2202, 2220	<code>__graphics_read_bb:n</code>
<code>__graphics_backend_include_-</code>	.. 1736, 1737, 1901, 1902, 2169, 2170
<code>jps:n</code> 1922	<code>\g__graphics_track_int</code>
<code>__graphics_backend_include_-</code> 1921, 1970, 1971
<code>pdf:n</code> 1829, 1835, 1874,	<code>\l__graphics_urx_dim</code>
1922, 1940, 2064, 2064, 2186, 2189 1746, 1825, 1929, 1984, 2059,
<code>__graphics_backend_include_-</code>	2098, 2102, 2105, 2113, 2198, 2211
<code>png:n</code> .. 1829, 1836, 1922, 1937, 2202, 2218	<code>\l__graphics_ury_dim</code>
<code>__graphics_backend_include_ps:n</code>	1747, 1826, 1930, 1985, 2060, 2099,
..... 1739, 1750,	2107, 2110, 2114, 2199, 2204, 2212
1837, 1884, 1922, 1933, 2186, 2188	group commands:
<code>__graphics_backend_include_-</code>	<code>\group_begin:</code> 163, 182
<code>svg:n</code> .. 2202, 2202, 2218, 2219, 2220	<code>\group_end:</code> 171
<code>__graphics_backend_loaded:n</code> ...	<code>\group_insert_after:N</code> ... 3257, 3307
1720, 1720, 1732, 1734, 1751, 1755,	
1894, 1899, 2002, 2082, 2167, 2223	

H

hbox commands:

`\hbox:n` 2206, 2333, 2336,

[2411](#), [2417](#), [2570](#), [2577](#), [3064](#), [3075](#)
`\hbox_overlap_right:n` [215](#),
[247](#), [263](#), [304](#), [320](#), [348](#), [432](#), [1324](#), [1519](#)
`\hbox_set:Nn` .. [1823](#), [2058](#), [2403](#), [2435](#)
`\hbox_set:Nw` [2386](#)
`\hbox_set_end:` [2401](#)
`\hbox_unpack:N` [2522](#)
hook commands:
`\hook_gput_code:nnn` .. [55](#), [1722](#), [1724](#)

I

int commands:
`\int_compare:nNnTF`
..... [1785](#), [1818](#), [1949](#), [1976](#),
[2024](#), [2067](#), [2494](#), [2595](#), [2878](#), [2906](#)
`\int_const:Nn` [445](#), [1799](#),
[1889](#), [1971](#), [2073](#), [2242](#), [2769](#), [2944](#)
`\int_eval:n` [465](#), [475](#), [621](#), [630](#), [643](#),
[645](#), [649](#), [662](#), [2619](#), [2623](#), [2856](#),
[2881](#), [2888](#), [2901](#), [3093](#), [3101](#), [3106](#)
`\int_gincr:N` [189](#),
[355](#), [1588](#), [1633](#), [1970](#), [2241](#), [2312](#),
[2343](#), [2420](#), [2943](#), [2980](#), [2993](#), [3013](#)
`\int_gset:Nn` [164](#), [183](#), [2483](#)
`\int_gset_eq:NN` [172](#), [2344](#), [2421](#), [2994](#)
`\int_if_exist:NTF` [1960](#)
`\int_if_odd:nTF` [2406](#)
`\int_max:nn` [2075](#)
`\int_new:N` [155](#),
[156](#), [402](#), [440](#), [1614](#), [1921](#), [2238](#),
[2324](#), [2355](#), [2357](#), [2939](#), [2990](#), [3006](#)
`\int_set_eq:NN` [160](#), [179](#), [2495](#)
`\int_step_function:nnnN` [647](#)
`\int_use:N`
. [357](#), [388](#), [574](#), [583](#), [731](#), [759](#), [808](#),
[814](#), [815](#), [869](#), [870](#), [879](#), [903](#), [1591](#),
[1597](#), [1604](#), [1636](#), [1644](#), [1786](#), [1819](#),
[1832](#), [1890](#), [1950](#), [1963](#), [1975](#), [1977](#),
[2068](#), [2076](#), [2247](#), [2314](#), [2319](#), [2347](#),
[2354](#), [2425](#), [2526](#), [2779](#), [2789](#), [2949](#),
[2982](#), [2987](#), [2997](#), [3005](#), [3018](#), [3029](#)
`\int_value:w`
..... [2648](#), [2683](#), [2827](#), [2838](#), [2856](#)
`\int_zero:N` ... [1764](#), [1907](#), [2009](#), [2174](#)
ior commands:
`\ior_close:N` [2119](#)
`\ior_if_eof:NTF` [2093](#)
`\ior_map_break:` [2115](#)
`\ior_open:Nn` [2092](#)
`\ior_str_map_inline:Nn` [2100](#)

