File I
Implementation

1 l3backend-basics implementation

```
1 ⟨∗package⟩
```

Whilst there is a reasonable amount of code overlap between backends, it is much clearer to have the blocks more-or-less separated than run in together and DocStripped out in parts. As such, most of the following is set up on a per-backend basis, though there is some common code (again given in blocks not interspersed with other material).

All the file identifiers are up-front so that they come out in the right place in the files.

```
2  \ProvidesExplFile
3 ⟨∗dvipdfmx⟩
4    {l3backend-dvipdfmx.def}{2025-10-09}{}
5    {L3 backend support: dvipdfmx}
6 ⟨/dvipdfmx⟩
7 ⟨∗dvips⟩
8    {l3backend-dvips.def}{2025-10-09}{}
9    {L3 backend support: dvips}
10 ⟨/dvips⟩
11 ⟨∗dvisvgm⟩
12    {l3backend-dvisvgm.def}{2025-10-09}{}
13    {L3 backend support: dvisvgm}
14 ⟨/dvisvgm⟩
15 ⟨∗luatex⟩
16    {l3backend-luatex.def}{2025-10-09}{}
17    {L3 backend support: PDF output (LuaTeX)}
18 ⟨/luatex⟩
19 ⟨∗pdftex⟩
20    {l3backend-pdftex.def}{2025-10-09}{}
21    {L3 backend support: PDF output (pdfTeX)}
22 ⟨/pdftex⟩
23 ⟨∗xetex⟩
24    {l3backend-xetex.def}{2025-10-09}{}
25    {L3 backend support: XeTeX}
26 ⟨/xetex⟩
```

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

```
27  \cs_if_exist:NTF \__kernel_dependency_version_check:nn
28    {
29      \__kernel_dependency_version_check:nn {2023-10-10}
30 ⟨dvipdfmx⟩       {l3backend-dvipdfmx.def}
31 ⟨dvips⟩        {l3backend-dvips.def}
32 ⟨dvisvgm⟩        {l3backend-dvisvgm.def}
33 ⟨luatex⟩         {l3backend-luatex.def}
34 ⟨pdftex⟩         {l3backend-pdftex.def}
35 ⟨xetex⟩         {l3backend-xetex.def}
```

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

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

- Color support is either dvips-like or LuaTeX/pdfTeX-like.

- LuaTeX/pdfTeX and dvipdfmx/XƎTeX share drawing routines.

- XƎTeX is the same as dvipdfmx other than image size extraction so takes most of the same code.

__kernel_backend_literal:e
__kernel_backend_literal:n

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

```
46  \cs_new_eq:NN \__kernel_backend_literal:e \tex_special:D
47  \cs_new_protected:Npn \__kernel_backend_literal:n #1
48    { \__kernel_backend_literal:e { \exp_not:n {#1} } }
```

(*End of definition for* __kernel_backend_literal:e.)

__kernel_backend_first_shipout:n

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

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

(*End of definition for* __kernel_backend_first_shipout:n.)

1.1 dvips backend

```
59  ⟨∗dvips⟩
```

__kernel_backend_literal_postscript:n
__kernel_backend_literal_postscript:e

Literal PostScript can be included using a few low-level formats. Here, we use the form with no positioning: this is overall more convenient as a wrapper. Note that this does require that where position is important, an appropriate wrapper is included.

```
60  \cs_new_protected:Npn \__kernel_backend_literal_postscript:n #1
61    { \__kernel_backend_literal:n { ps:: #1 } }
62  \cs_generate_variant:Nn \__kernel_backend_literal_postscript:n { e }
```

(*End of definition for* `\__kernel_backend_literal_postscript:n`.)

`\__kernel_backend_postscript:n`
`\__kernel_backend_postscript:e`

PostScript data that does have positioning, and also applying a shift to `SDict` (which is not done automatically by `ps:` or `ps::`, in contrast to `!` or `"`).

```
63 \cs_new_protected:Npn \__kernel_backend_postscript:n #1
64   { \__kernel_backend_literal:n { ps: SDict ~ begin ~ #1 ~ end } }
65 \cs_generate_variant:Nn \__kernel_backend_postscript:n { e }
```

(*End of definition for* `\__kernel_backend_postscript:n`.)

PostScript for the header: a small saving but makes the code clearer. This is held until the start of shipout such that a document with no actual output does not write anything.

```
66 \bool_if:NT \g__kernel_backend_header_bool
67   {
68     \__kernel_backend_first_shipout:n
69       { \__kernel_backend_literal:n { header = l3backend-dvips.pro } }
70   }
```

`\__kernel_backend_align_begin:`
`\__kernel_backend_align_end:`

In `dvips` there is no built-in saving of the current position, and so some additional Post-Script is required to set up the transformation matrix and also to restore it afterwards. Notice the use of the stack to save the current position "up front" and to move back to it at the end of the process. Notice that the `[begin]`/`[end]` pair here mean that we can use a run of PostScript statements in separate lines: not *required* but does make the code and output more clear.

```
71 \cs_new_protected:Npn \__kernel_backend_align_begin:
72   {
73     \__kernel_backend_literal:n { ps::[begin] }
74     \__kernel_backend_literal_postscript:n { currentpoint }
75     \__kernel_backend_literal_postscript:n { currentpoint~translate }
76   }
77 \cs_new_protected:Npn \__kernel_backend_align_end:
78   {
79     \__kernel_backend_literal_postscript:n { neg~exch~neg~exch~translate }
80     \__kernel_backend_literal:n { ps::[end] }
81   }
```

(*End of definition for* `\__kernel_backend_align_begin:` *and* `\__kernel_backend_align_end:`.)

`\__kernel_backend_scope_begin:`
`\__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.

```
82 \cs_new_protected:Npn \__kernel_backend_scope_begin:
83   { \__kernel_backend_literal:n { ps:gsave } }
84 \cs_new_protected:Npn \__kernel_backend_scope_end:
85   { \__kernel_backend_literal:n { ps:grestore } }
```

(*End of definition for* `\__kernel_backend_scope_begin:` *and* `\__kernel_backend_scope_end:`.)

```
86 ⟨/dvips⟩
```

3

1.2 LuaTₑX and pdfTₑX backends

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:e`

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

```
88 \cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1
89   {
90 ⟨∗luatex⟩
91     \tex_pdfextension:D literal
92 ⟨/luatex⟩
93 ⟨∗pdftex⟩
94     \tex_pdfliteral:D
95 ⟨/pdftex⟩
96       { \exp_not:n {#1} }
97   }
98 \cs_new_protected:Npn \__kernel_backend_literal_pdf:e #1
99   {
100 ⟨∗luatex⟩
101    \tex_pdfextension:D literal
102 ⟨/luatex⟩
103 ⟨∗pdftex⟩
104    \tex_pdfliteral:D
105 ⟨/pdftex⟩
106      {#1}
107  }
```

(*End of definition for* `\__kernel_backend_literal_pdf:n`.)

`\__kernel_backend_literal_page:n`
`\__kernel_backend_literal_page:e`

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

```
108 \cs_new_protected:Npn \__kernel_backend_literal_page:n #1
109   {
110 ⟨∗luatex⟩
111    \tex_pdfextension:D literal ~
112 ⟨/luatex⟩
113 ⟨∗pdftex⟩
114    \tex_pdfliteral:D
115 ⟨/pdftex⟩
116      page { \exp_not:n {#1} }
117  }
118 \cs_new_protected:Npn \__kernel_backend_literal_page:e #1
119   {
120 ⟨∗luatex⟩
121    \tex_pdfextension:D literal ~
122 ⟨/luatex⟩
123 ⟨∗pdftex⟩
124    \tex_pdfliteral:D
125 ⟨/pdftex⟩
126      page {#1}
127  }
```

(*End of definition for* `\__kernel_backend_literal_page:n`.)

`\__kernel_backend_scope_begin:`
`\__kernel_backend_scope_end:`

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

```
128 \cs_new_protected:Npn \__kernel_backend_scope_begin:
129   {
130 ⟨*luatex⟩
131     \tex_pdfextension:D save \scan_stop:
132 ⟨/luatex⟩
133 ⟨*pdftex⟩
134     \tex_pdfsave:D
135 ⟨/pdftex⟩
136   }
137 \cs_new_protected:Npn \__kernel_backend_scope_end:
138   {
139 ⟨*luatex⟩
140     \tex_pdfextension:D restore \scan_stop:
141 ⟨/luatex⟩
142 ⟨*pdftex⟩
143     \tex_pdfrestore:D
144 ⟨/pdftex⟩
145   }
```

(*End of definition for* `\__kernel_backend_scope_begin:` *and* `\__kernel_backend_scope_end:`.)

`\__kernel_backend_matrix:n`
`\__kernel_backend_matrix:e`

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.

```
146 \cs_new_protected:Npn \__kernel_backend_matrix:n #1
147   {
148 ⟨*luatex⟩
149     \tex_pdfextension:D setmatrix
150 ⟨/luatex⟩
151 ⟨*pdftex⟩
152     \tex_pdfsetmatrix:D
153 ⟨/pdftex⟩
154       { \exp_not:n {#1} }
155   }
156 \cs_new_protected:Npn \__kernel_backend_matrix:e #1
157   {
158 ⟨*luatex⟩
159     \tex_pdfextension:D setmatrix
160 ⟨/luatex⟩
161 ⟨*pdftex⟩
162     \tex_pdfsetmatrix:D
163 ⟨/pdftex⟩
164       {#1}
165   }
```

(*End of definition for* `\__kernel_backend_matrix:n`.)

```
166 ⟨/luatex | pdftex⟩
```

1.3 dvipdfmx backend

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

_kernel_backend_literal_pdf:n
_kernel_backend_literal_pdf:e

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.

```
168 \cs_new_protected:Npn \__kernel_backend_literal_pdf:n #1
169   { \__kernel_backend_literal:n { pdf:literal~ #1 } }
170 \cs_generate_variant:Nn \__kernel_backend_literal_pdf:n { e }
```

(*End of definition for* __kernel_backend_literal_pdf:n.)

_kernel_backend_literal_page:n

Whilst the manual says this is like literal direct in pdfTEX, it closes the BT block!

```
171 \cs_new_protected:Npn \__kernel_backend_literal_page:n #1
172   { \__kernel_backend_literal:n { pdf:literal~direct~ #1 } }
```

(*End of definition for* __kernel_backend_literal_page:n.)

_kernel_backend_scope_begin:
__kernel_backend_scope_end:

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

```
173 \cs_new_protected:Npn \__kernel_backend_scope_begin:
174   { \__kernel_backend_literal:n { x:gsave } }
175 \cs_new_protected:Npn \__kernel_backend_scope_end:
176   { \__kernel_backend_literal:n { x:grestore } }
```

(*End of definition for* __kernel_backend_scope_begin: *and* __kernel_backend_scope_end:.)

1.4 dvisvgm backend

_kernel_backend_literal_svg:n
_kernel_backend_literal_svg:e

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

```
179 \cs_new_protected:Npn \__kernel_backend_literal_svg:n #1
180   { \__kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }
181 \cs_generate_variant:Nn \__kernel_backend_literal_svg:n { e }
```

(*End of definition for* __kernel_backend_literal_svg:n.)

