

File I

Implementation

1 l3backend-basics Implementation

```
1 <*package>
```

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

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

```
2 \ProvidesExplFile
3 <*dvipdfmx>
4   {l3backend-dvipdfmx.def}{2021-02-06}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2021-02-06}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2021-02-06}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2021-02-06}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2021-02-06}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2021-02-06}{ }
25   {L3 backend support: XeTeX}
26 </xetex>
```

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

```
27 \cs_if_exist:NTF \_kernel_dependency_version_check:n
28 {
29   \_kernel_dependency_version_check:n {2020-09-01}
30 <dvipdfmx>   {l3backend-dvipdfmx.def}
31 <dvips>      {l3backend-dvips.def}
32 <dvisvgm>    {l3backend-dvisvgm.def}
33 <luatex>     {l3backend-luatex.def}
34 <pdftex>     {l3backend-pdftex.def}
35 <xetex>      {l3backend-xetex.def}
```

```

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

```

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

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

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

```

\__kernel_backend_literal:n
\__kernel_backend_literal:x
46 \cs_new_eq:NN \__kernel_backend_literal:e \tex_special:D
47 \cs_new_protected:Npn \__kernel_backend_literal:n #1
48 { \__kernel_backend_literal:e { \exp_not:n {#1} } }
49 \cs_generate_variant:Nn \__kernel_backend_literal:n { x }

```

(End definition for `__kernel_backend_literal:e`.)

1.1 dvips backend

```

50 <dvips>

```

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

```

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

```

(End definition for `__kernel_backend_literal_postscript:n`.)

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

```

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

```

(End definition for `__kernel_backend_postscript:n`.)

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

```

57 \bool_if:NT \g__kernel_backend_header_bool
58 {

```

```

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

```

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

```

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

```

(End definition for `__kernel_backend_align_begin:` and `__kernel_backend_align_end:.`)

`__kernel_backend_scope_begin:` Saving/restoring scope for general operations needs to be done with **dvips** positioning (try without to see this!). Thus we need the `ps:` version of the special here. As only the graphics state is ever altered within this pairing, we use the lower-cost `g`-versions.

```

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

```

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

```

79 </dvips>

```

1.2 LuaTeX and pdfTeX backends

```

80 <*luatex | pdftex>

```

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

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

`__kernel_backend_literal_pdf:x`

```

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

```

```

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

```

(End definition for __kernel_backend_literal_pdf:n.)

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

```

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

```

(End definition for __kernel_backend_literal_page:n.)

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

```

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

```

(End definition for __kernel_backend_scope_begin: and __kernel_backend_scope_end:.)

__kernel_backend_matrix:n Here the appropriate function is set up to insert an affine matrix into the PDF. With pdfTeX and LuaTeX in direct PDF output mode there is a primitive for this, which only needs the rotation/scaling/skew part.

```

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

```

```

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

(End definition for \__kernel_backend_matrix:n.)

131 </luatex | pdftex>

```

1.3 dvipdfmx backend

```

132 <*dvipdfmx | xetex>

```

The `dvipdfmx` shares code with the PDF mode one (using the common section to this file) but also with `XYTeX`. The latter is close to identical to `dvipdfmx` and so all of the code here is extracted for both backends, with some `clean up` for `XYTeX` as required. Undocumented but equivalent to `pdfTeX`'s `literal` keyword. It's similar to be not the same as the documented `contents` keyword as that adds a `q/Q` pair.

```

\__kernel_backend_literal_pdf:n
\__kernel_backend_literal_pdf:x

```

```

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

```

(End definition for `__kernel_backend_literal_pdf:n`.)

```

\__kernel_backend_literal_page:n

```

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

```

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

```

(End definition for `__kernel_backend_literal_page:n`.)

```

\__kernel_backend_scope_begin:
\__kernel_backend_scope_end:

```

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

```

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

```

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

```

142 <@@=sys>

```

```

\c__kernel_sys_dvipdfmx_version_int

```

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

```

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

```

(End definition for `\c__kernel_sys_dvipdfmx_version_int`.)

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

1.4 dvisvgm backend

```
159 <*dvisvgm>
```

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

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

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

(End definition for `_kernel_backend_literal_svg:n`.)

```
\g__kernel_backend_scope_int
\l__kernel_backend_scope_int
```

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

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

(End definition for `\g__kernel_backend_scope_int` and `\l__kernel_backend_scope_int`.)

```
\_kernel_backend_scope_begin:
\_kernel_backend_scope_end:
\_kernel_backend_scope_begin:n
\_kernel_backend_scope_begin:x
\_kernel_backend_scope:n
\_kernel_backend_scope:x
```

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

```
165 \cs_new_protected:Npn \_kernel_backend_scope_begin:
166 {
167   \_kernel_backend_literal_svg:n { <g> }
168   \int_set_eq:NN
169     \l__kernel_backend_scope_int
170     \g__kernel_backend_scope_int
171   \group_begin:
172     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
173 }
174 \cs_new_protected:Npn \_kernel_backend_scope_end:
175 {
176   \prg_replicate:nn
177     { \g__kernel_backend_scope_int }
178     { \_kernel_backend_literal_svg:n { </g> } }
179   \group_end:
180   \int_gset_eq:NN
181     \g__kernel_backend_scope_int
182     \l__kernel_backend_scope_int
183 }
184 \cs_new_protected:Npn \_kernel_backend_scope_begin:n #1
185 {
186   \_kernel_backend_literal_svg:n { <g ~ #1 > }
187   \int_set_eq:NN
188     \l__kernel_backend_scope_int
```

```

189     \g__kernel_backend_scope_int
190     \group_begin:
191     \int_gset:Nn \g__kernel_backend_scope_int { 1 }
192   }
193   \cs_generate_variant:Nn \__kernel_backend_scope_begin:n { x }
194   \cs_new_protected:Npn \__kernel_backend_scope:n #1
195   {
196     \__kernel_backend_literal_svg:n { <g ~ #1 > }
197     \int_gincr:N \g__kernel_backend_scope_int
198   }
199   \cs_generate_variant:Nn \__kernel_backend_scope:n { x }

```

(End definition for __kernel_backend_scope_begin: and others.)

```

200 </dvisvgm>
201 </package>

```

2 l3backend-box Implementation

```

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

```

2.1 dvips backend

```

204 <*dvips>

```

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

```

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

```

(End definition for __box_backend_clip:N.)

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

```

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

```

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

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

```

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

```

(End definition for `__box_backend_scale:Nnn`.)

258 `</dvips>`

2.2 LuaTeX and pdfTeX backends

259 `<*luatex | pdftex>`

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

```

260 \cs_new_protected:Npn \__box_backend_clip:N #1

```



```

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

```

(End definition for _box_backend_clip:N.)

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

```

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

```

(End definition for _box_backend_rotate:Nn and others.)

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

```

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

```

(End definition for __box_backend_scale:Nnn.)

```

315 </luatex | pdftex>

```

2.3 dvipdfmx/X_YTeX backend

```

316 <*dvipdfmx | xetex>

```

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

```

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

```

(End definition for __box_backend_clip:N.)

__box_backend_rotate:Nn Rotating in dvipdfmx/X_YTeX can be implemented using either PDF or backend-specific code. The former approach however is not “aware” of the content of boxes: this means that any embedded links would not be adjusted by the rotation. As such, the backend-native approach is preferred: the code therefore is similar (though not identical) to the dvips version (notice the rotation angle here is positive). As for dvips, zero rotation is written as 0 not -0.

```

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

```

```

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

```

(End definition for __box_backend_rotate:Nn and __box_backend_rotate_aux:Nn.)

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

```

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

```

(End definition for __box_backend_scale:Nnn.)

359 </dvipdfmx | xetex>

2.4 dvisvgm backend

360 <*dvisvgm>

__box_backend_clip:N 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!

\g__box_clip_path_int

```

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

```

```

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

```

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

```

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

```

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

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

```

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

```

```

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

```

(End definition for `__box_backend_rotate:Nn`.)

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

```

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

```

(End definition for `__box_backend_scale:Nnn`.)

```

443 </dvisvgm>
444 </package>

```

3 l3backend-color Implementation

```

445 <*package>
446 <@@=color>

```

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

3.1 Collecting information from L^AT_EX 2_ε

3.1.1 dvips-style

```

447 <*dvisvgm | dvipdfmx | dvips | xetex>

```

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

```

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

```

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

```

469 </dvipsgm | dvipdfmx | dvips | xetex>

```

3.1.2 Lua_T_EX and pdf_T_EX

```

470 < *luatex | pdftex>

```

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

```

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

```

```

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

```

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

```
503 </luatex | pdftex>
```

3.2 The color stack

For PDF-based engines, we have a color stack available inside the specials. This is used for concepts beyond color itself: it is needed to manage the graphics state generally. The exact form depends on the engine, and for `dvipdfmx/XYTeX` the backend version.

3.2.1 Common code

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

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

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

(End definition for `\l__color_backend_stack_int`.)

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

3.2.2 dvipdfmx/X_YTeX

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

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

```

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

```

```

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

```

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

_color_backend_stack_push:nn

_color_backend_stack_push:nx

_color_backend_stack_pop:n

Simple enough but needs a version check.

```

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

```

(End definition for _color_backend_stack_push:nn and _color_backend_stack_pop:n.)

564 </dvipdfmx | xetex>

3.2.3 LuaTeX and pdfTeX

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

$\backslash_kernel_color_backend_stack_init:Nnn$

```

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

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

$\backslash_color_backend_stack_push:nn$

$\backslash_color_backend_stack_push:nx$

$\backslash_color_backend_stack_pop:n$

```

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

(End definition for $\backslash_color_backend_stack_push:nn$ and $\backslash_color_backend_stack_pop:n$.)