K

kernel internal commands:
`__kernel_backend_align_begin:` ..
..... [72](#), [72](#), [200](#), [224](#), [239](#)

`__kernel_backend_align_end:` ...
..... [72](#), [78](#), [214](#), [232](#), [246](#)
`__kernel_backend_first_shipout:n`
..... [50](#), [54](#), [57](#), [59](#), [69](#), [571](#), [3149](#)
`\g_kernel_backend_header_bool` ..
..... [67](#), [569](#)
`__kernel_backend_literal:n`
..... [46](#), [46](#), [47](#), [48](#), [49](#),
[62](#), [65](#), [70](#), [74](#), [81](#), [84](#), [86](#), [142](#), [145](#),
[147](#), [149](#), [153](#), [329](#), [342](#), [489](#), [495](#),
[519](#), [524](#), [591](#), [727](#), [771](#), [923](#), [928](#),
[934](#), [939](#), [990](#), [1016](#), [1142](#), [1148](#),
[1443](#), [1450](#), [1456](#), [1516](#), [1521](#), [1741](#),
[1924](#), [1962](#), [1972](#), [2193](#), [2208](#), [2933](#),
[3031](#), [3093](#), [3097](#), [3102](#), [3107](#), [3151](#)
`__kernel_backend_literal_page:n`
..... [100](#),
[100](#), [144](#), [144](#), [2927](#), [2929](#), [3112](#), [3114](#)
`__kernel_backend_literal_pdf:n` .
..... [89](#), [89](#), [99](#), [141](#), [141](#),
[143](#), [255](#), [312](#), [1333](#), [3251](#), [3268](#), [3301](#)
`__kernel_backend_literal_-
postscript:n` [61](#),
[61](#), [63](#), [75](#), [76](#), [80](#), [201](#), [202](#), [204](#),
[205](#), [213](#), [225](#), [240](#), [1138](#), [2597](#), [2609](#)
`__kernel_backend_literal_svg:n` .
. [152](#), [152](#), [154](#), [159](#), [170](#), [178](#), [188](#),
[356](#), [358](#), [375](#), [753](#), [1527](#), [1702](#), [1713](#)
`__kernel_backend_matrix:n`
..... [128](#), [128](#), [138](#), [277](#), [298](#), [1433](#)
`__kernel_backend_postscript:n` ..
..... [64](#), [64](#),
[66](#), [491](#), [993](#), [995](#), [997](#), [1001](#), [2232](#),
[2289](#), [2304](#), [2333](#), [2339](#), [2379](#), [2411](#),
[2418](#), [2422](#), [2436](#), [2464](#), [2508](#), [2515](#),
[2521](#), [2529](#), [2536](#), [2570](#), [2577](#), [3205](#)
`__kernel_backend_scope:n`
..... [157](#), [186](#), [191](#), [385](#),
[390](#), [1056](#), [1534](#), [1579](#), [1581](#), [1601](#),
[1641](#), [1663](#), [1675](#), [1677](#), [1679](#), [1681](#),
[1683](#), [1685](#), [1687](#), [1689](#), [1692](#), [3320](#)
`__kernel_backend_scope_begin:` ..
[83](#), [83](#), [110](#), [110](#), [146](#), [146](#), [157](#), [157](#),
[199](#), [223](#), [238](#), [254](#), [271](#), [297](#), [311](#),
[328](#), [341](#), [1339](#), [1511](#), [1529](#), [1533](#), [1700](#)
`__kernel_backend_scope_begin:n` .
..... [157](#), [176](#), [185](#), [377](#), [405](#), [418](#)
`__kernel_backend_scope_end:` ...
..... [83](#), [85](#), [110](#), [119](#),
[146](#), [148](#), [157](#), [166](#), [216](#), [234](#), [248](#),
[264](#), [291](#), [305](#), [321](#), [337](#), [349](#), [400](#),
[414](#), [433](#), [1340](#), [1523](#), [1530](#), [1536](#), [1714](#)
`\g_kernel_backend_scope_int` ...
[155](#), [162](#), [164](#), [169](#), [173](#), [181](#), [183](#), [189](#)