\g__kernel_backend_scope_int
\l__kernel_backend_scope_int

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

```
182 \int_new:N \g__kernel_backend_scope_int
183 \int_new:N \l__kernel_backend_scope_int
```

(*End of definition for* \g__kernel_backend_scope_int *and* \l__kernel_backend_scope_int.)

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.

```
184 \cs_new_protected:Npn \__kernel_backend_scope_begin:
185   {
186     \__kernel_backend_literal_svg:n { <g> }
187     \int_set_eq:NN
188       \l__kernel_backend_scope_int
189       \g__kernel_backend_scope_int
190     \group_begin:
191       \int_gset:Nn \g__kernel_backend_scope_int { 1 }
192   }
193 \cs_new_protected:Npn \__kernel_backend_scope_end:
194   {
195     \prg_replicate:nn
196       { \g__kernel_backend_scope_int }
197       { \__kernel_backend_literal_svg:n { </g> } }
198     \group_end:
199     \int_gset_eq:NN
200       \g__kernel_backend_scope_int
201       \l__kernel_backend_scope_int
202   }
203 \cs_new_protected:Npn \__kernel_backend_scope_begin:n #1
204   {
205     \__kernel_backend_literal_svg:n { <g ~ #1 > }
206     \int_set_eq:NN
207       \l__kernel_backend_scope_int
208       \g__kernel_backend_scope_int
209     \group_begin:
210       \int_gset:Nn \g__kernel_backend_scope_int { 1 }
211   }
212 \cs_generate_variant:Nn \__kernel_backend_scope_begin:n { e }
213 \cs_new_protected:Npn \__kernel_backend_scope:n #1
214   {
215     \__kernel_backend_literal_svg:n { <g ~ #1 > }
216     \int_gincr:N \g__kernel_backend_scope_int
217   }
218 \cs_generate_variant:Nn \__kernel_backend_scope:n { e }
```

(*End of definition for* __kernel_backend_scope_begin: *and others.*)

```
219 ⟨/dvisvgm⟩
220 ⟨/package⟩
```

2 **l3backend-box** implementation

```
221 ⟨*package⟩
222 ⟨@@=box⟩
```

2.1 `dvips` backend

```
223 ⟨*dvips⟩
```

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

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

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

`\__box_backend_rotate:Nn`
`\__box_backend_rotate_aux: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.

```
246 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
247   { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
248 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
249   {
250     \__kernel_backend_scope_begin:
251     \__kernel_backend_align_begin:
252     \__kernel_backend_literal_postscript:e
253       {
254         \fp_compare:nNnTF {#2} = \c_zero_fp
255           { 0 }
256           { \fp_eval:n { round ( -(#2) , 5 ) } } ~
257         rotate
258       }
259     \__kernel_backend_align_end:
260     \box_use:N #1
261     \__kernel_backend_scope_end:
262   }
```

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

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

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

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

```
277 ⟨/dvips⟩
```

2.2   LuaTeX and pdfTeX backends

```
278 ⟨∗luatex | pdftex⟩
```

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

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

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

`\__box_backend_rotate:Nn`
`\__box_backend_rotate_aux:Nn`
`\l__box_backend_cos_fp`
`\l__box_backend_sin_fp`

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.

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

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

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

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

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

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

```
334  ⟨/luatex | pdftex⟩
```

2.3 dvipdfmx/X_ETEX backend

```
335  ⟨*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.

```
336  \cs_new_protected:Npn \__box_backend_clip:N #1
```

```
337     {
338       \__kernel_backend_scope_begin:
339       \__kernel_backend_literal_pdf:e
340         {
341           0~
342           \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
343           \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
344           \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
345           re~W~n
346         }
347       \hbox_overlap_right:n { \box_use:N #1 }
348       \__kernel_backend_scope_end:
349       \skip_horizontal:n { \box_wd:N #1 }
350     }
```

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

`\__box_backend_rotate:Nn`
`\__box_backend_rotate_aux:Nn`

Rotating in dvipdmfx/X$_{\overline{E}}$T$_E$X can be implemented using either PDF or backend-specific code. The former approach however is not "aware" of the content of boxes: this means that any embedded links would not be adjusted by the rotation. As such, the backend-native approach is preferred: the code therefore is similar (though not identical) to the dvips version (notice the rotation angle here is positive). As for dvips, zero rotation is written as 0 not -0.

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

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

`\__box_backend_scale:Nnn`

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

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

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

2.4 `dvisvgm` backend

`\__box_backend_clip:N`
`\g__kernel_clip_path_int`

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

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

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

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

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

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

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

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

(*End of definition for* __box_backend_rotate:Nn.)

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

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

13

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

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

```
462 ⟨/dvisvgm⟩
```

```
463 ⟨/package⟩
```

3 **l3backend-color** implementation

```
464 ⟨∗package⟩
465 ⟨@@=color⟩
```

Color support is split into parts: collecting data from LATEX 2$_\varepsilon$, 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/XꟻTEX in particular. Whilst it is in some ways convenient to use the same approach in multiple backends, the fact that dvipdfmx/XꟻTEX 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/XꟻTEX have multiple color stacks in recent releases, the way these interact with the original single stack and with other graphic state operations means that currently it is not feasible to use the multiple stacks.

3.1.1 Common code

```
466 ⟨∗luatex | pdftex⟩
```

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

```
467 \int_new:N \l__color_backend_stack_int
```

(*End of definition for* \l__color_backend_stack_int.)

```
468 ⟨/luatex | pdftex⟩
```

3.1.2 LuaTEXand pdfTEX

```
469 ⟨∗luatex | pdftex⟩
```

__kernel_color_backend_stack_init:Nnn

```
470 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
471   {
472     \int_const:Nn #1
473       {
474 ⟨∗luatex⟩
475         \tex_pdffeedback:D colorstackinit ~
476 ⟨/luatex⟩
```

14

```
477 ⟨*pdftex⟩
478         \tex_pdfcolorstackinit:D
479 ⟨/pdftex⟩
480         \tl_if_blank:nF {#2} { #2 ~ }
481         {#3}
482     }
483   }
```

(*End of definition for* `\__kernel_color_backend_stack_init:Nnn`.)

```
484 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
485   {
486 ⟨*luatex⟩
487     \tex_pdfextension:D colorstack ~
488 ⟨/luatex⟩
489 ⟨*pdftex⟩
490     \tex_pdfcolorstack:D
491 ⟨/pdftex⟩
492       \int_eval:n {#1} ~ push ~ {#2}
493   }
494 \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
495   {
496 ⟨*luatex⟩
497     \tex_pdfextension:D colorstack ~
498 ⟨/luatex⟩
499 ⟨*pdftex⟩
500     \tex_pdfcolorstack:D
501 ⟨/pdftex⟩
502       \int_eval:n {#1} ~ pop \scan_stop:
503   }
```

(*End of definition for* `\__kernel_color_backend_stack_push:nn` *and* `\__kernel_color_backend_stack_-pop:n`.)

```
504 ⟨/luatex | pdftex⟩
```

3.2 General color

3.2.1 `dvips`-style

```
505 ⟨*dvips | dvisvgm⟩
```

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

```
506 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
507   { \__color_backend_select:n { cmyk ~ #1 } }
508 \cs_new_protected:Npn \__color_backend_select_gray:n #1
509   { \__color_backend_select:n { gray ~ #1 } }
510 \cs_new_protected:Npn \__color_backend_select_named:n #1
511   { \__color_backend_select:n { ~ #1 } }
512 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
513   { \__color_backend_select:n { rgb ~ #1 } }
514 \cs_new_protected:Npn \__color_backend_select:n #1
515   {
516     \__kernel_backend_literal:n { color~push~ #1 }
```

```
517 ⟨*dvips⟩
518     \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
519 ⟨/dvips⟩
520   }
521 \cs_new_protected:Npn \__color_backend_reset:
522   { \__kernel_backend_literal:n { color~pop } }
```

(*End of definition for* __color_backend_select_cmyk:n *and others.*)

```
523 ⟨/dvips | dvisvgm⟩
```

3.2.2 LuaTeX and pdfTeX

```
524 ⟨*luatex | pdftex⟩
```

\l__color_backend_fill_tl
\l__color_backend_stroke_tl

```
525 \tl_new:N \l__color_backend_fill_tl
526 \tl_new:N \l__color_backend_stroke_tl
527 \tl_set:Nn \l__color_backend_fill_tl { 0 ~ g }
528 \tl_set:Nn \l__color_backend_stroke_tl { 0 ~ G }
```

(*End of definition for* \l__color_backend_fill_tl *and* \l__color_backend_stroke_tl.)

__color_backend_select_cmyk:n
__color_backend_select_gray:n
__color_backend_select_rgb:n
__color_backend_select:nn
__color_backend_reset:

Store the values then pass to the stack.

```
529 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
530   { \__color_backend_select:nn { #1 ~ k } { #1 ~ K } }
531 \cs_new_protected:Npn \__color_backend_select_gray:n #1
532   { \__color_backend_select:nn { #1 ~ g } { #1 ~ G } }
533 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
534   { \__color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
535 \cs_new_protected:Npn \__color_backend_select:nn #1#2
536   {
537     \tl_set:Nn \l__color_backend_fill_tl {#1}
538     \tl_set:Nn \l__color_backend_stroke_tl {#2}
539     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int { #1 ~ #2 }
540   }
541 \cs_new_protected:Npn \__color_backend_reset:
542   { \__kernel_color_backend_stack_pop:n \l__color_backend_stack_int }
```

(*End of definition for* __color_backend_select_cmyk:n *and others.*)

```
543 ⟨/luatex | pdftex⟩
```

3.2.3 dvipmdfx/XᴇTᴇX

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

```
544 ⟨*dvipdfmx | xetex⟩
```

16

`\__color_backend_select:n`
`\__color_backend_select_cmyk:n`
`\__color_backend_select_gray:n`
`\__color_backend_select_rgb:n`
`\__color_backend_reset:`

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

```
545 \cs_new_protected:Npn \__color_backend_select:n #1
546   { \__kernel_backend_literal:n { pdf : bc ~ [ #1 ] } }
547 \cs_new_eq:NN \__color_backend_select_cmyk:n \__color_backend_select:n
548 \cs_new_eq:NN \__color_backend_select_gray:n \__color_backend_select:n
549 \cs_new_eq:NN \__color_backend_select_rgb:n  \__color_backend_select:n
550 \cs_new_protected:Npn \__color_backend_reset:
551   { \__kernel_backend_literal:n { pdf : ec } }
```

(*End of 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`.

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

(*End of definition for* `\__color_backend_select_named:n`.)

```
560 ⟨/dvipdfmx | xetex⟩
```

3.3 Separations

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

```
561 ⟨*dvipdfmx | luatex | pdftex | xetex | dvips⟩
```

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

`\g__color_backend_colorant_prop`

```
562 \prop_new:N \g__color_backend_colorant_prop
```

(*End of definition for* `\g__color_backend_colorant_prop`.)

`\__color_backend_devicen_colorants:n`
`\__color_backend_devicen_colorants:w`

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

17

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

(*End of definition for* `\__color_backend_deviceen_colorants:n` *and* `\__color_backend_deviceen_colorants:w`.)