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

3.3 General color

3.3.1 dvips-style

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

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
    color.sc
603 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
604 { \__color_backend_select:n { cmyk ~ #1 } }
605 \cs_new_protected:Npn \__color_backend_select_gray:n #1
606 { \__color_backend_select:n { gray ~ #1 } }
607 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
608 { \__color_backend_select:n { rgb ~ #1 } }
609 \cs_new_protected:Npn \__color_backend_select:n #1
610 {
611     \__kernel_backend_literal:n { color~push~ #1 }
612 }
613 \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
614 }
615 \group_insert_after:N \__color_backend_reset:
616 }
617 \cs_new_protected:Npn \__color_backend_reset:
618 { \__kernel_backend_literal:n { color~pop } }

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

619 }

```

3.3.2 LuaTeX and pdfTeX

```

620 }
\l__color_backend_fill_tl
\l__color_backend_stroke_tl
621 \tl_new:N \l__color_backend_fill_tl
622 \tl_new:N \l__color_backend_stroke_tl

(End definition for \l__color_backend_fill_tl and \l__color_backend_stroke_tl.)

\__color_backend_select_cmyk:n
\__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.
623 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
624 { \__color_backend_select:nn { #1 ~ k } { #1 ~ K } }
625 \cs_new_protected:Npn \__color_backend_select_gray:n #1
626 { \__color_backend_select:nn { #1 ~ g } { #1 ~ G } }
627 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
628 { \__color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
629 \cs_new_protected:Npn \__color_backend_select:nn #1#2
630 {
631     \tl_set:Nn \l__color_backend_fill_tl {#1}
632     \tl_set:Nn \l__color_backend_stroke_tl {#2}
633     \__color_backend_stack_push:nn \l__color_backend_stack_int { #1 ~ #2 }
634     \group_insert_after:N \__color_backend_reset:
635 }
636 \cs_new_protected:Npn \__color_backend_reset:
637 { \__color_backend_stack_pop:n \l__color_backend_stack_int }

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

638 }

```

3.3.3 dvipdfmx/X_YTeX

639 \langle *dvipdfmx | xetex \rangle

These backends have the most possible approaches: it recognises both dvips-based color specials and it's own format, plus one can include PDF statements directly. Recent releases also have a color stack approach similar to pdfTeX. Of the stack methods, the dedicated the most versatile is the latter as it can cover all of the use cases we have. Thus it is used in preference to the dvips-style interface or the “native” color specials (which have only one stack).

Push the data to the stack.

```

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

```

(End definition for __color_backend_select_cmyk:n and others.)

652 \langle /dvipdfmx | xetex \rangle

3.4 Separations

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

653 \langle *dvips \rangle

```

654 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
655 { \__color_backend_select:n { separation ~ #1 ~ #2 } }
656 \cs_new_eq:NN \__color_backend_select_device:nn \__color_backend_select_separation:nn

```

(End definition for __color_backend_select_separation:nn and __color_backend_select_device:nn.)

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

```

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

```

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

```

805     {#3}
806   }

```

(End definition for `_color_backend_separation_init:nnnnn` and others.)

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

```

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

```

(End definition for `_color_backend_devicen_init:nnn`.)

```

825 </dvips>
826 <*dvisvgm>

```

`_color_backend_select_separation:nn` No support at present.

```

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

```

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

`_color_backend_separation_init:nnnnn` No support at present.

```

\_color_backend_separation_init_CIELAB:nnn
829 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #1#2#3#4#5 { }
830 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnnnn #1#2#3 { }

```

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

```

831 </dvisvgm>
832 <*dvipdfmx | luatex | pdftex | xetex>

```

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

```

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

```

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

_color_backend_separation_init:nnnnn
 _color_backend_separation_init:n
 _color_backend_separation_init_CIELAB:nnn

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

```

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

```

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

```

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

```



```

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

```

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

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

```

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

```

```

932 \cs_if_exist:NF \pdf_object_now:nn
933 { \cs_gset_protected:Npn \__color_backend_devicen_init:nnn #1#2#3 { } }
934 \cs_new:Npn \__color_backend_devicen_init:w #1 ~ #2 \s__color_stop
935 {
936   + 1
937   \tl_if_blank:NF {#2}
938   { \__color_backend_devicen_init:w #2 \s__color_stop }
939 }
940 \cs_new_eq:NN \__color_backend_devicen_init:n \__color_backend_separation_init:n
(End definition for \__color_backend_devicen_init:nnn, \__color_backend_devicen_init:w, and \_
_color_backend_devicen_init:n.)
941 </dvipdfmx | luatex | pdftex | xetex>
942 <*dvipdfmx | xetex>

```

```

\__color_backend_select_separation:nn
\__color_backend_select_devicen:nn

```

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

```

943 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
944 {
945   \cs_gset_protected:Npn \__color_backend_select_separation:nn #1#2 { }
946   \cs_gset_eq:NN \__color_backend_select_devicen:nn
947   \__color_backend_select_separation:nn
948 }
(End definition for \__color_backend_select_separation:nn and \__color_backend_select_devicen:nn.)
949 </dvipdfmx | xetex>

```

3.5 Fill and stroke color

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

```

950 <*dvipdfmx | luatex | pdftex | xetex>

```

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

```

\__color_backend_fill_cmyk:n
\__color_backend_fill_gray:n
\__color_backend_fill_rgb:n
\__color_backend_fill:n
\__color_backend_stroke_cmyk:n
\__color_backend_stroke_gray:n
\__color_backend_stroke_rgb:n
\__color_backend_stroke:n
951 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
952 { \__color_backend_fill:n { #1 ~ k } }
953 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
954 { \__color_backend_fill:n { #1 ~ g } }
955 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
956 { \__color_backend_fill:n { #1 ~ rg } }
957 \cs_new_protected:Npn \__color_backend_fill:n #1
958 {
959   \tl_set:Nn \l__color_backend_fill_tl {#1}
960   \__color_backend_stack_push:nn \l__color_backend_stack_int
961   { #1 ~ \l__color_backend_stroke_tl }
962   \group_insert_after:N \__color_backend_reset:
963 }

```

```

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

```

(End definition for __color_backend_fill_cmyk:n and others.)

```

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

```

(End definition for __color_backend_fill_separation:nn and others.)

```

983 </dvipdfmx | luatex | pdftex | xetex>
984 <*dvipdfmx | xetex>

```

Deal with older (x)dvipdfmx.

```

985 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
986 {
987   \cs_gset_protected:Npn \__color_backend_fill_cmyk:n #1
988   {
989     \__kernel_backend_literal:n { pdf: bc ~ [#1] }
990     \group_insert_after:N \__color_backend_reset:
991   }
992   \cs_gset_eq:NN \__color_backend_fill_gray:n \__color_backend_fill_cmyk:n
993   \cs_gset_eq:NN \__color_backend_fill_rgb:n \__color_backend_fill_cmyk:n
994   \cs_gset_protected:Npn \__color_backend_reset:
995   { \__kernel_backend_literal:n { pdf: ec } }
996   \cs_gset_protected:Npn \__color_backend_stroke:n #1
997   { \__kernel_backend_literal:n {#1} }
998   \cs_gset_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
999   \cs_gset_eq:NN \__color_backend_fill_devicen:nn
1000   \__color_backend_fill_separation:nn
1001   \cs_gset_eq:NN \__color_backend_stroke_separation:nn
1002   \__color_backend_fill_separation:nn
1003   \cs_gset_eq:NN \__color_backend_stroke_devicen:nn
1004   \__color_backend_stroke_separation:nn
1005 }

```

(End definition for __color_backend_fill_cmyk:n and others.)

```

1006 </dvipdfmx | xetex>

```

1007 <*dvips>

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

```

1008 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1009 { \__color_backend_fill:n { cmyk ~ #1 } }
1010 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1011 { \__color_backend_fill:n { gray ~ #1 } }
1012 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1013 { \__color_backend_fill:n { rgb ~ #1 } }
1014 \cs_new_protected:Npn \__color_backend_fill:n #1
1015 {
1016   \__kernel_backend_literal:n { color~push~ #1 }
1017   \group_insert_after:N \__color_backend_reset:
1018 }
1019 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1020 { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmycolor } 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 definition for __color_backend_fill_cmyk:n and others.)

```

\__color_backend_fill_separation:nn
\__color_backend_stroke_separation: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 definition for __color_backend_fill_separation:nn and others.)

1031 </dvips>

1032 <*dvisvgm>

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

```

1033 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1034 { \__color_backend_fill:n { cmyk ~ #1 } }
1035 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1036 { \__color_backend_fill:n { gray ~ #1 } }
1037 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1038 { \__color_backend_fill:n { rgb ~ #1 } }
1039 \cs_new_protected:Npn \__color_backend_fill:n #1
1040 {
1041   \__kernel_backend_literal:n { color~push~ #1 }
1042   \group_insert_after:N \__color_backend_reset:
1043 }

```

(End definition for __color_backend_fill_cmyk:n and others.)

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

```

1044 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1045 { \__color_backend_cmyk:w #1 \s__color_stop }

```

```

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

```

```

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

```

(End definition for __color_backend_stroke_cmyk:n and others.)

At present, these are no-ops.