\l_kernel_backend_scope_int	...
.....	155 , 161 , 174 , 180
\g_kernel_clip_path_int
	353 , 1588 , 1591 , 1604 , 1633 , 1636 , 1644
_kernel_color_backend_stack-	
init:Nnn 443 , 443 , 3230
_kernel_color_backend_stack-	
pop:n 457 , 467 , 515 , 3272
_kernel_color_backend_stack-	
push:nn
..	457 , 457 , 512 , 957 , 969 , 3254 , 3304
_kernel_dependency_version-	
check:Nn 1
_kernel_dependency_version-	
check:nn 27 , 29
_kernel_file_name_quote:n
.....	1850 , 1876
_kernel_kern:n
.....	2338 , 2340 , 2569 , 2573 ,
	2576 , 2580 , 3063 , 3071 , 3074 , 3090
L	
lua commands:	
\lua_load_module:n 1132
M	
\MessageBreak 40
mode commands:	
\mode_if_horizontal:TF	... 2485 , 2492
\mode_if_math:TF 2383
msg commands:	
\msg_error:nnn 529 , 2094
\msg_new:nnn 531
O	
\oddsidemargin 2407
opacity internal commands:	
_opacity_backend:nn
.....	3313 , 3314 , 3316 , 3318 , 3319
_opacity_backend:nnn	... 3179 ,
	3186 , 3187 , 3191 , 3198 , 3203 , 3222
_opacity_backend_fill:n
..	3179 , 3189 , 3275 , 3275 , 3313 , 3315
_opacity_backend_fill_stroke:nn
.....	3277 , 3283 , 3287 , 3310
\l_opacity_backend_fill_tl
.....	3236 , 3245 , 3284 , 3292
_opacity_backend_fillstroke:nn
.....	3275
_opacity_backend_reset:
.....	3238 , 3257 , 3265 , 3307
_opacity_backend_select:n
..	3179 , 3179 , 3238 , 3238 , 3313 , 3313
_opacity_backend_select_aux:n	...
.....	3179 , 3181 ,
	3184 , 3238 , 3240 , 3243 , 3263 , 3290
\c_opacity_backend_stack_int	...
.....	3225 , 3254 , 3272 , 3304
_opacity_backend_stroke:n
..	3179 , 3196 , 3275 , 3281 , 3313 , 3317
\l_opacity_backend_stroke_tl	...
.....	3236 , 3246 , 3279 , 3293
P	
pdf commands:	
\pdf_object_if_exist:nTF	823 , 889 , 907
\pdf_object_new:n
.....	814 , 825 , 869 , 891 , 909
\pdf_object_ref:n
.....	771 , 838 , 902 , 917 , 935 , 940
\pdf_object_ref_last:
.....	791 , 816 , 819 , 875
\pdf_object_unnamed_write:nn	...
.....	798 , 845 , 901 , 916
\pdf_object_write:nnn
.....	815 , 826 , 870 , 892 , 910
pdf internal commands:	
_pdf_backend:n	. 2932 , 2932 , 2934 ,
	2936 , 2938 , 2958 , 2963 , 2972 , 2995 ,
	3014 , 3027 , 3034 , 3066 , 3067 , 3077
_pdf_backend_annotation:nnnn	..
.....	2325 , 2325 ,
	2633 , 2633 , 2991 , 2991 , 3117 , 3117
_pdf_backend_annotation-	
aux:nnnn 2327 , 2330
\g_pdf_backend_annotation_int	..
..	2324 , 2344 , 2354 , 2990 , 2994 , 3005
_pdf_backend_annotation_last:	.
.....	2353 , 2353 ,
	2646 , 2646 , 3004 , 3004 , 3118 , 3118
_pdf_backend_bdc:nn	2627 , 2627 ,
	2926 , 2926 , 3111 , 3111 , 3143 , 3143
_pdf_backend_catalog_gput:nn	..
.....	2234 , 2234 ,
	2739 , 2739 , 2935 , 2935 , 3127 , 3127
_pdf_backend_compress_objects:n
.....	2593 , 2605 ,
	2847 , 2858 , 3092 , 3094 , 3137 , 3138
_pdf_backend_compresslevel:n	..
.....	2593 , 2593 ,
	2847 , 2847 , 3092 , 3092 , 3137 , 3137
\l_pdf_backend_content_box	3222 ,
	2386 , 2410 , 2413 , 2415 , 2444 , 2455
_pdf_backend_destination:nn	...