```
588 ⟨/dvipdfmx | luatex | pdftex | xetex | dvips⟩

589 ⟨∗dvips⟩
```

__color_backend_select_separation:nn
__color_backend_select_deviceen:nn

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

(*End of definition for* `\__color_backend_select_separation:nn` *and* `\__color_backend_select_deviceen:nn`.)

__color_backend_select_iccbased:nn

No support.

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

(*End of definition for* `\__color_backend_select_iccbased:nn`.)

__color_backend_separation_init:nnnnn
__color_backend_separation_init:neenn
__color_backend_separation_init_aux:nnnnnn
lor_backend_separation_init_/DeviceCMYK:nnn
lor_backend_separation_init_/DeviceGray:nnn
olor_backend_separation_init_/DeviceRGB:nnn
__color_backend_separation_init_Device:Nn
__color_backend_separation_init:nnn
__color_backend_separation_init_count:n
__color_backend_separation_init_count:w
__color_backend_separation_init:nnnn
__color_backend_separation_init:w
__color_backend_separation_init:n
__color_backend_separation_init:nw
__color_backend_separation_init_CIELAB:nnn

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

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

18

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

For the generic case, we cannot use /FunctionType 2 unfortunately, so we have to code that idea up in PostScript. Here, we will therefore assume that a range is *always* given. First, we count values in each argument: at the backend level, we can assume there are always well-behaved with spaces present.

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

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

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

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

Finally, we deal with the range limit if required. This is handled by splitting the range into pairs. It's then just a question of doing the comparisons, this time dropping everything except the desired result.

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

```
705    }
```

CIELAB support uses the detail from the PostScript reference, page 227; other than that block of PostScript, this is the same as for PDF-based routes.

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

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

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

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

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

(*End of definition for* `\__color_backend_devicen_init:nnn`.)

`\__color_backend_iccbased_init:nnn`   No support at present.

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

(*End of definition for* `\__color_backend_iccbased_init:nnn`.)

```
772 ⟨/dvips⟩
```

```
773 ⟨*dvisvgm⟩
```

`\__color_backend_select_separation:nn`   No support at present.
`\__color_backend_select_devicen:nn`

```
774 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2 { }
775 \cs_new_eq:NN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn
```

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

`\__color_backend_separation_init:nnnnn`   No support at present.
`\__color_backend_separation_init_CIELAB:nnn`

```
776 \cs_new_protected:Npn \__color_backend_separation_init:nnnnn #1#2#3#4#5 { }
777 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnnnnn #1#2#3 { }
```

(*End of definition for* `\__color_backend_separation_init:nnnnn` *and* `\__color_backend_separation_-`
`init_CIELAB:nnn`.)

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

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

22

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

(*End of definition for* __color_backend_select_iccbased:nn*.*)

```
794 ⟨/dvisvgm⟩
795 ⟨∗dvipdfmx | luatex | pdftex | xetex⟩
```

__color_backend_select_separation:nn
__color_backend_select_devicen:nn
__color_backend_select_iccbased:nn

```
796 ⟨∗dvipdfmx | xetex⟩
797 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
798    { \__kernel_backend_literal:e { pdf : bc ~ \pdf_object_ref:n {#1} ~ [ #2 ] } }
799 ⟨/dvipdfmx | xetex⟩
800 ⟨∗luatex | pdftex⟩
801 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
802    { \__color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn  } { /#1 ~ CS ~ #2 ~ SCN } }
803 ⟨/luatex | pdftex⟩
804 \cs_new_eq:NN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn
805 \cs_new_eq:NN \__color_backend_select_iccbased:nn \__color_backend_select_separation:nn
```

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

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

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

(*End of definition for* __color_backend_init_resource:n*.*)

__color_backend_separation_init:nnnnn
__color_backend_separation_init:nn
__color_backend_separation_init_CIELAB:nnn

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

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

23

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

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

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

(*End of definition for* __color_backend_separation_init:nnnnn, __color_backend_separation_-
init:nn, *and* __color_backend_separation_init_CIELAB:nnn.)

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

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

24

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

(*End of definition for* __color_backend_devicen_init:nnn *and* __color_backend_devicen_init:w*.*)

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

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

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

(*End of definition for* `\__color_backend_iccbased_init:nnn.`)

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

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

(*End of definition for* `\__color_backend_iccbased_device:nnn.`)

```
947 ⟨/dvipdfmx | luatex | pdftex | xetex⟩
```

3.4 Fill and stroke color

Here, dvipdfmx/X$_{\overline{E}}$T$_{E}$X 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). LuaT$_{E}$X and pdfT$_{E}$X have multiple stacks that can deal with fill and stroke. For dvips we have to manage fill and stroke color ourselves. We also handle dvisvgm independently, as there we can create SVG directly.

```
948 ⟨*dvipdfmx | xetex⟩
```

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

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

(*End of definition for* `\__color_backend_fill:n` *and others.*)

```
959 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
960   {
961     \__kernel_backend_literal:e
962       { pdf : bc ~ fill ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
963   }
964 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
965   {
966     \__kernel_backend_literal:e
967       { pdf : bc ~ stroke ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
968   }
969 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
970 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn
```

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

```
971 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
972 \cs_new_eq:NN \__color_backend_stroke_reset: \__color_backend_reset:
```

(*End of definition for* `\__color_backend_fill_reset:` *and* `\__color_backend_stroke_reset:`.)

```
973 ⟨/dvipdfmx | xetex⟩
974 ⟨*luatex | pdftex⟩
```

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

```
975 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
976   { \__color_backend_fill:n { #1 ~ k } }
977 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
978   { \__color_backend_fill:n { #1 ~ g } }
979 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
980   { \__color_backend_fill:n { #1 ~ rg } }
981 \cs_new_protected:Npn \__color_backend_fill:n #1
982   {
983     \tl_set:Nn \l__color_backend_fill_tl {#1}
984     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
985       { #1 ~ \l__color_backend_stroke_tl }
986   }
987 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
988   { \__color_backend_stroke:n { #1 ~ K } }
989 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
990   { \__color_backend_stroke:n { #1 ~ G } }
991 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
992   { \__color_backend_stroke:n { #1 ~ RG } }
993 \cs_new_protected:Npn \__color_backend_stroke:n #1
994   {
995     \tl_set:Nn \l__color_backend_stroke_tl {#1}
996     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
997       { \l__color_backend_fill_tl \c_space_tl #1 }
998   }
```

(*End of definition for* `\__color_backend_fill_cmyk:n` *and others.*)

`\__color_backend_fill_separation:nn`
`\__color_backend_stroke_separation:nn`
`\__color_backend_fill_devicen:nn`
`\__color_backend_stroke_devicen:nn`

```
999 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
1000   { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
1001 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1002   { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
1003 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1004 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn
```

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

`\__color_backend_fill_reset:`
`\__color_backend_stroke_reset:`

```
1005 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1006 \cs_new_eq:NN \__color_backend_stroke_reset: \__color_backend_reset:
```

(*End of definition for* `\__color_backend_fill_reset:` *and* `\__color_backend_stroke_reset:`.)

```
1007 ⟨/luatex | pdftex⟩
```

```
1008 ⟨*dvips⟩
```

`\__color_backend_fill_cmyk:n`
`\__color_backend_fill_gray:n`
`\__color_backend_fill_rgb:n`
`\__color_backend_fill:n`
`\__color_backend_stroke_cmyk:n`
`\__color_backend_stroke_gray:n`
`\__color_backend_stroke_rgb:n`

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

```
1009 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1010   { \__color_backend_fill:n { cmyk ~ #1 } }
1011 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1012   { \__color_backend_fill:n { gray ~ #1 } }
1013 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1014   { \__color_backend_fill:n { rgb ~ #1 } }
1015 \cs_new_protected:Npn \__color_backend_fill:n #1
1016   {
1017     \__kernel_backend_literal:n { color~push~ #1 }
1018   }
1019 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1020   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1021 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1022   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1023 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1024   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }
```

(*End of definition for* `\__color_backend_fill_cmyk:n` *and others.*)

`\__color_backend_fill_separation:nn`
`\__color_backend_stroke_separation:nn`
`\__color_backend_fill_devicen:nn`
`\__color_backend_stroke_devicen:nn`

```
1025 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
1026   { \__color_backend_fill:n { separation ~ #1 ~ #2 } }
1027 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1028   { \__kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1029 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1030 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn
```

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

`\__color_backend_fill_reset:`
`\__color_backend_stroke_reset:`

```
1031 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1032 \cs_new_protected:Npn \__color_backend_stroke_reset: { }
```

(*End of definition for* `\__color_backend_fill_reset:` *and* `\__color_backend_stroke_reset:`.)

```
1033 ⟨/dvips⟩
1034 ⟨*dvisvgm⟩
```

`\__color_backend_fill_cmyk:n`
`\__color_backend_fill_gray:n`
`\__color_backend_fill_rgb:n`
`\__color_backend_fill:n`

Fill color here is the same as general color.

```
1035 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1036   { \__color_backend_fill:n { cmyk ~ #1 } }
1037 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1038   { \__color_backend_fill:n { gray ~ #1 } }
1039 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1040   { \__color_backend_fill:n { rgb ~ #1 } }
1041 \cs_new_protected:Npn \__color_backend_fill:n #1
1042   {
1043     \__kernel_backend_literal:n { color~push~ #1 }
1044   }
```

(*End of definition for* `\__color_backend_fill_cmyk:n` *and others.*)

`\__color_backend_stroke_cmyk:n`
`\__color_backend_stroke_gray:n`
`\__color_backend_stroke_gray_aux:n`
`\__color_backend_stroke_rgb:n`
`\__color_backend_stroke_rgb:w`
`\__color_backend:nnn`

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

```
1045 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1046   {
1047     \__color_backend_fill_cmyk:n {#1}
1048     \__kernel_backend_scope:n { stroke = "{?color}" }
1049     \__color_backend_reset:
1050   }
1051 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1052   {
1053     \use:e
1054       {
1055         \__color_backend_stroke_gray_aux:n
1056           { \fp_eval:n { 100 * (#1) } }
1057       }
1058   }
1059 \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1060   { \__color_backend:nnn {#1} {#1} {#1} }
1061 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1062   { \__color_backend_rgb:w #1 \s__color_stop }
1063 \cs_new_protected:Npn \__color_backend_stroke_rgb:w
1064   #1 ~ #2 ~ #3 \s__color_stop
1065   {
1066     \use:e
1067       {
1068         \__color_backend:nnn
1069           { \fp_eval:n { 100 * (#1) } }
1070           { \fp_eval:n { 100 * (#2) } }
1071           { \fp_eval:n { 100 * (#3) } }
1072       }
1073   }
1074 \cs_new_protected:Npe \__color_backend:nnn #1#2#3
1075   {
1076     \__kernel_backend_scope:n
```

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

(*End of definition for* __color_backend_stroke_cmyk:n *and others.*)

__color_backend_fill_separation:nn
__color_backend_stroke_separation:nn
__color_backend_fill_devicen:nn
__color_backend_stroke_devicen:nn

At present, these are no-ops.