```

\__color_backend_fill_separation:mn
\__color_backend_stroke_separation:mn
\__color_backend_fill_devicen:mn
\__color_backend_stroke_devicen:mn

```

```

1095 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2 { }

```

```

1096 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2 { }
1097 \cs_new_eq:NN \__color_backend_fill_device:n \__color_backend_fill_separation:nn
1098 \cs_new_eq:NN \__color_backend_stroke_device:n \__color_backend_stroke_separation:nn
(End definition for \__color_backend_fill_separation:nn and others.)
1099 </dvips>
1100 </package>

```

4 l3backend-draw Implementation

```

1101 <*package>
1102 <@@=draw>

```

4.1 dvips backend

```

1103 <*dvips>

```

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

`__draw_backend_literal:x`

```

1104 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_postscript:n
1105 \cs_generate_variant:Nn \__draw_backend_literal:n { x }
(End definition for \__draw_backend_literal:n.)

```

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

`__draw_backend_end:`

```

1106 \cs_new_protected:Npn \__draw_backend_begin:
1107 {
1108   \__kernel_backend_literal:n { ps::[begin] }
1109   \__draw_backend_literal:n { @beginspecial }
1110 }
1111 \cs_new_protected:Npn \__draw_backend_end:
1112 {
1113   \__draw_backend_literal:n { @endspecial }
1114   \__kernel_backend_literal:n { ps::[end] }
1115 }

```

(End definition for `__draw_backend_begin:` and `__draw_backend_end:.`)

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

`__draw_backend_scope_end:`

```

1116 \cs_new_protected:Npn \__draw_backend_scope_begin:
1117 { \__draw_backend_literal:n { save } }
1118 \cs_new_protected:Npn \__draw_backend_scope_end:
1119 { \__draw_backend_literal:n { restore } }

```

(End 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 x-type expansion is included here to ensure that any variable values are forced to literals before any possible caching. There is no native rectangular path command (without also clipping, filling or stroking), so that task is done using a small amount of PostScript.

```

1120 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1121 {
1122   \__draw_backend_literal:x
1123   {
1124     \dim_to_decimal_in_bp:n {#1} ~
1125     \dim_to_decimal_in_bp:n {#2} ~ moveto
1126   }
1127 }
1128 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1129 {
1130   \__draw_backend_literal:x
1131   {
1132     \dim_to_decimal_in_bp:n {#1} ~
1133     \dim_to_decimal_in_bp:n {#2} ~ lineto
1134   }
1135 }
1136 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1137 {
1138   \__draw_backend_literal:x
1139   {
1140     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1141     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1142     moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1143   }
1144 }
1145 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1146 {
1147   \__draw_backend_literal:x
1148   {
1149     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1150     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1151     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1152     curveto
1153   }
1154 }

```

(End definition for __draw_backend_moveto:nn and others.)

```

\_draw_backend_evenodd_rule:
\_draw_backend_nonzero_rule:
\g__draw_draw_eor_bool

```

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

```

1155 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1156 { \bool_gset_true:N \g__draw_draw_eor_bool }
1157 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1158 { \bool_gset_false:N \g__draw_draw_eor_bool }
1159 \bool_new:N \g__draw_draw_eor_bool

```

(End definition for __draw_backend_evenodd_rule:, __draw_backend_nonzero_rule:, and \g__draw_draw_eor_bool.)

```

\__draw_backend_closepath:
  \__draw_backend_stroke:
\__draw_backend_closestroke:
  \__draw_backend_fill:
  \__draw_backend_fillstroke:
  \__draw_backend_clip:
\__draw_backend_discardpath:
  \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 stroke color is simple but the fill one has to be inserted by hand. For clipping, the required ordering is achieved using a T_EX switch. All of the operations end with a new path instruction as they do not terminate (again in contrast to PDF).

```

1160 \cs_new_protected:Npn \__draw_backend_closepath:
1161 { \__draw_backend_literal:n { closepath } }
1162 \cs_new_protected:Npn \__draw_backend_stroke:
1163 {
1164   \__draw_backend_literal:n { gsave }
1165   \__draw_backend_literal:n { color.sc }
1166   \__draw_backend_literal:n { stroke }
1167   \__draw_backend_literal:n { grestore }
1168   \bool_if:NT \g__draw_draw_clip_bool
1169   {
1170     \__draw_backend_literal:x
1171     {
1172       \bool_if:NT \g__draw_draw_eor_bool { eo }
1173       clip
1174     }
1175   }
1176   \__draw_backend_literal:n { newpath }
1177   \bool_gset_false:N \g__draw_draw_clip_bool
1178 }
1179 \cs_new_protected:Npn \__draw_backend_closestroke:
1180 {
1181   \__draw_backend_closepath:
1182   \__draw_backend_stroke:
1183 }
1184 \cs_new_protected:Npn \__draw_backend_fill:
1185 {
1186   \__draw_backend_literal:x
1187   {
1188     \bool_if:NT \g__draw_draw_eor_bool { eo }
1189     fill
1190   }
1191   \bool_if:NT \g__draw_draw_clip_bool
1192   {
1193     \__draw_backend_literal:x
1194     {
1195       \bool_if:NT \g__draw_draw_eor_bool { eo }
1196       clip
1197     }
1198   }
1199   \__draw_backend_literal:n { newpath }
1200   \bool_gset_false:N \g__draw_draw_clip_bool
1201 }
1202 \cs_new_protected:Npn \__draw_backend_fillstroke:
1203 {
1204   \__draw_backend_literal:x
1205   {
1206     \bool_if:NT \g__draw_draw_eor_bool { eo }
1207     fill
1208   }
1209   \__draw_backend_literal:n { gsave }

```



```

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

```

(End definition for __draw_backend_closepath: and others.)

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1240 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1241 {
1242   \__draw_backend_literal:x
1243   {
1244     [
1245       \exp_args:Nf \use:n
1246       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1247     ] ~
1248     \dim_to_decimal_in_bp:n {#2} ~ setdash
1249   }
1250 }
1251 \cs_new:Npn \__draw_backend_dash:n #1
1252 { ~ \dim_to_decimal_in_bp:n {#1} }
1253 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1254 {
1255   \__draw_backend_literal:x
1256   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1257 }
1258 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1259 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }

```

```

1260 \cs_new_protected:Npn \__draw_backend_cap_but:
1261 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1262 \cs_new_protected:Npn \__draw_backend_cap_round:
1263 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1264 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1265 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1266 \cs_new_protected:Npn \__draw_backend_join_miter:
1267 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1268 \cs_new_protected:Npn \__draw_backend_join_round:
1269 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1270 \cs_new_protected:Npn \__draw_backend_join_bevel:
1271 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

(End definition for __draw_backend_dash_pattern:nn and others.)

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

```

1272 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1273 {
1274   \__draw_backend_literal:n
1275   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1276 }

```

(End definition for __draw_backend_cm:nnnn.)

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

```

1277 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1278 {
1279   \__draw_backend_literal:n { @endspecial }
1280   \__draw_backend_literal:n { [end] }
1281   \__draw_backend_literal:n { [begin] }
1282   \__draw_backend_literal:n { save }
1283   \__draw_backend_literal:n { currentpoint }
1284   \__draw_backend_literal:n { currentpoint~translate }
1285   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1286   \__draw_backend_cm:nnnn { #2 } { #3 } { #4 } { #5 }
1287   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1288   \__draw_backend_literal:n { neg~exch~neg~exch~translate }
1289   \__draw_backend_literal:n { [end] }
1290   \hbox_overlap_right:n { \box_use:N #1 }
1291   \__draw_backend_literal:n { [begin] }

```

```

1292     \_draw_backend_literal:n { restore }
1293     \_draw_backend_literal:n { [end] }
1294     \_draw_backend_literal:n { [begin] }
1295     \_draw_backend_literal:n { @beginspecial }
1296 }

```

(End definition for _draw_backend_box_use:Nnnnn.)

```

1297 </dvips>

```

4.2 LuaTeX, pdfTeX, dvipdfmx and XeTeX

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

```

1298 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

```

\_draw_backend_literal:n Pass data through using a dedicated interface.
\_draw_backend_literal:x
1299 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_pdf:n
1300 \cs_generate_variant:Nn \_draw_backend_literal:n { x }

```

(End definition for _draw_backend_literal:n.)

```

\_draw_backend_begin: No special requirements here, so simply set up a drawing scope.
  \_draw_backend_end:
1301 \cs_new_protected:Npn \_draw_backend_begin:
1302   { \_draw_backend_scope_begin: }
1303 \cs_new_protected:Npn \_draw_backend_end:
1304   { \_draw_backend_scope_end: }

```

(End definition for _draw_backend_begin: and _draw_backend_end:.)

```

\_draw_backend_scope_begin: Use the backend-level scope mechanisms.
\_draw_backend_scope_end:
1305 \cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:
1306 \cs_new_eq:NN \_draw_backend_scope_end: \_kernel_backend_scope_end:

```

(End definition for _draw_backend_scope_begin: and _draw_backend_scope_end:.)

```

\_draw_backend_moveto:nn Path creation operations all resolve directly to PDF primitive steps, with only the need
\_draw_backend_lineto:nn to convert to bp.
  \_draw_backend_curveto:nnnnnn
  \_draw_backend_rectangle:nnnn

```

```

1307 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1308   {
1309     \_draw_backend_literal:x
1310     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1311   }
1312 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1313   {
1314     \_draw_backend_literal:x
1315     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1316   }
1317 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1318   {
1319     \_draw_backend_literal:x
1320     {
1321       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~

```

```

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

```

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

```

\__draw_backend_evenodd_rule: The even-odd rule here can be implemented as a simply switch.
\__draw_backend_nonzero_rule:
\g__draw_draw_eor_bool
1336 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1337 { \bool_gset_true:N \g__draw_draw_eor_bool }
1338 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1339 { \bool_gset_false:N \g__draw_draw_eor_bool }
1340 \bool_new:N \g__draw_draw_eor_bool

```

(End definition for `__draw_backend_evenodd_rule:`, `__draw_backend_nonzero_rule:`, and `\g__draw_draw_eor_bool`.)