_pdf_backend_destination:nnnn .
 [2534](#), [2560](#),
 [2702](#), [2725](#), [3032](#), [3054](#), [3125](#), [3126](#)
 _pdf_backend_destination_
 aux:nnnn
 .. [2534](#), [2562](#), [2565](#), [3032](#), [3056](#), [3059](#)
 _pdf_backend_emc: .. [2627](#), [2629](#),
 [2926](#), [2928](#), [3111](#), [3113](#), [3143](#), [3144](#)
 _pdf_backend_info_gput:nn
 [2234](#), [2236](#),
 [2739](#), [2749](#), [2935](#), [2937](#), [3127](#), [3128](#)
 _pdf_backend_link:nw [2364](#)
 _pdf_backend_link_aux:nw ... [2364](#)
 _pdf_backend_link_begin:n
 [3007](#), [3008](#), [3010](#), [3011](#)
 _pdf_backend_link_begin:nnnw ..
 .. [2657](#), [2658](#), [2660](#), [2661](#), [3119](#), [3121](#)
 _pdf_backend_link_begin:nw ...
 [2366](#), [2370](#), [2371](#)
 _pdf_backend_link_begin_aux:nw
 [2374](#), [2376](#)
 _pdf_backend_link_begin_
 goto:nnw [2364](#), [2364](#),
 [2657](#), [2657](#), [3007](#), [3007](#), [3119](#), [3119](#)
 _pdf_backend_link_begin_
 user:nnw [2364](#), [2369](#),
 [2657](#), [2659](#), [3007](#), [3009](#), [3119](#), [3120](#)
 \g_pdf_backend_link_bool
 [2359](#), [2373](#), [2378](#), [2393](#), [2431](#)
 \g_pdf_backend_link_dict_tl ...
 [2356](#), [2381](#), [2426](#)
 _pdf_backend_link_end:
 [2364](#), [2391](#),
 [2657](#), [2672](#), [3007](#), [3026](#), [3119](#), [3122](#)
 _pdf_backend_link_end_aux: ...
 [2364](#), [2394](#), [2396](#)
 \g_pdf_backend_link_int
 [2355](#), [2421](#),
 [2425](#), [2526](#), [3006](#), [3013](#), [3018](#), [3029](#)
 _pdf_backend_link_last:
 [2525](#), [2525](#),
 [2681](#), [2681](#), [3028](#), [3028](#), [3123](#), [3123](#)
 _pdf_backend_link_margin:n ...
 [2527](#), [2527](#),
 [2692](#), [2692](#), [3030](#), [3030](#), [3124](#), [3124](#)
 \g_pdf_backend_link_math_bool ..
 [2358](#), [2384](#), [2385](#), [2388](#), [2398](#)
 _pdf_backend_link_minima:
 [2364](#), [2402](#), [2433](#)
 _pdf_backend_link_outerbox:n ..
 [2364](#), [2404](#), [2462](#)
 \g_pdf_backend_link_sf_int
 [2357](#), [2483](#), [2494](#), [2495](#)