```
1089 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1090 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2 { }
1091 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1092 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn
```

(*End of definition for* __color_backend_fill_separation:nn *and others.*)

__color_backend_fill_reset:
__color_backend_stroke_reset:

```
1093 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1094 \cs_new_protected:Npn \__color_backend_stroke_reset: { }
```

(*End of definition for* __color_backend_fill_reset: *and* __color_backend_stroke_reset:.)

__color_backend_devicen_init:nnn
__color_backend_iccbased_init:nnn

No support at present.

```
1095 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3 { }
1096 \cs_new_protected:Npn \__color_backend_iccbased_init:nnn #1#2#3 { }
```

(*End of definition for* __color_backend_devicen_init:nnn *and* __color_backend_iccbased_init:nnn.)

```
1097 ⟨/dvisvgm⟩
```

```
1098 ⟨/package⟩
```

3.5   Font handling integration

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

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

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

1159 ⟨/lua⟩

1160 ⟨∗luatex⟩

1161 ⟨∗package⟩

1162 \lua_load_module:n {l3backend-luatex}

1163 ⟨/package⟩

4 **l3backend-draw** implementation

1165 ⟨∗package⟩
1166 ⟨@@=draw⟩

4.1 **dvips** backend

1167 ⟨∗dvips⟩

__draw_backend_literal:n
__draw_backend_literal:e

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

1168 `\cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_postscript:n`
1169 `\cs_generate_variant:Nn \__draw_backend_literal:n { e }`

(*End of definition for* `\__draw_backend_literal:n`.)

__draw_backend_begin:
__draw_backend_end:

The `ps::[begin]` special here deals with positioning but allows us to continue on to a matching `ps::[end]`: contrast with `ps:`, which positions but where we can't split material between separate calls. The `@beginspecial`/`@endspecial` pair are from `special.pro` and correct the scale and *y*-axis direction. As for `pgf`, we need to save the current point as this is required for box placement. (Note that `@beginspecial`/`@endspecial` forms a backend scope.)

1170 `\cs_new_protected:Npn \__draw_backend_begin:`
1171 `  {`
1172 `    \__draw_backend_literal:n { [begin] }`
1173 `    \__draw_backend_literal:n { /draw.x~currentpoint~/draw.y~exch~def~def }`
1174 `    \__draw_backend_literal:n { @beginspecial }`
1175 `  }`
1176 `\cs_new_protected:Npn \__draw_backend_end:`
1177 `  {`
1178 `    \__draw_backend_literal:n { @endspecial }`
1179 `    \__draw_backend_literal:n { [end] }`
1180 `  }`

(*End of definition for* `\__draw_backend_begin:` *and* `\__draw_backend_end:`.)

__draw_backend_scope_begin:
__draw_backend_scope_end:

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

1181 `\cs_new_protected:Npn \__draw_backend_scope_begin:`
1182 `  { \__draw_backend_literal:n { save } }`
1183 `\cs_new_protected:Npn \__draw_backend_scope_end:`
1184 `  { \__draw_backend_literal:n { restore } }`

(*End of definition for* `\__draw_backend_scope_begin:` *and* `\__draw_backend_scope_end:`.)

__draw_backend_moveto:nn
__draw_backend_lineto:nn
__draw_backend_rectangle:nnnn
__draw_backend_curveto:nnnnnn

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

1185 `\cs_new_protected:Npn \__draw_backend_moveto:nn #1#2`
1186 `  {`
1187 `    \__draw_backend_literal:e`

32

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

(*End of definition for* __draw_backend_moveto:nn *and others.*)

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

__draw_backend_evenodd_rule:
__draw_backend_nonzero_rule:
\g__draw_draw_eor_bool

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

(*End of definition for* __draw_backend_evenodd_rule: *,* __draw_backend_nonzero_rule: *, and* \g__-
draw_draw_eor_bool*.*)

__draw_backend_closepath:
__draw_backend_stroke:
__draw_backend_closestroke:
__draw_backend_fill:
__draw_backend_fillstroke:
__draw_backend_clip:
__draw_backend_discardpath:
\g__draw_draw_clip_bool

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 stoke color is simple but the fill one has to be inserted by hand. For clipping, the required ordering is achieved using a TEX switch. All of the operations end with a new path instruction as they do not terminate (again in contrast to PDF).

```
1225 \cs_new_protected:Npn \__draw_backend_closepath:
1226    { \__draw_backend_literal:n { closepath } }
1227 \cs_new_protected:Npn \__draw_backend_stroke:
```

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

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

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

<div style="float:left">

__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:

</div>

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

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

```
1332    { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1333 \cs_new_protected:Npn \__draw_backend_join_round:
1334    { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1335 \cs_new_protected:Npn \__draw_backend_join_bevel:
1336    { \__draw_backend_literal:n { 2 ~ setlinejoin } }
```

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

`\__draw_backend_transform:nnnn`
`\__draw_backend_shift:nn`

In dvips, keeping the transformations in line with the engine is unfortunately not possible for scaling and rotations: even if we decompose the matrix into those operations, there is still no backend tracking (*cf.* dvipdfmx/X$_{\overline{E}}$T$_{E}$X). Thus we take the shortest path available and simply dump the matrix as given.

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

(*End of definition for* `\__draw_backend_transform:nnnn` *and* `\__draw_backend_shift:nn`.)

`\__draw_backend_box_use:Nnnnn`

Inside a picture `@beginspecial`/`@endspecial` are active, which is normally a good thing but means that the position and scaling would be off if the box was inserted directly. To deal with that, there are a number of possible approaches. A previous implementation suggested by Tom Rokici used `@endspecial`/`@beginspecial`. This avoids needing internals of dvips, but fails if there the box is used inside a scope (see https://github.com/latex3/latex3/issues/1504). Instead, we use the same method as pgf, which means tracking the position at the PostScript level. Also note that using `@endspecial` would close the scope it creates, meaning that after a box insertion, any local changes would be lost. Keeping dvips on track is non-trivial, hence the `[begin]`/`[end]` pair before the `save` and around the `restore`.

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

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

(*End of definition for* __draw_backend_box_use:Nnnnn.)

```
1370   ⟨/dvips⟩
```

4.2   LuaTeX, pdfTeX, dvipdfmx and XᴇTeX

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

```
1371   ⟨∗dvipdfmx | luatex | pdftex | xetex⟩
```

4.2.1   Drawing

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

(*End of definition for* __draw_backend_literal:n.)

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

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

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

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

__draw_backend_moveto:nn   Path creation operations all resolve directly to PDF primitive steps, with only the need
__draw_backend_lineto:nn   to convert to bp.
__draw_backend_curveto:nnnnnn
__draw_backend_rectangle:nnnn
```
1380   \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1381      {
1382        \__draw_backend_literal:e
1383          { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1384      }
1385   \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1386      {
1387        \__draw_backend_literal:e
1388          { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1389      }
1390   \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1391      {
1392        \__draw_backend_literal:e
1393          {
1394            \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
```

37

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

(*End of definition for* `\__draw_backend_moveto:nn` *and others.*)

`\__draw_backend_evenodd_rule:`
`\__draw_backend_nonzero_rule:`
`\g__draw_draw_eor_bool`

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

```
1409  \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1410      { \bool_gset_true:N \g__draw_draw_eor_bool }
1411  \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1412      { \bool_gset_false:N \g__draw_draw_eor_bool }
1413  \bool_new:N \g__draw_draw_eor_bool
```

(*End of definition for* `\__draw_backend_evenodd_rule:` *,* `\__draw_backend_nonzero_rule:` *, and* `\g__-draw_draw_eor_bool`*.*)

`\__draw_backend_closepath:`
`\__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.

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

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

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

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

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

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

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

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

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

```
1512 \cs_new_protected:Npn \__draw_backend_shift:nn #1#2
1513   {
1514     \__draw_backend_literal:n
1515       { 1 ~ 0 ~ 0 ~ 1 ~ #1 ~ #2 ~ cm }
1516   }
```

(*End of definition for* __draw_backend_transform:nnnn, __draw_backend_transform_aux:nnnn, *and* __draw_backend_shift:nn.)

__draw_backend_transform_decompose:nnnnN
__draw_backend_transform_decompose_auxi:nnnnN
__draw_backend_transform_decompose_auxii:nnnnN
__draw_backend_transform_decompose_auxiii:nnnnN

Internally, transformations for drawing are tracked as a matrix. Not all engines provide a way of dealing with this: if we use a raw matrix, the engine looses 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

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

at which point we just have to do various pieces of re-arrangement to get all of the values. (See J. Blinn, *IEEE Comput. Graph. Appl.*, 1996, **16**, 82–88.) There is one wrinkle: the PostScript (and PDF) way of specifying a transformation matrix exchanges where one would normally expect B and C to be.

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

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

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

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

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

(*End of definition for* __draw_backend_box_use:Nnnnn.)

```
1577 ⟨/dvipdfmx | luatex | pdftex | xetex⟩
```

4.3 `dvisvgm` backend

```
1578 ⟨*dvisvgm⟩
```

__draw_backend_literal:n   The same as the more general literal call.
__draw_backend_literal:e
```
1579 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1580 \cs_generate_variant:Nn \__draw_backend_literal:n { e }
```

(*End of definition for* __draw_backend_literal:n.)

__draw_backend_scope_begin: Use the backend-level scope mechanisms.
__draw_backend_scope_end:
```
1581 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1582 \cs_new_eq:NN \__draw_backend_scope_end: \__kernel_backend_scope_end:
```

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

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

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

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

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

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

43

```
1627            }
1628        }
1629    \tl_new:N \g__draw_backend_path_tl
```

(*End of definition for* __draw_backend_moveto:nn *and others.*)

__draw_backend_evenodd_rule: The fill rules here have to be handled as scopes.
__draw_backend_nonzero_rule:

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

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

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

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

44

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

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

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

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

45

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

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

`\__draw_backend_transform:nnnn`
`\__draw_backend_shift:nn`

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

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

(*End of definition for* `\__draw_backend_transform:nnnn` *and* `\__draw_backend_shift:nn`.)

`\__draw_backend_box_use:Nnnnn`

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

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

46

```
1762        \__kernel_backend_literal_svg:n
1763          {
1764            < g~
1765                stroke="none"~
1766                transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1767            >
1768          }
1769        \box_set_wd:Nn #1 { 0pt }
1770        \box_set_ht:Nn #1 { 0pt }
1771        \box_set_dp:Nn #1 { 0pt }
1772        \box_use:N #1
1773        \__kernel_backend_literal_svg:n { </g> }
1774        \__kernel_backend_scope_end:
1775      }
```

(*End of definition for* __draw_backend_box_use:Nnnnn.)

1776 ⟨/dvisvgm⟩

1777 ⟨/package⟩

5   l3backend-graphics implementation

1778 ⟨∗package⟩
1779 ⟨@@=graphics⟩

5.1   `dvips` backend

1780 ⟨∗dvips⟩

\l_graphics_search_ext_seq

```
1781 \seq_set_from_clist:Nn \l_graphics_search_ext_seq { .eps , .ps }
```

(*End of definition for* \l_graphics_search_ext_seq.)

__graphics_backend_getbb_eps:n
__graphics_backend_getbb_ps:n

Simply use the generic function.