```

\__draw_backend_closepath: Converting paths to output is again a case of mapping directly to PDF operations.
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
1341 \cs_new_protected:Npn \__draw_backend_closepath:
1342 { \__draw_backend_literal:n { h } }
1343 \cs_new_protected:Npn \__draw_backend_stroke:
1344 { \__draw_backend_literal:n { S } }
1345 \cs_new_protected:Npn \__draw_backend_closestroke:
1346 { \__draw_backend_literal:n { s } }
1347 \cs_new_protected:Npn \__draw_backend_fill:
1348 {
1349     \__draw_backend_literal:x
1350     { f \bool_if:NT \g__draw_draw_eor_bool * }
1351 }
1352 \cs_new_protected:Npn \__draw_backend_fillstroke:
1353 {
1354     \__draw_backend_literal:x
1355     { B \bool_if:NT \g__draw_draw_eor_bool * }
1356 }
1357 \cs_new_protected:Npn \__draw_backend_clip:
1358 {
1359     \__draw_backend_literal:x
1360     { W \bool_if:NT \g__draw_draw_eor_bool * }
1361 }
1362 \cs_new_protected:Npn \__draw_backend_discardpath:
1363 { \__draw_backend_literal:n { n } }

```

(End definition for `__draw_backend_closepath:` and others.)

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:

```

```

1364 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1365 {
1366   \__draw_backend_literal:x
1367   {
1368     [
1369       \exp_args:Nf \use:n
1370       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1371     ] ~
1372     \dim_to_decimal_in_bp:n {#2} ~ d
1373   }
1374 }
1375 \cs_new:Npn \__draw_backend_dash:n #1
1376 { ~ \dim_to_decimal_in_bp:n {#1} }
1377 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1378 {
1379   \__draw_backend_literal:x
1380   { \dim_to_decimal_in_bp:n {#1} ~ w }
1381 }
1382 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1383 { \__draw_backend_literal:x { #1 ~ M } }
1384 \cs_new_protected:Npn \__draw_backend_cap_butt:
1385 { \__draw_backend_literal:n { 0 ~ J } }
1386 \cs_new_protected:Npn \__draw_backend_cap_round:
1387 { \__draw_backend_literal:n { 1 ~ J } }
1388 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1389 { \__draw_backend_literal:n { 2 ~ J } }
1390 \cs_new_protected:Npn \__draw_backend_join_miter:
1391 { \__draw_backend_literal:n { 0 ~ j } }
1392 \cs_new_protected:Npn \__draw_backend_join_round:
1393 { \__draw_backend_literal:n { 1 ~ j } }
1394 \cs_new_protected:Npn \__draw_backend_join_bevel:
1395 { \__draw_backend_literal:n { 2 ~ j } }

```

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

```

\__draw_backend_cm:nnnn
\__draw_backend_cm_aux:nnnn

```

Another split here between LuaTeX/pdfTeX and dvipdfmx/X_YTeX. In the former, we have a direct method to maintain alignment: the backend can use a matrix itself. For dvipdfmx/X_YTeX, we can to decompose the matrix into rotations and a scaling, then use those operations as they are handled by the backend. (There is backend support for matrix operations in dvipdfmx/X_YTeX, but as a matched pair so not suitable for the “stand alone” transformation set up here.) The specials used here are from xdvipdfmx originally: they are well-tested, but probably equivalent to the pdf₊ versions!

```

1396 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1397 {
1398   <*luatex | pdftex>
1399   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1400   </luatex | pdftex>
1401   <*dvipdfmx | xetex>
1402   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1403   \__draw_backend_cm_aux:nnnn
1404   </dvipdfmx | xetex>
1405   }
1406   <*dvipdfmx | xetex>

```

```

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

```

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

```

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

```

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

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

The parent matrix can be converted to

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

From these, we can find that

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

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

```

1431 <*dvipdfmx|xetex>
1432 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1433 {
1434   \use:x
1435   {
1436     \__draw_backend_cm_decompose_auxi:nnnnN
1437     { \fp_eval:n { (#1 + #4) / 2 } }
1438     { \fp_eval:n { (#1 - #4) / 2 } }
1439     { \fp_eval:n { (#3 + #2) / 2 } }
1440     { \fp_eval:n { (#3 - #2) / 2 } }
1441   }
1442   #5
1443 }
1444 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1445 {
1446   \use:x
1447   {
1448     \__draw_backend_cm_decompose_auxii:nnnnN
1449     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1450     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1451     { \fp_eval:n { atand ( #3 , #2 ) } }
1452     { \fp_eval:n { atand ( #4 , #1 ) } }
1453   }
1454   #5
1455 }
1456 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1457 {
1458   \use:x
1459   {
1460     \__draw_backend_cm_decompose_auxiii:nnnnN
1461     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1462     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1463     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1464     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1465   }
1466   #5
1467 }
1468 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1469 {
1470   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1471   { #5 {#1} {#2} {#3} {#4} }
1472   { #5 {#1} {#3} {#2} {#4} }
1473 }
1474 </dvipdfmx|xetex>

```

(End definition for __draw_backend_cm_decompose:nnnnN and others.)

__draw_backend_box_use:Nnnnn

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

```

1475 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1476 {

```

```

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

(End definition for \__draw_backend_box_use:Nnnnn.)

1491 </dvipdfmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1492 <*dvisvgm>

```

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

```

\__draw_backend_literal:x
1493 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1494 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for __draw_backend_literal:n.)

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

```

1495 \cs_new_protected:Npn \__draw_backend_begin:
1496 {
1497     \__kernel_backend_scope_begin:
1498     \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1499     }
1500 \cs_new_eq:NN \__draw_backend_end: \__kernel_backend_scope_end:

```

(End definition for __draw_backend_begin: and __draw_backend_end:.)

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

```

\__draw_backend_lineto:nn
\__draw_backend_rectangle:nnnn
\__draw_backend_curveto:nnnnnn
\__draw_backend_add_to_path:n
\g__draw_draw_path_tl
1501 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1502 {
1503     \__draw_backend_add_to_path:n
1504     { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1505     }
1506 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1507 {
1508     \__draw_backend_add_to_path:n
1509     { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1510     }

```



```

1511 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1512 {
1513   \__draw_backend_add_to_path:n
1514   {
1515     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1516     h ~ \dim_to_decimal:n {#3} ~
1517     v ~ \dim_to_decimal:n {#4} ~
1518     h ~ \dim_to_decimal:n { -#3 } ~
1519     Z
1520   }
1521 }
1522 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1523 {
1524   \__draw_backend_add_to_path:n
1525   {
1526     C ~
1527     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1528     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1529     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1530   }
1531 }
1532 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1533 {
1534   \tl_gset:Nx \g__draw_draw_path_tl
1535   {
1536     \g__draw_draw_path_tl
1537     \tl_if_empty:NF \g__draw_draw_path_tl { \c_space_tl }
1538     #1
1539   }
1540 }
1541 \tl_new:N \g__draw_draw_path_tl

```

(End definition for __draw_backend_moveto:nn and others.)

```

\__draw_backend_evenodd_rule: The fill rules here have to be handled as scopes.
\__draw_backend_nonzero_rule:
1542 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1543 { \__draw_backend_scope:n { fill-rule="evenodd" } }
1544 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1545 { \__draw_backend_scope:n { fill-rule="nonzero" } }

```

(End definition for __draw_backend_evenodd_rule: and __draw_backend_nonzero_rule:.)

```

\__draw_backend_path:n
\__draw_backend_closepath:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
1546 \cs_new_protected:Npn \__draw_backend_closepath:
1547 { \__draw_backend_add_to_path:n { Z } }
\__draw_backend_discardpath:
1548 \cs_new_protected:Npn \__draw_backend_path:n #1
1549 {
1550   \bool_if:NTF \g__draw_draw_clip_bool
1551   {
1552     \int_gincr:N \g__draw_clip_path_int

```

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

```

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

```

```

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

```

(End definition for _draw_backend_path:n and others.)

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_dash_aux:nn
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_but:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1614 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1615 {
1616     \use:x
1617     {
1618         \__draw_backend_dash_aux:nn
1619         { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1620         { \dim_to_decimal:n {#2} }
1621     }
1622 }
1623 \cs_new:Npn \__draw_backend_dash:n #1
1624 { , \dim_to_decimal_in_bp:n {#1} }
1625 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1626 {
1627     \__draw_backend_scope:x
1628     {
1629         stroke-dasharray =
1630         "
1631         \tl_if_empty:oTF { \use_none:n #1 }
1632         { none }
1633         { \use_none:n #1 }
1634         " ~
1635         stroke-offset=" #2 "
1636     }
1637 }
1638 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1639 { \__draw_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1640 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1641 { \__draw_backend_scope:x { stroke-miterlimit=" #1 " } }
1642 \cs_new_protected:Npn \__draw_backend_cap_but:
1643 { \__draw_backend_scope:n { stroke-linecap="butt" } }
1644 \cs_new_protected:Npn \__draw_backend_cap_round:
1645 { \__draw_backend_scope:n { stroke-linecap="round" } }
1646 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1647 { \__draw_backend_scope:n { stroke-linecap="square" } }
1648 \cs_new_protected:Npn \__draw_backend_join_miter:
1649 { \__draw_backend_scope:n { stroke-linejoin="miter" } }
1650 \cs_new_protected:Npn \__draw_backend_join_round:
1651 { \__draw_backend_scope:n { stroke-linejoin="round" } }
1652 \cs_new_protected:Npn \__draw_backend_join_bevel:
1653 { \__draw_backend_scope:n { stroke-linejoin="bevel" } }

```

(End definition for _draw_backend_dash_pattern:nn and others.)

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

```

1654 \cs_new_protected:Npn \_draw_backend_cm:nnnn #1#2#3#4
1655 {
1656   \_draw_backend_scope:n
1657   {
1658     transform =
1659     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1660   }
1661 }

```

(End definition for `_draw_backend_cm:nnnn`.)

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

```

1662 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5#6#7
1663 {
1664   \_kernel_backend_scope_begin:
1665   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1666   \_kernel_backend_literal_svg:n
1667   {
1668     < g~
1669     stroke="none"~
1670     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1671     >
1672   }
1673   \box_set_wd:Nn #1 { Opt }
1674   \box_set_ht:Nn #1 { Opt }
1675   \box_set_dp:Nn #1 { Opt }
1676   \box_use:N #1
1677   \_kernel_backend_literal_svg:n { </g> }
1678   \_kernel_backend_scope_end:
1679 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```

1680 </dvisvgm>
1681 </package>

```

5 l3backend-graphics Implementation

```

1682 <*package>
1683 <@@=graphics>

```

5.1 dvips backend

```

1684 <*dvips>

```

`_graphics_backend_getbb_eps:n` Simply use the generic function.

```

1685 \cs_new_eq:NN \_graphics_backend_getbb_eps:n \graphics_read_bb:n

```

(End definition for `_graphics_backend_getbb_eps:n`.)