_pdf_backend_link_sf_restore: .
 [2364](#), [2387](#), [2430](#), [2490](#)
 _pdf_backend_link_sf_save: ...
 [2364](#), [2382](#), [2400](#), [2481](#)
 \l_pdf_backend_model_box . [2323](#),
 [2403](#), [2435](#), [2443](#), [2454](#), [2469](#), [2471](#)
 _pdf_backend_objcompresslevel:n
 [2847](#), [2861](#), [2862](#), [2864](#)
 \g_pdf_backend_object_int
 [2238](#), [2241](#),
 [2244](#), [2312](#), [2314](#), [2319](#), [2343](#), [2344](#),
 [2347](#), [2420](#), [2421](#), [2939](#), [2943](#), [2946](#),
 [2980](#), [2982](#), [2987](#), [2993](#), [2994](#), [2997](#)
 _pdf_backend_object_last:
 [2318](#), [2318](#),
 [2825](#), [2825](#), [2986](#), [2986](#), [3129](#), [3135](#)
 _pdf_backend_object_new:n [2239](#),
 [2239](#), [2760](#), [2760](#), [2941](#), [2941](#), [3129](#)
 _pdf_backend_object_new:nn . [3129](#)
 _pdf_backend_object_now:nn ...
 [2310](#), [2310](#), [2317](#), [2814](#), [2814](#), [2824](#),
 [2978](#), [2978](#), [2985](#), [3129](#), [3133](#), [3134](#)
 \g_pdf_backend_object_prop
 [2759](#), [2939](#)
 _pdf_backend_object_ref:n
 [2239](#), [2246](#), [2251](#), [2760](#),
 [2778](#), [2941](#), [2948](#), [2953](#), [3129](#), [3130](#)
 _pdf_backend_object_write:nn ..
 [2780](#), [2791](#), [2793](#), [2822](#), [3129](#)
 _pdf_backend_object_write:nnn .
 [2248](#), [2248](#), [2254](#), [2780](#), [2780](#), [2811](#),
 [2950](#), [2950](#), [2955](#), [3129](#), [3131](#), [3132](#)
 _pdf_backend_object_write_
 array:nn ... [2248](#), [2272](#), [2950](#), [2956](#)
 _pdf_backend_object_write_
 aux:nnn [2248](#), [2250](#), [2255](#), [2313](#)
 _pdf_backend_object_write_
 dict:nn [2248](#), [2277](#), [2950](#), [2961](#)
 _pdf_backend_object_write_
 fstream:nn . [2248](#), [2282](#), [2950](#), [2966](#)
 _pdf_backend_object_write_
 fstream:nnn [2285](#), [2287](#)
 _pdf_backend_object_write_
 stream:nn .. [2248](#), [2297](#), [2950](#), [2968](#)
 _pdf_backend_object_write_
 stream:nnn [2248](#), [2300](#), [2302](#)
 _pdf_backend_object_write_
 stream:nnnn . [2950](#), [2967](#), [2969](#), [2970](#)
 _pdf_backend_pageobject_ref:n .
 [2320](#), [2320](#),
 [2836](#), [2836](#), [2988](#), [2988](#), [3129](#), [3136](#)
 _pdf_backend_pagesize_gset:nn .
 .. [3147](#), [3147](#), [3166](#), [3166](#), [3173](#), [3173](#)