```
1782 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1783 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
```

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

__graphics_backend_include_eps:n
__graphics_backend_include_ps:n

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

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

(*End of definition for* __graphics_backend_include_eps:n *and* __graphics_backend_include_ps:n.)

1796 `\cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n`

(*End of definition for* `\__graphics_backend_get_pagecount:n`.)

1797 ⟨/dvips⟩

5.2 LuaTeX and pdfTeX backends

1798 ⟨∗luatex | pdftex⟩

\l_graphics_search_ext_seq

1799 `\seq_set_from_clist:Nn \l_graphics_search_ext_seq`
1800   `{ .pdf , .eps , .ps , .png , .jpg , .jpeg }`

(*End of definition for* `\l_graphics_search_ext_seq`.)

\l__graphics_attr_tl   In PDF mode, additional attributes of an graphic (such as page number) are needed both to obtain the bounding box and when inserting the graphic: this occurs as the graphic dictionary approach means they are read as part of the bounding box operation. As such, it is easier to track additional attributes using a dedicated `tl` rather than build up the same data twice.

1801 `\tl_new:N \l__graphics_attr_tl`

(*End of definition for* `\l__graphics_attr_tl`.)

\l__graphics_transgroup_bool   Needed to indicate that a transparency group should be applied: only currently for PDF images, but could be extended.

1802 `\bool_new:N \l__graphics_transgroup_bool`

(*End of definition for* `\l__graphics_transgroup_bool`.)

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

1803 `\cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1`
1804   `{`
1805   `\int_zero:N \l__graphics_page_int`
1806   `\tl_clear:N \l__graphics_pagebox_tl`
1807   `\bool_set_false:N\l__graphics_transgroup_bool`
1808   `\tl_set:Ne \l__graphics_attr_tl`
1809   `{`
1810   `\tl_if_empty:NF \l__graphics_decodearray_str`
1811   `{ :D \l__graphics_decodearray_str }`
1812   `\bool_if:NT \l__graphics_interpolate_bool`
1813   `{ :I }`
1814   `\str_if_empty:NF \l__graphics_pdf_str`
1815   `{ :X \l__graphics_pdf_str }`
1816   `}`
1817   `\__graphics_backend_getbb_auxi:n {#1}`
1818   `}`
1819 `\cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n`
1820 `\cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n`

48

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

Measuring the graphic is done by boxing up: for PDF graphics we could use \tex_pdfximagebbox:D, but if doesn't work for other types. As the box always starts at $(0, 0)$ there is no need to worry about the lower-left position. Quotes need to be *removed* as LuaTeX does not like them here. We always apply a transparency group attribute here as included PDFs otherwise may have non-obvious behavior.

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

```
1869                }
1870              }
1871          \int_compare:nNnT \l__graphics_page_int > 0
1872            { page ~ \int_use:N \l__graphics_page_int }
1873          \tl_if_empty:NF \l__graphics_pagebox_tl
1874            { \l__graphics_pagebox_tl }
1875          {#1}
1876        \hbox_set:Nn \l__graphics_tmp_box
1877          { \tex_pdfrefximage:D \tex_pdflastximage:D }
1878        \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_tmp_box }
1879        \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_tmp_box }
1880      }
1881    \cs_new:Npn \__graphics_backend_dequote:w #1 " #2 " #3 \s__graphics_stop {#2}
```

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

`\__graphics_backend_include_jpg:n`
`\__graphics_backend_include_jpeg:n`
`\__graphics_backend_include_pdf:n`
`\__graphics_backend_include_png:n`

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

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

(*End of definition for* `\__graphics_backend_include_jpg:n` *and others.*)

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

EPS graphics may be included in LuaTeX/pdfTeX by conversion to PDF: this requires restricted shell escape. Modeled on the epstopdf LaTeX 2$_\varepsilon$ package, but simplified, conversion takes place here if we have shell access.

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

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

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

`\__graphics_backend_get_pagecount:n` Simply load and store.

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

(*End of definition for* `\__graphics_backend_get_pagecount:n.`)

```
1945 ⟨/luatex | pdftex⟩
```

5.3 dvipdfmx backend

```
1946 ⟨*dvipdfmx | xetex⟩
```

`\l_graphics_search_ext_seq`

```
1947 \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1948   { .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }
```

(*End of definition for* `\l_graphics_search_ext_seq.`)

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`

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

(*End of definition for* `\_\_graphics_backend_getbb_eps:n` *and others.*)

`\l__graphics_transgroup_bool` Needed to indicate that a transparency group should be applied: only currently for PDF images, but could be extended.

```
1968 \bool_new:N \l__graphics_transgroup_bool
```

(*End of definition for* `\l__graphics_transgroup_bool`.)

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

```
1969 \int_new:N \g__graphics_track_int
```

(*End of definition for* `\g__graphics_track_int`.)

`\_\_graphics_backend_include_eps:n`
`\_\_graphics_backend_include_ps:n`
`\_\_graphics_backend_include_jpg:n`
`\_\_graphics_backend_include_jpseg:n`
`\_\_graphics_backend_include_pdf:n`
`\_\_graphics_backend_include_png:n`
`\_\_graphics_backend_include_bmp:n`
`\_\_graphics_backend_include_auxi:n`
`\_\_graphics_backend_include_auxii:nn`
`\_\_graphics_backend_include_auxii:en`
`\_\_graphics_backend_include_auxiii:nn`

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

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

Graphic inclusion is set up to use the fact that each image is stored in the PDF as an XObject. This means that we can include repeated images only once and refer to them. To allow that, track the nature of each image: much the same as for the direct PDF mode case.

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

Inclusion using the specials is relatively straight-forward, but there is one wrinkle. To get
the pagebox correct for PDF graphics in all cases, it is necessary to provide both that
information and the bbox argument: odd things happen otherwise! We use the dvipdfmx
special in all cases as it allows attributes to be added to the XObject.

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

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

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

```
2058 ⟨*dvipdfmx⟩
2059 \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n
2060 ⟨/dvipdfmx⟩
```

(*End of definition for* `\__graphics_backend_get_pagecount:n`*.*)

```
2061 ⟨/dvipdfmx | xetex⟩
```

5.4   X꜂TEX backend

```
2062 ⟨*xetex⟩
```

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

```
2063 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2064   {
2065     \int_zero:N \l__graphics_page_int
2066     \tl_clear:N \l__graphics_pagebox_tl
2067     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2068   }
2069 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2070 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
```

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

(*End of definition for* __graphics_backend_getbb_jpg:n *and others.*)

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

```
2120 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
```

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

(*End of definition for* __graphics_backend_get_pagecount:n.)

```
2129  ⟨/xetex⟩
```

5.5 dvisvgm backend

```
2130  ⟨∗dvisvgm⟩
```

\l_graphics_search_ext_seq

```
2131  \seq_set_from_clist:Nn \l_graphics_search_ext_seq
2132    { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }
```

(*End of definition for* \l_graphics_search_ext_seq.)

__graphics_backend_getbb_svg:n
__graphics_backend_getbb_svg_auxi:nNn
__graphics_backend_getbb_svg_auxii:w
__graphics_backend_getbb_svg_auxiii:Nw
__graphics_backend_getbb_svg_auxiv:Nw
__graphics_backend_getbb_svg_auxv:Nw
__graphics_backend_getbb_svg_auxvi:Nn
__graphics_backend_getbb_svg_auxvii:w

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

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

56

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

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

Simply use the generic function.

```
2212 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2213 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
```

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

These can be included by extracting the bounding box data.

```
2214 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2215   {
2216     \int_zero:N \l__graphics_page_int
2217     \tl_clear:N \l__graphics_pagebox_tl
2218     \__graphics_extract_bb:n {#1}
2219   }
2220 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2221 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
```

(*End of definition for* __graphics_backend_getbb_png:n, __graphics_backend_getbb_jpg:n, *and* _- _graphics_backend_getbb_jpeg:n.)

Same as for `dvipdfmx`: use the generic function

```
2222 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2223   {
2224     \tl_clear:N \l__graphics_decodearray_str
2225     \bool_set_false:N \l__graphics_interpolate_bool
2226     \__graphics_extract_bb:n {#1}
2227   }
```

(*End of definition for* __graphics_backend_getbb_pdf:n.)

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

```
2228 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2229   { \__graphics_backend_include:nn { PSfile } {#1} }
2230 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
2231 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2232   { \__graphics_backend_include:nn { pdffile } {#1} }
2233 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
2234   {
2235     \__kernel_backend_literal:e
2236       {
2237         #1 = #2 \c_space_tl
2238         llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2239         lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2240         urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2241         ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2242       }
2243   }
```

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

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

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

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

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

`\__graphics_backend_get_pagecount:n`

```
2265 \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n
```

(*End of definition for* `\__graphics_backend_get_pagecount:n`.)

```
2266 ⟨/dvisvgm⟩
2267 ⟨/package⟩
```

6 **l3backend-pdf** implementation

```
2268 ⟨∗package⟩
2269 ⟨@@=pdf⟩
```

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

6.1 **dvips backend**

```
2270 ⟨∗dvips⟩
```

`\__pdf_backend_pdfmark:n`
`\__pdf_backend_pdfmark:e`

Used often enough it should be a separate function.

```
2271 \cs_new_protected:Npn \__pdf_backend_pdfmark:n #1
2272   { \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2273 \cs_generate_variant:Nn \__pdf_backend_pdfmark:n { e }
```

(*End of definition for* `\__pdf_backend_pdfmark:n`.)

6.1.1 Catalogue entries

__pdf_backend_catalog_gput:nn
__pdf_backend_info_gput:nn

```
2274 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2275   { \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2276 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2277   { \__pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }
```

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

6.1.2 Objects

__pdf_backend_object_new:
__pdf_backend_object_ref:n
__pdf_backend_object_id:n

```
2278 \cs_new_protected:Npn \__pdf_backend_object_new:
2279   { \int_gincr:N \g__pdf_backend_object_int }
2280 \cs_new:Npn \__pdf_backend_object_ref:n #1 { { pdf.obj #1 } }
2281 \cs_new_eq:NN \__pdf_backend_object_id:n \__pdf_backend_object_ref:n
```

(*End of definition for* __pdf_backend_object_new: , __pdf_backend_object_ref:n , *and* __pdf_-
backend_object_id:n.)

__pdf_backend_object_write:nnn
__pdf_backend_object_write:nne
__pdf_backend_object_write_aux:nnn
__pdf_backend_object_write_array:nn
__pdf_backend_object_write_dict:nn
__pdf_backend_object_write_fstream:nn
__pdf_backend_object_write_stream:nn
__pdf_backend_object_write_stream:nnn

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.

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

60

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

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

__pdf_backend_object_now:nn   No anonymous objects, so things are done manually.
__pdf_backend_object_now:ne
```
2344  \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2345    {
2346      \int_gincr:N \g__pdf_backend_object_int
2347      \__pdf_backend_object_write_aux:nnn
2348        { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2349        {#1} {#2}
2350    }
2351  \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }
```

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

__pdf_backend_object_last: Much like the annotation version.
```
2352  \cs_new:Npn \__pdf_backend_object_last:
2353    { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
```

(*End of definition for* __pdf_backend_object_last:.)

__pdf_backend_pageobject_ref:n   Page references are easy in dvips.
```
2354  \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2355    { { Page #1 } }
```

(*End of definition for* __pdf_backend_pageobject_ref:n.)

6.1.3 Destinations

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

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

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

(*End of definition for* __pdf_backend_destination:nn, __pdf_backend_destination:nnnn, *and* __-
pdf_backend_destination_aux:nnnn.)

6.1.4 Structure

__pdf_backend_compresslevel:n
__pdf_backend_compress_objects:n

Doable for the usual **ps2pdf** method.

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

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

__pdf_backend_version_major_gset:n
__pdf_backend_version_minor_gset:n

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

63

(*End of definition for* `\__pdf_backend_version_major_gset:n` *and* `\__pdf_backend_version_minor_-gset:n`.)

`\__pdf_backend_version_major:`
`\__pdf_backend_version_minor:`

Data not available!