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

```

1686 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1687 {
1688   \_kernel_backend_literal:x
1689   {
1690     PSfile = #1 \c_space_tl
1691     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1692     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1693     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1694     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1695   }
1696 }

```

(End definition for `_graphics_backend_include_eps:n`.)

```

1697 </dvips>

```

5.2 LuaTeX and pdfTeX backends

```

1698 <*luatex | pdftex>

```

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

```

1699 \tl_new:N \l_graphics_graphics_attr_tl

```

(End definition for `\l_graphics_graphics_attr_tl`.)

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

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

```

1700 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1701 {
1702   \int_zero:N \l_graphics_page_int
1703   \tl_clear:N \l_graphics_pagebox_tl
1704   \tl_set:Nx \l_graphics_graphics_attr_tl
1705   {
1706     \tl_if_empty:NF \l_graphics_decodearray_tl
1707     { :D \l_graphics_decodearray_tl }
1708     \bool_if:NT \l_graphics_interpolate_bool
1709     { :I }
1710   }
1711   \tl_clear:N \l_graphics_graphics_attr_tl
1712   \_graphics_backend_getbb_auxi:n {#1}
1713 }
1714 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1715 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1716 {
1717   \tl_clear:N \l_graphics_decodearray_tl
1718   \bool_set_false:N \l_graphics_interpolate_bool

```

```

1719 \tl_set:Nx \l__graphics_graphics_attr_tl
1720 {
1721   : \l_graphics_pagebox_tl
1722   \int_compare:nNnT \l_graphics_page_int > 1
1723   { :P \int_use:N \l_graphics_page_int }
1724 }
1725 \__graphics_backend_getbb_auxi:n {#1}
1726 }
1727 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1728 {
1729   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1730   { \__graphics_backend_getbb_auxii:n {#1} }
1731 }

```

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.

```

1732 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1733 {
1734   \tex_immediate:D \tex_pdfximage:D
1735   \bool_lazy_or:nnT
1736   { \l_graphics_interpolate_bool }
1737   { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1738   {
1739     attr ~
1740     {
1741       \tl_if_empty:NF \l_graphics_decodearray_tl
1742       { /Decode~[ \l_graphics_decodearray_tl ] }
1743       \bool_if:NT \l_graphics_interpolate_bool
1744       { /Interpolate~true }
1745     }
1746   }
1747   \int_compare:nNnT \l_graphics_page_int > 0
1748   { page ~ \int_use:N \l_graphics_page_int }
1749   \tl_if_empty:NF \l_graphics_pagebox_tl
1750   { \l_graphics_pagebox_tl }
1751   {#1}
1752   \hbox_set:Nn \l__graphics_internal_box
1753   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1754   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1755   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1756   \int_const:cn { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1757   { \tex_the:D \tex_pdflastximage:D }
1758   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1759 }

```

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

```

\_graphics_backend_include_jpg:n
\_graphics_backend_include_pdf:n
\_graphics_backend_include_png:n

```

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

```

1760 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1761 {
1762   \tex_pdfrefximage:D

```

```

1763     \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1764   }
1765   \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1766   \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

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

```

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

```

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

1767 \sys_if_shell:T
1768 {
1769   \str_new:N \l__graphics_backend_dir_str
1770   \str_new:N \l__graphics_backend_name_str
1771   \str_new:N \l__graphics_backend_ext_str
1772   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1773   {
1774     \file_parse_full_name:nNNN {#1}
1775     \l__graphics_backend_dir_str
1776     \l__graphics_backend_name_str
1777     \l__graphics_backend_ext_str
1778     \exp_args:Nx \__graphics_backend_getbb_eps:nn
1779     {
1780       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1781       -converted-to.pdf
1782     }
1783     {#1}
1784   }
1785   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1786   {
1787     \file_compare_timestamp:nNnT {#2} > {#1}
1788     {
1789       \sys_shell_now:n
1790       { repstopdf ~ #2 ~ #1 }
1791     }
1792     \tl_set:Nn \l__graphics_name_tl {#1}
1793     \__graphics_backend_getbb_pdf:n {#1}
1794   }
1795   \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1796   {
1797     \file_parse_full_name:nNNN {#1}
1798     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1799     \exp_args:Nx \__graphics_backend_include_pdf:n
1800     {
1801       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1802       -converted-to.pdf
1803     }
1804   }
1805 }

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

1806 </luatex | pdftex>

```

5.3 dvipdfmx backend

1807 `<*dvipdfmx | xetex>`

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

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

```
1808 \cs_new_eq:NN \_graphics_backend_getbb_eps:n \graphics_read_bb:n
1809 <*dvipdfmx>
1810 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1811 {
1812   \int_zero:N \l_graphics_page_int
1813   \tl_clear:N \l_graphics_pagebox_tl
1814   \graphics_extract_bb:n {#1}
1815 }
1816 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1817 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1818 {
1819   \tl_clear:N \l_graphics_decodearray_tl
1820   \bool_set_false:N \l_graphics_interpolate_bool
1821   \graphics_extract_bb:n {#1}
1822 }
1823 </dvipdfmx>
```

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

`\g__graphics_track_int`

Used to track the object number associated with each graphic.

```
1824 \int_new:N \g__graphics_track_int
```

(End definition for `\g__graphics_track_int`.)

`_graphics_backend_include_eps:n`
`_graphics_backend_include_jpg:n`
`_graphics_backend_include_pdf:n`
`_graphics_backend_include_png:n`
`_graphics_backend_include_auxi:nn`
`_graphics_backend_include_auxii:nnn`
`_graphics_backend_include_auxiii:nnn`

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

```
1825 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1826 {
1827   \__kernel_backend_literal:x
1828   {
1829     PSfile = #1 \c_space_tl
1830     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1831     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1832     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1833     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1834   }
1835 }
1836 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1837 { \_graphics_backend_include_auxi:nn {#1} { image } }
1838 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1839 <*dvipdfmx>
1840 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1841 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1842 </dvipdfmx>
```

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


```

1843 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1844 {
1845   \__graphics_backend_include_auxii:xnn
1846   {
1847     \tl_if_empty:NF \l_graphics_pagebox_tl
1848     { : \l_graphics_pagebox_tl }
1849     \int_compare:nNnT \l_graphics_page_int > 1
1850     { :P \int_use:N \l_graphics_page_int }
1851     \tl_if_empty:NF \l_graphics_decodearray_tl
1852     { :D \l_graphics_decodearray_tl }
1853     \bool_if:NT \l_graphics_interpolate_bool
1854     { :I }
1855   }
1856   {#1} {#2}
1857 }
1858 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1859 {
1860   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1861   {
1862     \__kernel_backend_literal:x
1863     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1864   }
1865   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1866 }
1867 \cs_generate_variant:Nn \__graphics_backend_include_auxii:nnn { x }

```

Inclusion using the specials is relatively straight-forward, but there is one wrinkle. To get the `pagebox` correct for PDF graphics in all cases, it is necessary to provide both that information and the `bbox` argument: odd things happen otherwise!

```

1868 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1869 {
1870   \int_gincr:N \g__graphics_track_int
1871   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1872   \__kernel_backend_literal:x
1873   {
1874     pdf:#3~
1875     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1876     \int_compare:nNnT \l_graphics_page_int > 1
1877     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1878     \tl_if_empty:NF \l_graphics_pagebox_tl
1879     {
1880       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
1881       bbox ~
1882       \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1883       \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1884       \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1885       \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
1886     }
1887     (#1)
1888     \bool_lazy_or:nnT
1889     { \l_graphics_interpolate_bool }
1890     { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1891     {
1892       <<

```

```

1893         \tl_if_empty:NF \l_graphics_decodearray_tl
1894         { /Decode~[ \l_graphics_decodearray_tl ] }
1895         \bool_if:NT \l_graphics_interpolate_bool
1896         { /Interpolate~true> }
1897     >>
1898 }
1899 }
1900 }

```

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

```
1901 </dvipdfmx | xetex>
```

5.4 X_YTeX backend

```
1902 <*xetex>
```

5.4.1 Images

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

```

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

```

```

1903 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1904 {
1905     \int_zero:N \l_graphics_page_int
1906     \tl_clear:N \l_graphics_pagebox_tl
1907     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
1908 }
1909 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1910 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1911 {
1912     \tl_clear:N \l_graphics_decodearray_tl
1913     \bool_set_false:N \l_graphics_interpolate_bool
1914     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
1915 }
1916 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
1917 {
1918     \int_compare:nNnTF \l_graphics_page_int > 1
1919     { \__graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
1920     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
1921 }
1922 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
1923 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
1924 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
1925 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
1926 {
1927     \tl_if_empty:NTF \l_graphics_pagebox_tl
1928     { \__graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
1929     { \__graphics_backend_getbb_auxv:nNnn }
1930     {#1} #2 {#3} {#4}
1931 }
1932 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
1933 {
1934     \use:x

```

```

1935     {
1936       \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
1937       { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
1938     }
1939   }
1940   \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
1941   \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
1942   {
1943     \graphics_bb_restore:nF {#1#3}
1944     { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
1945   }
1946   \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
1947   {
1948     \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
1949     \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1950     \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1951     \graphics_bb_save:n {#1#3}
1952   }
1953   \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

__graphics_backend_include_pdf:n
__graphics_backend_include_bitmap_quote:w

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

```

1954   \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1955   {
1956     \tex_XeTeXpdf_file:D
1957     \__graphics_backend_include_pdf_quote:w #1 "#1" \s__graphics_stop \c_space_tl
1958     \int_compare:nNnT \l_graphics_page_int > 0
1959     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
1960     \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
1961   }
1962   \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \s__graphics_stop
1963   { " #2 " }

```

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

```

1964 </xetex>

```

5.5 dvisvgm backend

```

1965 <*dvisvgm>

```

__graphics_backend_getbb_eps:n Simply use the generic function.