_pdf_backend_pdfmark:n ..	2231 , 2231 , 2233 , 2235 , 2237 , 2257 , 2274 , 2279 , 2345 , 2537 , 2581 , 2628 , 2630	pdf.llx	2364 , 3385
_pdf_backend_version_major: 2619 , 2625 , 2625 , 2903 , 2903 , 3101 , 3102 , 3109 , 3109 , 3141 , 3141	pdf.lly	2364 , 3385
_pdf_backend_version_major_-		pdf.originx	3456
gset:n	2617 , 2617 , 2875 , 2875 , 3099 , 3099 , 3139 , 3139	pdf.originy	3456
_pdf_backend_version_minor: 2623 , 2625 , 2626 , 2903 , 2916 , 3106 , 3107 , 3109 , 3110 , 3141 , 3142	pdf.outerbox	2364 , 3698
_pdf_backend_version_minor_-		pdf.pdfmark	3698
gset:n	2617 , 2621 , 2875 , 2892 , 3099 , 3104 , 3139 , 3140	pdf.pdfmark.dict	3698
\l_pdf_breaklink_pdfmark_tl 2360 , 2428 , 2520	pdf.pdfmark.good	3698
_pdf_breaklink_postscript:n 2362 , 2362 , 2412 , 2414 , 2521	pdf.pt.dvi	3378
_pdf_breaklink_usebox:N 2363 , 2363 , 2413 , 2522	pdf.rect	3385
_pdf_exp_not_i:nn 2780 , 2801 , 2806 , 2812	pdf.rect.ht	3378
_pdf_exp_not_ii:nn 2780 , 2802 , 2807 , 2813	pdf.rightboundary	3456
\l_pdf_internal_box	2229	pdf.save.linkll	3385
pdf.baselineskip	2364 , 3698	pdf.save.linkur	3385
pdf.bordertracking	3456	pdf.save.ll	3385
pdf.bordertracking.begin	3456	pdf.save.ur	3385
pdf.bordertracking.continue	3456	pdf.tmpa	3421
pdf.bordertracking.end	3456	pdf.tmpb	3421
pdf.bordertracking.endpage	3456	pdf.tmpc	3421
pdf.breaklink	3594	pdf.tmpd	3421
pdf.breaklink.write	3594	pdf.urx	3385
pdf.brokenlink.dict	3456	pdf.ury	2364 , 3385
pdf.brokenlink.rect	3456	pdfmanagement commands:	
pdf.brokenlink.skip	3456	\pdfmanagement_add:nnn 788 , 3233 , 3247 , 3294 , 3297
pdf.count	3594	\pdfmanagement_if_active_p: 783 , 784 , 3226 , 3227 , 3260 , 3261
pdf.currentrect	3594	peek commands:	
pdf.cvs	3378	\peek_meaning:Ntf	2135 , 2138
pdf.dest.anchor	3421	\peek_remove_spaces:n	2133
pdf.dest.point	3421	prg commands:	
pdf.dest.x	3421	\prg_replicate:nn 168 , 619 , 640 , 650 , 851
pdf.dest.y	3421	prop commands:	
pdf.dest2device	3421	\prop_gput:Nnn	577 , 818
pdf.dev.x	3421	\prop_if_in:NnTF	554
pdf.dev.y	3421	\prop_item:Nn	557
pdf.dvi.pt	3378	\prop_new:N	535 , 2759 , 2940
pdf.globaldict	3375	\ProvidesExplFile	2
pdf.leftboundary	3456		
pdf.link.dict	2364		
pdf.linkdp.pad	2364 , 3382		
pdf.linkht.pad	2364 , 3382		
pdf.linkmargin	3382		

Q

quark commands:	
\quark_if_recursion_tail_stop:n	553
\q_recursion_stop	546
\q_recursion_tail	545

S

scan commands:	
\scan_stop:	113 , 122 , 475 , 2163 , 2166 , 2675 , 2700 , 2723 , 2737 , 2856 , 2873 , 2881 , 2888 , 2901
scan internal commands:	
\s_color_stop	630 , 631 , 635 , 639 , 652 , 655 , 659 , 663 , 677 , 852 , 881 , 885 , 1019 , 1021 , 1042 , 1044

<code>\tl_to_str:n</code>	2127, 2149, 2243,	1023, 1033, 1046, 1279, 1403, 1468,
	2247, 2770, 2779, 2790, 2945, 2949	1480, 1492, 1652, 2040, 2124, 2146
<code>\tl_use:N</code>	718, 831	<code>\use_none:n</code> 1669, 2498
token commands:		
<code>\c_math_toggle_token</code>	2389, 2399	V
		<code>\value</code> 2406
U		vbox commands:
use commands:		<code>\vbox_set:Nn</code> 2506
<code>\use:N</code>	43, 2270, 2952, 2981	<code>\vbox_to_zero:n</code> 2567, 2574, 3061, 3072
<code>\use:n</code>	59, 786, 812, 867,	<code>\vbox_unpack_drop:N</code> 2514