```
2447 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2448 \cs_new:Npn \__pdf_backend_version_minor: { -1 }
```

(*End of definition for* `\__pdf_backend_version_major:` *and* `\__pdf_backend_version_minor:`.)

6.1.5 Marked content

`\__pdf_backend_bdc:nn`
`\__pdf_backend_emc:`

Simple wrappers.

```
2449 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2450   { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2451 \cs_new_protected:Npn \__pdf_backend_emc:
2452   { \__pdf_backend_pdfmark:n { /EMC } }
```

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

```
2453 ⟨/dvips⟩
```

6.2 LuaTeX and pdfTeX backend

```
2454 ⟨*luatex | pdftex⟩
```

6.2.1 Destinations

`\__pdf_backend_destination:nn`
`\__pdf_backend_destination:nnnn`

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.

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

64

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

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

6.2.2   Catalogue entries

<div style="float:left">_pdf_backend_catalog_gput:nn
__pdf_backend_info_gput:nn</div>

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

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

6.2.3   Objects

\g__pdf_backend_object_prop   For tracking objects to allow finalization.

```
2512 \prop_new:N \g__pdf_backend_object_prop
```

(*End of definition for* \g__pdf_backend_object_prop.)

<div style="float:left">__pdf_backend_object_new:
__pdf_backend_object_ref:n
__pdf_backend_object_id:n</div>   Declaring objects means reserving at the PDF level plus starting tracking.

```
2513 \cs_new_protected:Npn \__pdf_backend_object_new:
2514   {
2515 ⟨*luatex⟩
2516     \tex_pdfextension:D obj ~
2517 ⟨/luatex⟩
```

```
2518 ⟨*pdftex⟩
2519     \tex_pdfobj:D
2520 ⟨/pdftex⟩
2521       reserveobjnum ~
2522     \int_gset:Nn \g__pdf_backend_object_int
2523 ⟨*luatex⟩
2524       { \tex_pdffeedback:D lastobj }
2525 ⟨/luatex⟩
2526 ⟨*pdftex⟩
2527       { \tex_pdflastobj:D }
2528 ⟨/pdftex⟩
2529     }
2530 \cs_new:Npn \__pdf_backend_object_ref:n #1 { #1 ~ 0 ~ R }
2531 \cs_new:Npn \__pdf_backend_object_id:n #1 {#1}
```

(*End of definition for* __pdf_backend_object_new: , __pdf_backend_object_ref:n , *and* __pdf_-
backend_object_id:n.)

__pdf_backend_object_write:nnn
__pdf_backend_object_write:nne
__pdf_backend_object_write:nn
__pdf_exp_not_i:nn
__pdf_exp_not_ii:nn

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

```
2532 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2533   {
2534 ⟨*luatex⟩
2535     \tex_immediate:D \tex_pdfextension:D obj ~
2536 ⟨/luatex⟩
2537 ⟨*pdftex⟩
2538     \tex_immediate:D \tex_pdfobj:D
2539 ⟨/pdftex⟩
2540       useobjnum ~ #1
2541     \__pdf_backend_object_write:nn {#2} {#3}
2542   }
2543 \cs_new:Npn \__pdf_backend_object_write:nn #1#2
2544   {
2545     \str_case:nn {#1}
2546       {
2547         { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2548         { dict }  { { << ~ \exp_not:n {#2} ~ >> } }
2549         { fstream }
2550           {
2551             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2552               file ~ { \__pdf_exp_not_ii:nn #2 }
2553           }
2554         { stream }
2555           {
2556             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2557               { \__pdf_exp_not_ii:nn #2 }
2558           }
2559       }
2560   }
2561 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2562 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2563 \cs_new:Npn \__pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }
```

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

__pdf_backend_object_now:nn
__pdf_backend_object_now:ne

Much like writing, but direct creation.

```
2564    \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2565      {
2566  ⟨*luatex⟩
2567        \tex_immediate:D \tex_pdfextension:D obj ~
2568  ⟨/luatex⟩
2569  ⟨*pdftex⟩
2570        \tex_immediate:D \tex_pdfobj:D
2571  ⟨/pdftex⟩
2572          \__pdf_backend_object_write:nn {#1} {#2}
2573      }
2574    \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }
```

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

__pdf_backend_object_last: Much like annotation.

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

(*End of definition for* __pdf_backend_object_last:.)

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

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

(*End of definition for* __pdf_backend_pageobject_ref:n.)

6.2.4   Structure

__pdf_backend_compresslevel:n
__pdf_backend_compress_objects:n
__pdf_backend_objcompresslevel:n

Simply pass data to the engine.

```
2597    \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2598      {
2599        \tex_global:D
2600  ⟨*luatex⟩
2601        \tex_pdfvariable:D compresslevel
2602  ⟨/luatex⟩
2603  ⟨*pdftex⟩
2604        \tex_pdfcompresslevel:D
```

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

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

<div style="float:left">_pdf_backend_version_major_gset:n
_pdf_backend_version_minor_gset:n</div>

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

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

(*End of definition for* `\__pdf_backend_version_major_gset:n` *and* `\__pdf_backend_version_minor_-`
`gset:n` *.*)

`\__pdf_backend_version_major:`
`\__pdf_backend_version_minor:`

As above.

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

(*End of definition for* `\__pdf_backend_version_major:` *and* `\__pdf_backend_version_minor:`.)

6.2.5 Marked content

`\__pdf_backend_bdc:nn`
`\__pdf_backend_emc:`

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

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

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

```
2680 ⟨/luatex | pdftex⟩
```

6.3 dvipdfmx backend

```
2681 ⟨*dvipdfmx | xetex⟩
```

`\__pdf_backend:n`
`\__pdf_backend:e`

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

```
2682 \cs_new_protected:Npe \__pdf_backend:n #1
2683   { \__kernel_backend_literal:n { pdf: #1 } }
2684 \cs_generate_variant:Nn \__pdf_backend:n { e }
```

(*End of definition for* `\__pdf_backend:n`.)

6.3.1 Catalogue entries

```
2685 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2686   { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2687 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2688   { \__pdf_backend:n { docinfo << /#1 ~ #2 >> } }
```

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

6.3.2 Objects

For tracking objects to allow finalization.

```
2689 \prop_new:N \g__pdf_backend_object_prop
```

(*End of definition for* \g__pdf_backend_object_prop*.*)

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

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

(*End of definition for* __pdf_backend_object_new: *,* __pdf_backend_object_ref:n *, and* __pdf_-
backend_object_id:n*.*)

This is where we choose the actual type.

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

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

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

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

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

`\__pdf_backend_object_last:`

```
2730 \cs_new:Npn \__pdf_backend_object_last:
2731   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
```

(*End of definition for* `\__pdf_backend_object_last:`.)

`\__pdf_backend_pageobject_ref:n`
Page references are easy in `dvipdfmx`/X$_{\overline{H}}$T$_E$X.

```
2732 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2733   { @page #1 }
```

(*End of definition for* `\__pdf_backend_pageobject_ref:n`.)

6.3.3 Destinations

`\__pdf_backend_destination:nn`
`\__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$_E$X by using the backend data for `@xpos` and `@ypos`. `/FitR` without rule spec doesn't work, so it falls back to `/Fit` here.

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

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

(*End of definition for* __pdf_backend_destination:nn *,* __pdf_backend_destination:nnnn *, and* __-
pdf_backend_destination_aux:nnnn*.*)

6.3.4 Structure

__pdf_backend_compresslevel:n
__pdf_backend_compress_objects:n

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

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

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

`\_pdf_backend_version_major_gset:n`
`\_pdf_backend_version_minor_gset:n`

We start with the assumption that the default is active.

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

(*End of definition for* `\__pdf_backend_version_major_gset:n` *and* `\__pdf_backend_version_minor_-gset:n`.)

`\_pdf_backend_version_major:`
`\_pdf_backend_version_minor:`

We start with the assumption that the default is active.

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

(*End of definition for* `\__pdf_backend_version_major:` *and* `\__pdf_backend_version_minor:`.)

6.3.5 Marked content

`\_pdf_backend_bdc:nn`
`\_pdf_backend_emc:`

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

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

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

```
2817 ⟨/dvipdfmx ∣ xetex⟩
```

6.4 `dvisvgm` backend

```
2818 ⟨∗dvisvgm⟩
```

6.4.1 Destinations

`\_pdf_backend_destination:nn`
`\_pdf_backend_destination:nnnn`

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

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

6.4.2 Catalogue entries

`\_pdf_backend_catalog_gput:nn`
`\__pdf_backend_info_gput:nn`

No-op.

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

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

6.4.3 Objects

All no-ops here.

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

__pdf_backend_object_new:
__pdf_backend_object_ref:n
__pdf_backend_object_id:n
__pdf_backend_object_write:nnn
__pdf_backend_object_write:ne
__pdf_backend_object_now:nn
__pdf_backend_object_now:ne
__pdf_backend_object_last:
__pdf_backend_pageobject_ref:n

(*End of definition for* __pdf_backend_object_new: *and others.*)

6.4.4 Structure

These are all no-ops.

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

__pdf_backend_compresslevel:n
__pdf_backend_compress_objects:n

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

Data not available!

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

__pdf_backend_version_major_gset:n
__pdf_backend_version_minor_gset:n

(*End of definition for* __pdf_backend_version_major_gset:n *and* __pdf_backend_version_minor_-gset:n.)

Data not available!

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

__pdf_backend_version_major:
__pdf_backend_version_minor:

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

More no-ops.

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

__pdf_backend_bdc:nn
__pdf_backend_emc:

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

```
2840 ⟨/dvisvgm⟩
```

6.5 PDF Page size (media box)

For setting the media box, the split between backends is somewhat different to other areas, thus we approach this separately. The code here assumes a recent LaTeX 2$_\varepsilon$: that is ensured at the level above.