```

1966   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \graphics_read_bb:n

```

(End definition for __graphics_backend_getbb_eps:n.)

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

```

\__graphics_backend_getbb_jpg:n
1967   \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1968   {
1969     \int_zero:N \l_graphics_page_int

```

```

1970     \tl_clear:N \l_graphics_pagebox_tl
1971     \graphics_extract_bb:n {#1}
1972   }
1973   \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
(End definition for \__graphics_backend_getbb_png:n and \__graphics_backend_getbb_jpg:n.)

```

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

```

1974   \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1975   {
1976     \tl_clear:N \l_graphics_decodearray_tl
1977     \bool_set_false:N \l_graphics_interpolate_bool
1978     \graphics_extract_bb:n {#1}
1979   }
(End definition for \__graphics_backend_getbb_pdf:n.)

```

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

```

\__graphics_backend_include_pdf:n
\__graphics_backend_include:nn
1980   \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1981   { __graphics_backend_include:nn { PSfile } {#1} }
1982   \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1983   { __graphics_backend_include:nn { pdffile } {#1} }
1984   \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
1985   {
1986     \__kernel_backend_literal:x
1987     {
1988       #1 = #2 \c_space_tl
1989       llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1990       lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1991       urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1992       ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1993     }
1994   }
(End definition for \__graphics_backend_include_eps:n, \__graphics_backend_include_pdf:n, and \__graphics_backend_include:nn.)

```

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

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

__graphics_backend_include_bitmap_quote:w

```

1995   \cs_new_protected:Npn \__graphics_backend_include_png:n #1
1996   {
1997     \__kernel_backend_literal:x
1998     {
1999       dvisvgm:img~
2000       \dim_to_decimal:n { \l_graphics_ury_dim } ~
2001       \dim_to_decimal:n { \l_graphics_ury_dim } ~
2002       \__graphics_backend_include_bitmap_quote:w #1 " #1 " \s__graphics_stop
2003     }
2004   }
2005   \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_png:n
2006   \cs_new:Npn \__graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \s__graphics_stop
2007   { " #2 " }

```

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

2008 `</dvisvgm>`

2009 `</package>`

6 l3backend-pdf Implementation

2010 `<*package>`

2011 `<@=pdf>`

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

6.1 Shared code

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

`\l__pdf_internal_box`

2012 `\box_new:N \l__pdf_internal_box`

(End definition for `\l__pdf_internal_box`.)

6.2 dvips backend

2013 `<*dvips>`

`__pdf_backend_pdfmark:n` Used often enough it should be a separate function.

`__pdf_backend_pdfmark:x`

2014 `\cs_new_protected:Npn __pdf_backend_pdfmark:n #1`

2015 `{ __kernel_backend_postscript:n { mark #1 ~ pdfmark } }`

2016 `\cs_generate_variant:Nn __pdf_backend_pdfmark:n { x }`

(End definition for `__pdf_backend_pdfmark:n`.)

6.2.1 Catalogue entries

`__pdf_backend_catalog_gput:nn`

`__pdf_backend_info_gput:nn`

2017 `\cs_new_protected:Npn __pdf_backend_catalog_gput:nn #1#2`

2018 `{ __pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }`

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

2020 `{ __pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }`

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

6.2.2 Objects

\g__pdf_backend_object_int
\g__pdf_backend_object_prop

For tracking objects to allow finalisation.

```
2021 \int_new:N \g__pdf_backend_object_int
2022 \prop_new:N \g__pdf_backend_object_prop
```

(End definition for \g__pdf_backend_object_int and \g__pdf_backend_object_prop.)

__pdf_backend_object_new:nn
__pdf_backend_object_ref:n

Tracking objects is similar to dvipdfmx.

```
2023 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2024 {
2025   \int_gincr:N \g__pdf_backend_object_int
2026   \int_const:cn
2027     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2028     { \g__pdf_backend_object_int }
2029   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2030 }
2031 \cs_new:Npn \__pdf_backend_object_ref:n #1
2032 { { pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } } }
```

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

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

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

```
2033 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2034 {
2035   \__pdf_backend_pdfmark:x
2036   {
2037     /_objdef ~ \__pdf_backend_object_ref:n {#1}
2038     /type
2039     \str_case_e:nn
2040       { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2041       {
2042         { array } { /array }
2043         { dict } { /dict }
2044         { fstream } { /stream }
2045         { stream } { /stream }
2046       }
2047     /OBJ
2048   }
2049   \use:c
2050     { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2051     { \__pdf_backend_object_ref:n {#1} } {#2}
2052 }
2053 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2054 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2055 {
2056   \__pdf_backend_pdfmark:x
2057   { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2058 }
2059 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2060 {
2061   \__pdf_backend_pdfmark:x
2062   { #1 << \exp_not:n {#2} >> /PUT }
2063 }
2064 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
```

```

2065 {
2066   \exp_args:Nx
2067   \__pdf_backend_object_write_fstream:nnn {#1} #2
2068 }
2069 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2070 {
2071   \__kernel_backend_postscript:n
2072   {
2073     SDict ~ begin ~
2074     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2075     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2076     end
2077   }
2078 }
2079 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2080 {
2081   \exp_args:Nx
2082   \__pdf_backend_object_write_stream:nnn {#1} #2
2083 }
2084 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2085 {
2086   \__kernel_backend_postscript:n
2087   {
2088     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2089     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2090   }
2091 }

```

(End definition for __pdf_backend_object_write:nn and others.)

__pdf_backend_object_now:nn
__pdf_backend_object_now:nx

No anonymous objects, so things are done manually.

```

2092 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2093 {
2094   \int_gincr:N \g__pdf_backend_object_int
2095   \__pdf_backend_pdfmark:x
2096   {
2097     /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2098     /type
2099     \str_case:nn
2100     {#1}
2101     {
2102       { array } { /array }
2103       { dict } { /dict }
2104       { fstream } { /stream }
2105       { stream } { /stream }
2106     }
2107     /OBJ
2108   }
2109   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2110   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2111 }
2112 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for __pdf_backend_object_now:nn.)

`__pdf_backend_object_last:` Much like the annotation version.

```

2113 \cs_new:Npn \__pdf_backend_object_last:
2114   { { pdf.obj \int_use:N \g__pdf_backend_object_int } }

(End definition for \__pdf_backend_object_last:.)

```

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

```

2115 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2116   { { Page #1 } }

(End definition for \__pdf_backend_pageobject_ref:n.)

```

6.2.3 Annotations

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

`\l__pdf_backend_content_box` The content of an annotation.

```

2117 \box_new:N \l__pdf_backend_content_box

(End definition for \l__pdf_backend_content_box.)

```

`\l__pdf_backend_model_box` For creating model sizing for links.

```

2118 \box_new:N \l__pdf_backend_model_box

(End definition for \l__pdf_backend_model_box.)

```

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

```

2119 \int_new:N \g__pdf_backend_annotation_int

(End definition for \g__pdf_backend_annotation_int.)

```

`__pdf_backend_annotation:nnnn` Annotations are objects, but we track them separately. Notably, they are not in the object data lists. Here, to get the co-ordinates of the annotation, we need to have the data collected at the PostScript level. That requires a bit of box trickery (effectively a L^AT_EX 2_ε picture of zero size). Once the data is collected, use it to set up the annotation border.

```

2120 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2121   {
2122     \exp_args:Nf \__pdf_backend_annotation_aux:nnnn
2123       { \dim_eval:n {#1} } {#2} {#3} {#4}
2124   }
2125 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2126   {
2127     \box_move_down:nn {#3}
2128     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2129     \box_move_up:nn {#2}
2130     {
2131       \hbox:n
2132       {
2133         \__kernel_kern:n {#1}
2134         \__kernel_backend_postscript:n { pdf.save.ur }
2135         \__kernel_kern:n { -#1 }
2136       }

```



```

2137     }
2138     \int_gincr:N \g__pdf_backend_object_int
2139     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2140     \__pdf_backend_pdfmark:x
2141     {
2142         /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2143         pdf.rect
2144         #4 ~
2145         /ANN
2146     }
2147 }

```

(End definition for __pdf_backend_annotation:nnnn.)

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

```

2148 \cs_new:Npn \__pdf_backend_annotation_last:
2149 { { pdf.obj \int_use:N \g__pdf_backend_annotation_int } }

```

(End definition for __pdf_backend_annotation_last:.)

\g__pdf_backend_link_int To track annotations which are links.

```

2150 \int_new:N \g__pdf_backend_link_int

```

(End definition for \g__pdf_backend_link_int.)

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

```

2151 \tl_new:N \g__pdf_backend_link_dict_tl

```

(End definition for \g__pdf_backend_link_dict_tl.)

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

```

2152 \int_new:N \g__pdf_backend_link_sf_int

```

(End definition for \g__pdf_backend_link_sf_int.)

\g__pdf_backend_link_math_bool Needed to save/restore math mode.

```

2153 \bool_new:N \g__pdf_backend_link_math_bool

```

(End definition for \g__pdf_backend_link_math_bool.)

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

```

2154 \bool_new:N \g__pdf_backend_link_bool

```

(End definition for \g__pdf_backend_link_bool.)

\l__pdf_breaklink_pdfmark_tl Swappable content for link breaking.

```

2155 \tl_new:N \l__pdf_breaklink_pdfmark_tl
2156 \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdfmark }

```

(End definition for \l__pdf_breaklink_pdfmark_tl.)