```
2841 ⟨*dvipdfmx | dvips⟩
```

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

```
2842 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2
2843   {
2844     \__kernel_backend_first_shipout:n
2845       {
2846         \__kernel_backend_literal:e
2847           {
2848 ⟨*dvipdfmx⟩
2849             pdf:pagesize ~
2850               width  ~ \dim_eval:n {#1} ~
2851               height ~ \dim_eval:n {#2}
2852 ⟨/dvipdfmx⟩
2853 ⟨*dvips⟩
2854             papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
2855 ⟨/dvips⟩
2856           }
2857       }
2858   }
```

(*End of definition for* __pdf_backend_pagesize_gset:nn.)

```
2859 ⟨/dvipdfmx | dvips⟩
2860 ⟨*luatex | pdftex | xetex⟩
```

__pdf_backend_pagesize_gset:nn   Pass to the primitives.

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

(*End of definition for* __pdf_backend_pagesize_gset:nn.)

```
2866 ⟨/luatex | pdftex | xetex⟩
2867 ⟨*dvisvgm⟩
```

__pdf_backend_pagesize_gset:nn   A no-op.

```
2868 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2 { }
```

(*End of definition for* __pdf_backend_pagesize_gset:nn.)

```
2869 ⟨/dvisvgm⟩
2870 ⟨/package⟩
```

7 **l3backend-pdfannot** implementation

```
2871 ⟨*package⟩
2872 ⟨@@=pdfannot⟩
```

7.1 **dvips backend**

```
2873 ⟨*dvips⟩
```

In dvips, annotations have to be constructed manually. As such, we need the object code above for some definitions. Here, the PostScript uses the pdf namespace: unlike for

expl3, we do not really control the namespacing and also have to cut across PDF-related areas.

`\l__pdfannot_backend_content_box` The content of an annotation.

```
2874 \box_new:N \l__pdfannot_backend_content_box
```

(*End of definition for* `\l__pdfannot_backend_content_box`.)

`\l__pdfannot_backend_model_box` For creating model sizing for links.

```
2875 \box_new:N \l__pdfannot_backend_model_box
```

(*End of definition for* `\l__pdfannot_backend_model_box`.)

`\g__pdfannot_backend_int` Needed to track annotations.

```
2876 \int_new:N \g__pdfannot_backend_int
```

(*End of definition for* `\g__pdfannot_backend_int`.)

`\__pdfannot_backend_generic:nnnn`
`\__pdfannot_backend_generic_aux:nnnn`
Annotations are objects but they are not in the object data lists. Here, to get the coordinates of the annotation, we need to have the data collected at the PostScript level. That requires a bit of box trickery (effectively a LaTeX 2ε `picture` of zero size). Once the data is collected, use it to set up the annotation border.

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

(*End of definition for* `\__pdfannot_backend_generic:nnnn` *and* `\__pdfannot_backend_generic_aux:nnnn`.)

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

```
2906 \cs_new:Npn \__pdfannot_backend_last:
2907   { { pdf.annot \int_use:N \g__pdfannot_backend_int } }
```

(*End of definition for* __pdfannot_backend_last:.)

\g__pdfannot_backend_link_int	To track annotations which are links.

```
2908 \int_new:N \g__pdfannot_backend_link_int
```

(*End of definition for* \g__pdfannot_backend_link_int.)

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

```
2909 \tl_new:N \g__pdfannot_backend_link_dict_tl
```

(*End of definition for* \g__pdfannot_backend_link_dict_tl.)

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

```
2910 \int_new:N \g__pdfannot_backend_link_sf_int
```

(*End of definition for* \g__pdfannot_backend_link_sf_int.)

\g__pdfannot_backend_link_math_bool	Needed to save/restore math mode.

```
2911 \bool_new:N \g__pdfannot_backend_link_math_bool
```

(*End of definition for* \g__pdfannot_backend_link_math_bool.)

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

```
2912 \bool_new:N \g__pdfannot_backend_link_bool
```

(*End of definition for* \g__pdfannot_backend_link_bool.)

\l__pdfannot_backend_breaklink_pdfmark_tl	Swappable content for link breaking.

```
2913 \tl_new:N \l__pdfannot_backend_breaklink_pdfmark_tl
2914 \tl_set:Nn \l__pdfannot_backend_breaklink_pdfmark_tl { pdfmark }
```

(*End of definition for* \l__pdfannot_backend_breaklink_pdfmark_tl.)

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

```
2915 \cs_new_protected:Npn \__pdfannot_backend_breaklink_postscript:n #1 { }
```

(*End of definition for* __pdfannot_backend_breaklink_postscript:n.)

__pdfannot_backend_breaklink_usebox:N	Swappable box unpacking or use.

```
2916 \cs_new_eq:NN \__pdfannot_backend_breaklink_usebox:N \box_use:N
```

(*End of definition for* __pdfannot_backend_breaklink_usebox:N.)

Links are created like annotations but with dedicated code to allow for adjusting the size of the rectangle. In contrast to hyperref, we grab the link content as a box which can then unbox: this allows the same interface as for pdfTeX.

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

Taking the idea of evenboxes from hypdvips, we implement a minimum box height and depth for link placement. This means that "underlining" with a hyperlink will generally give an even appearance. However, to ensure that the full content is always above the link border, we do not allow this to be negative (contrast hypdvips approach). The result should be similar to pdfTeX in the vast majority of foreseeable cases.

The object number for a link is saved separately from the rest of the dictionary as this allows us to insert it just once, at either an unbroken link or only in the first line of a broken one. That makes the code clearer but also avoids a low-level PostScript error with the code as taken from hypdvips.

Getting the outer dimensions of the text area may be better using a two-pass approach and `\tex_savepos:D`. That plus generic mode are still to re-examine.

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

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

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

```
2950    {
2951        \bool_if:NT \g__pdfannot_backend_link_math_bool
2952          { \c_math_toggle_token }
2953        \__pdfannot_backend_link_sf_save:
2954      \hbox_set_end:
2955      \__pdfannot_backend_link_minima:
2956      \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
2957      \exp_args:Ne \__pdfannot_backend_link_outerbox:n
2958        {
2959          \int_if_odd:nTF { \value { page } }
2960            { \oddsidemargin }
2961            { \evensidemargin }
2962        }
2963      \box_move_down:nn { \box_dp:N \l__pdfannot_backend_content_box }
2964        { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2965      \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.begin }
2966      \__pdfannot_backend_breaklink_usebox:N \l__pdfannot_backend_content_box
2967      \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.end }
2968      \box_move_up:nn { \box_ht:N \l__pdfannot_backend_content_box }
2969        {
2970          \hbox:n
2971            { \__kernel_backend_postscript:n { pdf.save.linkur } }
2972        }
2973      \int_gincr:N \g__pdfannot_backend_int
2974      \int_gset_eq:NN \g__pdfannot_backend_link_int \g__pdfannot_backend_int
2975      \__kernel_backend_postscript:e
2976        {
2977          mark
2978          /_objdef { pdf.annot \int_use:N \g__pdfannot_backend_link_int }
2979          \g__pdfannot_backend_link_dict_tl \c_space_tl
2980          pdf.rect
2981          /ANN ~ \l__pdfannot_backend_breaklink_pdfmark_tl
2982        }
2983      \__pdfannot_backend_link_sf_restore:
2984      \bool_gset_false:N \g__pdfannot_backend_link_bool
2985    }
2986  \cs_new_protected:Npn \__pdfannot_backend_link_minima:
2987    {
2988      \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
2989      \__kernel_backend_postscript:e
2990        {
2991          /pdf.linkdp.pad ~
2992            \dim_to_decimal:n
2993              {
2994                \dim_max:nn
2995                  {
2996                      \box_dp:N \l__pdfannot_backend_model_box
2997                    - \box_dp:N \l__pdfannot_backend_content_box
2998                  }
2999                  { 0pt }
3000              } ~
3001              pdf.pt.dvi ~ def
3002          /pdf.linkht.pad ~
3003            \dim_to_decimal:n
```

79

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

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

Hooks to allow link breaking: something will be needed in format mode at some stage. At present this code is disabled, pending a decision to activate.

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

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

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

```
3077 \cs_new:Npn \__pdfannot_backend_link_last:
3078   { { pdf.annot \int_use:N \g__pdfannot_backend_link_int } }
```

(*End of definition for* __pdfannot_backend_link_last:.)

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

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

(*End of definition for* __pdfannot_backend_link_margin:n.)

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

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

```
3088 ⟨/dvips⟩
```

7.2 LuaTₑX and pdfTₑX backend

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

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

(*End of definition for* __pdfannot_backend_generic:nnnn.)

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

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

(*End of definition for* __pdfannot_backend_last:.)

__pdfannot_backend_link_begin_goto:nnw
__pdfannot_backend_link_begin_user:nnw
__pdfannot_backend_link_begin:nnnw
__pdfannot_backend_link_end: Links are all created using the same internals.

```
3114 \cs_new_protected:Npn \__pdfannot_backend_link_begin_goto:nnw #1#2
3115   { \__pdfannot_backend_link_begin:nnnw {#1} { goto~name } {#2} }
3116 \cs_new_protected:Npn \__pdfannot_backend_link_begin_user:nnw #1#2
3117   { \__pdfannot_backend_link_begin:nnnw {#1} { user } {#2} }
3118 \cs_new_protected:Npn \__pdfannot_backend_link_begin:nnnw #1#2#3
3119   {
3120 ⟨∗luatex⟩
3121     \tex_pdfextension:D startlink ~
3122 ⟨/luatex⟩
3123 ⟨∗pdftex⟩
3124     \tex_pdfstartlink:D
3125 ⟨/pdftex⟩
3126     attr {#1}
3127     #2 {#3}
3128   }
3129 \cs_new_protected:Npn \__pdfannot_backend_link_end:
```

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

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

__pdfannot_backend_link_last: Formatted for direct use.

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

(*End of definition for* __pdfannot_backend_link_last:.)

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

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

(*End of definition for* __pdfannot_backend_link_margin:n.)

__pdfannot_backend_link_on: Separate definitions for the two engines.
__pdfannot_backend_link_off:
```
3159 \cs_new_protected:Npn \__pdfannot_backend_link_on:
3160 ⟨*luatex⟩
3161    { \tex_pdfextension:D linkstate 0 ~ }
3162 ⟨/luatex⟩
3163 ⟨*pdftex⟩
3164    { \tex_pdfrunninglinkon:D }
3165 ⟨/pdftex⟩
3166 \cs_new_protected:Npn \__pdfannot_backend_link_off:
3167 ⟨*luatex⟩
3168    { \tex_pdfextension:D linkstate 1 ~ }
3169 ⟨/luatex⟩
3170 ⟨*pdftex⟩
3171    { \tex_pdfrunninglinkoff:D }
3172 ⟨/pdftex⟩
```

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

3173 ⟨/luatex | pdftex⟩

7.3 dvipdfmx backend

3174 ⟨∗dvipdfmx | xetex⟩

__pdfannot_backend:n
__pdfannot_backend:e

A generic function for the backend PDF specials

```
3175 \cs_new_protected:Npe \__pdfannot_backend:n #1
3176   { \__kernel_backend_literal:n { pdf: #1 } }
3177 \cs_generate_variant:Nn \__pdfannot_backend:n { e }
```

(*End of definition for* __pdfannot_backend:n.)

\g__pdfannot_backend_int

Annotations are objects: but made with a separate tracker integer.