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

```

2157 \cs_new_protected:Npn \__pdf_breaklink_postscript:n #1 { }

```

(End definition for __pdf_breaklink_postscript:n.)

`_pdf_breaklink_usebox:N` Swappable box unpacking or use.

2158 `\cs_new_eq:NN _pdf_breaklink_usebox:N \box_use:N`

(End definition for `_pdf_breaklink_usebox:N`.)

`_pdf_backend_link_begin_goto:nnw`

`_pdf_backend_link_begin_user:nnw`

`_pdf_backend_link:nw`

`_pdf_backend_link_aux:nw`

`_pdf_backend_link_end:`

`_pdf_backend_link_end_aux:`

`_pdf_backend_link_minima:`

`_pdf_backend_link_outerbox:n`

`_pdf_backend_link_sf_save:`

`_pdf_backend_link_sf_restore:`

`pdf.linkdp.pad`

`pdf.linkht.pad`

`pdf.llx`

`pdf.lly`

`pdf.ury`

`pdf.link.dict`

`pdf.outerbox`

`pdf.baselineskip`

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

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

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

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

```
2159 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2160 { \_pdf_backend_link_begin:nw { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2161 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2162 { \_pdf_backend_link_begin:nw {#1#2} }
2163 \cs_new_protected:Npn \_pdf_backend_link_begin:nw #1
2164 {
2165   \bool_if:NF \g__pdf_backend_link_bool
2166   { \_pdf_backend_link_begin_aux:nw {#1} }
2167 }
2168 \cs_new_protected:Npn \_pdf_backend_link_begin_aux:nw #1
2169 {
2170   \bool_gset_true:N \g__pdf_backend_link_bool
2171   \_kernel_backend_postscript:n
2172   { /pdf.link.dict ( #1 ) def }
2173   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2174   \_pdf_backend_link_sf_save:
2175   \mode_if_math:TF
2176   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2177   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2178   \hbox_set:Nw \l__pdf_backend_content_box
2179   \_pdf_backend_link_sf_restore:
2180   \bool_if:NT \g__pdf_backend_link_math_bool
2181   { \c_math_toggle_token }
2182 }
2183 \cs_new_protected:Npn \_pdf_backend_link_end:
2184 {
2185   \bool_if:NT \g__pdf_backend_link_bool
2186   { \_pdf_backend_link_end_aux: }
2187 }
2188 \cs_new_protected:Npn \_pdf_backend_link_end_aux:
2189 {
2190   \bool_if:NT \g__pdf_backend_link_math_bool
2191   { \c_math_toggle_token }
2192   \_pdf_backend_link_sf_save:
```

```

2193 \hbox_set_end:
2194 \__pdf_backend_link_minima:
2195 \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2196 \exp_args:Nx \__pdf_backend_link_outerbox:n
2197 {
2198   \int_if_odd:nTF { \value { page } }
2199     { \oddsidemargin }
2200     { \evensidemargin }
2201 }
2202 \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2203 { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2204 \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2205 \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2206 \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2207 \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2208 {
2209   \hbox:n
2210     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2211 }
2212 \int_gincr:N \g__pdf_backend_object_int
2213 \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2214 \__kernel_backend_postscript:x
2215 {
2216   mark
2217   /objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2218   \g__pdf_backend_link_dict_tl \c_space_tl
2219   pdf.rect
2220   /ANN ~ \l__pdf_breaklink_pdfmark_tl
2221 }
2222 \__pdf_backend_link_sf_restore:
2223 \bool_gset_false:N \g__pdf_backend_link_bool
2224 }
2225 \cs_new_protected:Npn \__pdf_backend_link_minima:
2226 {
2227   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2228   \__kernel_backend_postscript:x
2229   {
2230     /pdf.linkdp.pad ~
2231     \dim_to_decimal:n
2232     {
2233       \dim_max:nn
2234       {
2235         \box_dp:N \l__pdf_backend_model_box
2236         - \box_dp:N \l__pdf_backend_content_box
2237       }
2238       { Opt }
2239     } ~
2240     pdf.pt.dvi ~ def
2241     /pdf.linkht.pad ~
2242     \dim_to_decimal:n
2243     {
2244       \dim_max:nn
2245       {
2246         \box_ht:N \l__pdf_backend_model_box

```

```

2247         - \box_ht:N \l__pdf_backend_content_box
2248     }
2249     { Opt }
2250 } ~
2251 pdf.pt.dvi ~ def
2252 }
2253 }
2254 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2255 {
2256     \__kernel_backend_postscript:x
2257     {
2258         /pdf.outerbox
2259         [
2260             \dim_to_decimal:n {#1} ~
2261             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2262             \dim_to_decimal:n { #1 + \textwidth } ~
2263             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2264         ]
2265         [ exch { pdf.pt.dvi } forall ] def
2266         /pdf.baselineskip ~
2267         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2268         { pdf.pt.dvi ~ def }
2269         { pop ~ pop }
2270     ifelse
2271 }
2272 }
2273 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2274 {
2275     \int_gset:Nn \g__pdf_backend_link_sf_int
2276     {
2277         \mode_if_horizontal:TF
2278         { \tex_spacefactor:D }
2279         { 0 }
2280     }
2281 }
2282 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2283 {
2284     \mode_if_horizontal:T
2285     {
2286         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2287         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2288     }
2289 }

```

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

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

```

2290 \use_none:n
2291 {
2292     \cs_if_exist:NT \@makecol@hook
2293     {

```

```

2294 \tl_put_right:Nn \@makecol@hook
2295 {
2296   \box_if_empty:NF \@cclv
2297   {
2298     \vbox_set:Nn \@cclv
2299     {
2300       \__kernel_backend_postscript:n
2301       {
2302         pdf.globaldict /pdf.brokenlink.rect ~ known
2303         { pdf.bordertracking.continue }
2304         if
2305       }
2306       \vbox_unpack_drop:N \@cclv
2307       \__kernel_backend_postscript:n
2308       { pdf.bordertracking.endpage }
2309     }
2310   }
2311 }
2312 \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2313 \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2314 \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2315 }
2316 }

```

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

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

```

2317 \cs_new:Npn \__pdf_backend_link_last:
2318 { { pdf.obj \int_use:N \g__pdf_backend_link_int } }

```

(End definition for __pdf_backend_link_last:.)

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

```

2319 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2320 {
2321   \__kernel_backend_postscript:x
2322   {
2323     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2324   }
2325 }

```

(End definition for __pdf_backend_link_margin:n.)

_pdf_backend_destination:nn
_pdf_backend_destination:nnnn
_pdf_backend_destination_aux:nnnn

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

```

2326 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2327 {
2328   \__kernel_backend_postscript:n { pdf.dest.anchor }
2329   \__pdf_backend_pdfmark:x
2330   {
2331     /View
2332     [

```

```

2333     \str_case:nnF {#2}
2334     {
2335         { xyz }    { /XYZ ~ pdf.dest.point ~ null }
2336         { fit }    { /Fit }
2337         { fitb }   { /FitB }
2338         { fitbh }  { /FitBH ~ pdf.dest.y }
2339         { fitbv }  { /FitBV ~ pdf.dest.x }
2340         { fith }   { /FitH ~ pdf.dest.y }
2341         { fitv }   { /FitV ~ pdf.dest.x }
2342         { fitr }   { /Fit }
2343     }
2344     {
2345         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2346     }
2347 ]
2348 /Dest ( \exp_not:n {#1} ) cvn
2349 /DEST
2350 }
2351 }
2352 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2353 {
2354     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2355     { \dim_eval:n {#2} } {#1} {#3} {#4}
2356 }
2357 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2358 {
2359     \vbox_to_zero:n
2360     {
2361         \__kernel_kern:n {#4}
2362         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2363         \tex_vss:D
2364     }
2365     \__kernel_kern:n {#1}
2366     \vbox_to_zero:n
2367     {
2368         \__kernel_kern:n { -#3 }
2369         \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2370         \tex_vss:D
2371     }
2372     \__kernel_kern:n { -#1 }
2373     \__pdf_backend_pdfmark:n
2374     {
2375         /View
2376         [
2377             /FitR ~
2378             pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2379             pdf.urx ~ pdf.ury ~ pdf.dest2device
2380         ]
2381         /Dest ( #2 ) cvn
2382         /DEST
2383     }
2384 }

```

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

6.2.4 Structure

_pdf_backend_compresslevel:n
_pdf_backend_compress_objects:n

Doable for the usual ps2pdf method.

```

2385 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2386 {
2387   \int_compare:nNnT {#1} = 0
2388   {
2389     \__kernel_backend_literal_postscript:n
2390     {
2391       /setdistillerparams ~ where
2392       { pop << /CompressPages ~ false >> setdistillerparams }
2393       if
2394     }
2395   }
2396 }
2397 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2398 {
2399   \bool_if:nF {#1}
2400   {
2401     \__kernel_backend_literal_postscript:n
2402     {
2403       /setdistillerparams ~ where
2404       { pop << /CompressStreams ~ false >> setdistillerparams }
2405       if
2406     }
2407   }
2408 }
```

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

_pdf_backend_version_major_gset:n
_pdf_backend_version_minor_gset:n

Data not available!

```

2409 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
2410 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1 { }
```

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

_pdf_backend_version_major:
_pdf_backend_version_minor:

Data not available!

```

2411 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2412 \cs_new:Npn \_pdf_backend_version_minor: { -1 }
```

(End definition for _pdf_backend_version_major: and _pdf_backend_version_minor:.)