```
3178 \int_new:N \g__pdfannot_backend_int
```

(*End of definition for* \g__pdfannot_backend_int.)

__pdfannot_backend_generic:nnnn

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

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

(*End of definition for* __pdfannot_backend_generic:nnnn.)

__pdfannot_backend_last:

```
3191 \cs_new:Npn \__pdfannot_backend_last:
3192   { @pdfannot \int_use:N \g__pdfannot_backend_int }
```

(*End of definition for* __pdfannot_backend_last:.)

\g__pdfannot_backend_link_int

To track annotations which are links.

```
3193 \int_new:N \g__pdfannot_backend_link_int
```

(*End of definition for* \g__pdfannot_backend_link_int.)

__pdfannot_backend_link_begin_goto:nnw
__pdfannot_backend_link_begin_user:nnw
__pdfannot_backend_link_begin:n
__pdfannot_backend_link_end:

All created using the same internals.

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

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

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

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

```
3220 \cs_new:Npn \__pdfannot_backend_link_last:
3221   { @pdfannot \int_use:N \g__pdfannot_backend_link_int }
```

(*End of definition for* __pdfannot_backend_link_last:.)

__pdfannot_backend_link_margin:n Pass to dvipdfmx.

```
3222 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1
3223   { \__kernel_backend_literal:e { dvipdfmx:config~g~ \dim_eval:n {#1} } }
```

(*End of definition for* __pdfannot_backend_link_margin:n.)

__pdfannot_backend_link_on:
__pdfannot_backend_link_off:

```
3224 \cs_new_protected:Npn \__pdfannot_backend_link_on:  { \__pdfannot_backend:n { link } }
3225 \cs_new_protected:Npn \__pdfannot_backend_link_off: { \__pdfannot_backend:n { nolink } }
```

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

```
3226 ⟨/dvipdfmx | xetex⟩
```

7.4 dvisvgm backend

```
3227 ⟨*dvisvgm⟩
```

__pdfannot_backend_generic:nnnn

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

(*End of definition for* __pdfannot_backend_generic:nnnn.)

__pdfannot_backend_last:

```
3229 \cs_new:Npn \__pdfannot_backend_last: { }
```

(*End of definition for* __pdfannot_backend_last:.)

`\__pdfannot_backend_link_begin_goto:nnw`
`\__pdfannot_backend_link_begin_user:nnw`
`\__pdfannot_backend_link_begin:nnnw`
`\__pdfannot_backend_link_end:`

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

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

`\__pdfannot_backend_link_last:`

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

(*End of definition for* `\__pdfannot_backend_link_last:`.)

`\__pdfannot_backend_link_margin:n`

```
3235 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1 { }
```

(*End of definition for* `\__pdfannot_backend_link_margin:n`.)

`\__pdfannot_backend_link_on:`
`\__pdfannot_backend_link_off:`

For handling places like headers.

```
3236 \cs_new_protected:Npn \__pdfannot_backend_link_on:  { }
3237 \cs_new_protected:Npn \__pdfannot_backend_link_off: { }
```

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

```
3238 ⟨/dvisvgm⟩
```

7.5 Transitional code

This block is temporary: we have moved the backend functions here to a dedicated prefix. To facilitate that, we turn off DocStrip substitution and handle things manually.

```
3239 ⟨@@=⟩
```

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

```
3252 ⟨/package⟩
```

8 l3backend-opacity implementation

Although opacity is not color, it needs to be managed in a somewhat similar way: using a dedicated stack if possible. Depending on the backend, that may not be possible. There is also the need to cover fill/stroke setting as well as more general running opacity. It is easiest to describe the value used in terms of opacity, although commonly this is referred to as transparency.

3255 ⟨∗dvips⟩

`\__opacity_backend_select:n`
`\__opacity_backend_fill:n`
`\__opacity_backend_stroke:n`
`\__opacity_backend:nnn`
`\__opacity_backend_reset_fill:`
`\__opacity_backend_reset_stroke:`

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.

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

87

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

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

```
3312 ⟨/dvips⟩
```

```
3313 ⟨*dvipdfmx | luatex | pdftex | xetex⟩
```

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

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

(*End of definition for* \c__opacity_backend_stack_int.)

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

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

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

__opacity_backend_select:n   Much the same as color.
__opacity_backend_reset:

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

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

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

__opacity_backend_fill:n
__opacity_backend_stroke:n
_opacity_backend_fill_stroke:nn

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

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

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

(*End of definition for* __opacity_backend_fill:n*,* __opacity_backend_stroke:n*, and* __opacity_-
backend_fill_stroke:nn*.*)

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

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

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

```
3397 ⟨/dvipdfmx | luatex | pdftex | xetex⟩
```

```
3398 ⟨*dvisvgm⟩
```

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

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

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

```
3407 ⟨/dvisvgm⟩
```

```
3408 ⟨/package⟩
```

8.1 Font handling integration

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

```
3409 ⟨*lua⟩
```

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

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

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

9 l3backend-header implementation

3456 ⟨∗dvips & header⟩

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

```
3457 /color.sc { } def
```

(*End of definition for* color.sc.)

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

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

(*End of definition for* TeXcolorseparation *and* separation*.*)

pdf.globaldict

A small global dictionary for backend use.

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

(*End of definition for* pdf.globaldict*.*)

pdf.cvs
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.

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

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

pdf.linkmargin
pdf.linkdp.pad
pdf.linkht.pad

Settings which are defined up-front in SDict.

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

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

pdf.rect
pdf.save.ll
pdf.save.ur
pdf.save.linkll
pdf.save.linkur
pdf.llx
pdf.lly
pdf.urx
pdf.ury

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

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

92

```
3496      def
3497 /pdf.save.linkur
3498    {
3499      currentpoint
3500      pdf.linkmargin sub
3501      pdf.linkht.pad sub
3502      /pdf.ury exch def
3503      pdf.linkmargin add
3504      /pdf.urx exch def
3505    }
3506      def
```

(*End of definition for* pdf.rect *and others.*)

pdf.dest.anchor
pdf.dest.x
pdf.dest.y
pdf.dest.point
pdf.dest2device
pdf.dev.x
pdf.dev.y
pdf.tmpa
pdf.tmpb
pdf.tmpc
pdf.tmpd

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

```
3507 /pdf.dest.anchor
3508    {
3509      currentpoint exch
3510      pdf.dvi.pt 72 add
3511      /pdf.dest.x exch def
3512      pdf.dvi.pt
3513      vsize 72 sub exch sub
3514      /pdf.dest.y exch def
3515    }
3516      def
3517 /pdf.dest.point
3518    { pdf.dest.x pdf.dest.y } def
3519 /pdf.dest2device
3520    {
3521      /pdf.dest.y exch def
3522      /pdf.dest.x exch def
3523      matrix currentmatrix
3524      matrix defaultmatrix
3525      matrix invertmatrix
3526      matrix concatmatrix
3527      cvx exec
3528      /pdf.dev.y exch def
3529      /pdf.dev.x exch def
3530      /pdf.tmpd exch def
3531      /pdf.tmpc exch def
3532      /pdf.tmpb exch def
3533      /pdf.tmpa exch def
3534      pdf.dest.x pdf.tmpa mul
3535        pdf.dest.y pdf.tmpc mul add
3536        pdf.dev.x add
3537      pdf.dest.x pdf.tmpb mul
3538        pdf.dest.y pdf.tmpd mul add
3539        pdf.dev.y add
3540    }
3541      def
```

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

To know where a breakable link can go, we need to track the boundary rectangle. That can be done by hooking into `a` and `x` operations: those names have to be retained. The boundary is stored at the end of the operation. Special effort is needed at the start and end of pages (or rather galleys), such that everything works properly.

```
3542 /pdf.bordertracking false def
3543 /pdf.bordertracking.begin
3544   {
3545     SDict /pdf.bordertracking true put
3546     SDict /pdf.leftboundary undef
3547     SDict /pdf.rightboundary undef
3548     /a where
3549       {
3550         /a
3551           {
3552             currentpoint pop
3553             SDict /pdf.rightboundary known dup
3554               {
3555                 SDict /pdf.rightboundary get 2 index lt
3556                   { not }
3557                 if
3558               }
3559             if
3560               { pop }
3561               { SDict exch /pdf.rightboundary exch put }
3562             ifelse
3563             moveto
3564             currentpoint pop
3565             SDict /pdf.leftboundary known dup
3566               {
3567                 SDict /pdf.leftboundary get 2 index gt
3568                   { not }
3569                 if
3570               }
3571             if
3572               { pop }
3573               { SDict exch /pdf.leftboundary exch put }
3574             ifelse
3575           }
3576         put
3577       }
3578     if
3579   }
3580   def
3581 /pdf.bordertracking.end
3582   {
3583     /a where { /a { moveto } put } if
3584     /x where { /x { 0 exch rmoveto } put } if
3585     SDict /pdf.leftboundary known
3586       { pdf.outerbox 0 pdf.leftboundary put }
3587     if
3588     SDict /pdf.rightboundary known
3589       { pdf.outerbox 2 pdf.rightboundary put }
```

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

95

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

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

<div style="float:left">

pdf.breaklink
pdf.breaklink.write
pdf.count
pdf.currentrect

</div>

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.

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

```
{
    dup 4 array copy
    dup dup
      1 get
      pdf.outerbox pdf.rect.ht
      pdf.linkmargin 2 mul add sub
      3 exch put
    dup
      pdf.outerbox 2 get
      pdf.linkmargin add
      2 exch put
    dup dup
      3 get
      pdf.outerbox pdf.rect.ht
      pdf.linkmargin 2 mul add add
      1 exch put
    /pdf.currentrect exch def
    pdf.breaklink.write
      {
        pdf.currentrect
        dup
          pdf.outerbox 0 get
          pdf.linkmargin sub
          0 exch put
        dup
          pdf.outerbox 2 get
          pdf.linkmargin add
          2 exch put
        dup dup
          1 get
          pdf.baselineskip add
          1 exch put
        dup dup
          3 get
          pdf.baselineskip add
          3 exch put
        /pdf.currentrect exch def
        pdf.breaklink.write
      }
    1 index 3 get
    pdf.linkmargin 2 mul add
    pdf.outerbox pdf.rect.ht add
    2 index 1 get sub
    pdf.baselineskip div round cvi 1 sub
      exch
    repeat
    pdf.currentrect
    dup
      pdf.outerbox 0 get
      pdf.linkmargin sub
      0 exch put
    dup dup
      1 get
      pdf.baselineskip add
```

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

(*End of definition for* `pdf.breaklink` *and others.*)

pdf.pdfmark
pdf.pdfmark.good
pdf.outerbox
pdf.baselineskip
pdf.pdfmark.dict

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

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

98

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

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

3838 ⟨/dvips & header⟩

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

101

D

105

106

110