6.2.5 Marked content

_pdf_backend_bdc:nn
_pdf_backend_emc:

Simple wrappers.

```

2413 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2414 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2415 \cs_new_protected:Npn \_pdf_backend_emc:
2416 { \_pdf_backend_pdfmark:n { /EMC } }
```

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

```

2417 </dvips>
```

6.3 LuaTeX and pdfTeX backend

2418 `<*luatex | pdftex>`

6.3.1 Annotations

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

```

2419 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2420 {
2421   <*luatex>
2422     \tex_pdfextension:D annot ~
2423   </luatex>
2424   <*pdftex>
2425     \tex_pdfannot:D
2426   </pdftex>
2427     width ~ \dim_eval:n {#1} ~
2428     height ~ \dim_eval:n {#2} ~
2429     depth ~ \dim_eval:n {#3} ~
2430     {#4}
2431 }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

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

```

2432 \cs_new:Npx \_pdf_backend_annotation_last:
2433 {
2434   \exp_not:N \int_value:w
2435   <*luatex>
2436     \exp_not:N \tex_pdffeedback:D lastannot ~
2437   </luatex>
2438   <*pdftex>
2439     \exp_not:N \tex_pdflastannot:D
2440   </pdftex>
2441     \c_space_tl 0 ~ R
2442 }
```

(End definition for `_pdf_backend_annotation_last:`.)

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

```

\_pdf_backend_link_begin_user:nnw 2443 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
\_pdf_backend_link_begin:nnnw 2444 { \_pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
\_pdf_backend_link_end: 2445 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2446 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2447 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2448 {
2449   <*luatex>
2450     \tex_pdfextension:D startlink ~
2451   </luatex>
2452   <*pdftex>
2453     \tex_pdfstartlink:D
2454   </pdftex>
2455     attr {#1}
2456     #2 {#3}
```



```

2457 }
2458 \cs_new_protected:Npn \__pdf_backend_link_end:
2459 {
2460   \*luatex
2461     \tex_pdfextension:D endlink \scan_stop:
2462   \*pdftex
2463     \tex_pdfendlink:D
2464   \*pdftex
2465 }
2466

```

(End definition for __pdf_backend_link_begin_goto:nmw and others.)

__pdf_backend_link_last: Formatted for direct use.

```

2467 \cs_new:Npx \__pdf_backend_link_last:
2468 {
2469   \exp_not:N \int_value:w
2470   \*luatex
2471     \exp_not:N \tex_pdffeedback:D lastlink ~
2472   \*pdftex
2473     \exp_not:N \tex_pdflastlink:D
2474   \c_space_tl 0 ~ R
2475 }
2476

```

(End definition for __pdf_backend_link_last:.)

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

```

2478 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2479 {
2480   \*luatex
2481     \tex_pdfvariable:D linkmargin
2482   \*pdftex
2483     \tex_pdflinkmargin:D
2484   \dim_eval:n {#1} \scan_stop:
2485 }
2486

```

(End definition for __pdf_backend_link_margin:n.)

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

__pdf_backend_destination:nnnn

```

2488 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2489 {
2490   \*luatex
2491     \tex_pdfextension:D dest ~
2492   \*pdftex
2493     \tex_pdfdest:D
2494     name {#1}
2495     \str_case:nnF {#2}
2496

```

```

2498         {
2499             { xyz } { xyz }
2500             { fit } { fit }
2501             { fitb } { fitb }
2502             { fitbh } { fitbh }
2503             { fitbv } { fitbv }
2504             { fith } { fith }
2505             { fitv } { fitv }
2506             { fitr } { fitr }
2507         }
2508         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2509         \scan_stop:
2510     }
2511     \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2512     {
2513         \*luatex
2514         \tex_pdfextension:D dest ~
2515         \*pdftex
2516         \tex_pdfdest:D
2517         \*pdftex
2518         name {#1}
2519         fitr ~
2520         width \dim_eval:n {#2} ~
2521         height \dim_eval:n {#3} ~
2522         depth \dim_eval:n {#4} \scan_stop:
2523     }
2524 }

```

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

6.3.2 Catalogue entries

```

\__pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
2525 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2526 {
2527     \*luatex
2528     \tex_pdfextension:D catalog
2529     \*pdftex
2530     \tex_pdfcatalog:D
2531     \*pdftex
2532     { / #1 ~ #2 }
2533 }
2534 }
2535 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2536 {
2537     \*luatex
2538     \tex_pdfextension:D info
2539     \*pdftex
2540     \tex_pdfinfo:D
2541     \*pdftex
2542     { / #1 ~ #2 }
2543 }
2544 }

```

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

6.3.3 Objects

`\g__pdf_backend_object_prop` For tracking objects to allow finalisation.

```
2545 \prop_new:N \g__pdf_backend_object_prop
```

(End definition for `\g__pdf_backend_object_prop`.)

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

`__pdf_backend_object_ref:n`

```
2546 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2547 {
2548   \*luatex
2549   \tex_pdfextension:D obj ~
2550   \*pdfTeX
2551   \tex_pdfobj:D
2552   \*pdfTeX
2553   \*pdfTeX
2554   reserveobjnum ~
2555   \int_const:cn
2556   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2557   \*luatex
2558   { \tex_pdffeedback:D lastobj }
2559   \*pdfTeX
2560   \*pdfTeX
2561   { \tex_pdflastobj:D }
2562   \*pdfTeX
2563   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2564   }
2565   \cs_new:Npn \__pdf_backend_object_ref:n #1
2566   { \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }
```

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

`__pdf_backend_object_write:nn`

`__pdf_backend_object_write:nx`

`__pdf_exp_not_i:nn`

`__pdf_exp_not_ii:nn`

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

```
2567 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2568 {
2569   \*luatex
2570   \tex_immediate:D \tex_pdfextension:D obj ~
2571   \*pdfTeX
2572   \tex_immediate:D \tex_pdfobj:D
2573   \*pdfTeX
2574   \*pdfTeX
2575   useobjnum ~
2576   \int_use:c
2577   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2578   \str_case_e:nn
2579   { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2580   {
2581     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2582     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2583     { fstream }
2584     {
2585       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2586       file ~ { \__pdf_exp_not_ii:nn #2 }
2587     }
2588     { stream }
```

```

2589         {
2590             stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2591             { \_pdf_exp_not_ii:nn #2 }
2592         }
2593     }
2594 }
2595 \cs_generate_variant:Nn \_pdf_backend_object_write:nn { nx }
2596 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2597 \cs_new:Npn \_pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

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

```

_pdf_backend_object_now:nn Much like writing, but direct creation.

```

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

(End definition for \_pdf_backend_object_now:nn.)

```

_pdf_backend_object_last: Much like annotation.

```

2624 \cs_new:Npx \_pdf_backend_object_last:
2625 {
2626     \exp_not:N \int_value:w
2627     <*luatex>
2628     \exp_not:N \tex_pdffeedback:D lastobj ~
2629     </luatex>
2630     <*pdfTeX>
2631     \exp_not:N \tex_pdflastobj:D
2632     </pdfTeX>
2633     \c_space_tl 0 ~ R
2634 }

```

(End definition for `_pdf_backend_object_last:.`)

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

```

2635 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2636 {
2637   \exp_not:N \int_value:w
2638   <*luatex>
2639   \exp_not:N \tex_pdffeedback:D pageref
2640   </luatex>
2641   <*pdftex>
2642   \exp_not:N \tex_pdfpageref:D
2643   </pdftex>
2644   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2645 }
```

(End definition for `_pdf_backend_pageobject_ref:n`.)

6.3.4 Structure

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

```

\_pdf_backend_compress_objects:n 2646 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
\_pdf_backend_objcompresslevel:n 2647 {
2648   \tex_global:D
2649   <*luatex>
2650   \tex_pdfvariable:D compresslevel
2651   </luatex>
2652   <*pdftex>
2653   \tex_pdfcompresslevel:D
2654   </pdftex>
2655   \int_value:w \int_eval:n {#1} \scan_stop:
2656 }
2657 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2658 {
2659   \bool_if:nTF {#1}
2660   { \_pdf_backend_objcompresslevel:n { 2 } }
2661   { \_pdf_backend_objcompresslevel:n { 0 } }
2662 }
2663 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2664 {
2665   \tex_global:D
2666   <*luatex>
2667   \tex_pdfvariable:D objcompresslevel
2668   </luatex>
2669   <*pdftex>
2670   \tex_pdfobjcompresslevel:D
2671   </pdftex>
2672   #1 \scan_stop:
2673 }
```

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

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

```

\_pdf_backend_version_minor_gset:n 2674 \cs_new_protected:Npx \_pdf_backend_version_major_gset:n #1
```

```

2675 {
2676 <*luatex>
2677   \int_compare:nNnT \tex luatexversion:D > { 106 }
2678   {
2679     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2680     \exp_not:N \int_eval:n {#1} \scan_stop:
2681   }
2682 </luatex>
2683 <*pdftex>
2684   \cs_if_exist:NT \tex_pdfmajorversion:D
2685   {
2686     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2687     \exp_not:N \int_eval:n {#1} \scan_stop:
2688   }
2689 </pdftex>
2690 }
2691 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2692 {
2693   \tex_global:D
2694   <*luatex>
2695     \tex_pdfvariable:D minorversion
2696   </luatex>
2697   <*pdftex>
2698     \tex_pdfminorversion:D
2699   </pdftex>
2700   \int_eval:n {#1} \scan_stop:
2701 }

```

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

__pdf_backend_version_major: As above.
 __pdf_backend_version_minor:

```

2702 \cs_new:Npx \__pdf_backend_version_major:
2703 {
2704 <*luatex>
2705   \int_compare:nNnTF \tex luatexversion:D > { 106 }
2706   { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2707   { 1 }
2708 </luatex>
2709 <*pdftex>
2710   \cs_if_exist:NTF \tex_pdfmajorversion:D
2711   { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2712   { 1 }
2713 </pdftex>
2714 }
2715 \cs_new:Npn \__pdf_backend_version_minor:
2716 {
2717   \tex_the:D
2718   <*luatex>
2719     \tex_pdfvariable:D minorversion
2720   </luatex>
2721   <*pdftex>
2722     \tex_pdfminorversion:D
2723   </pdftex>
2724 }

```

(End definition for `_pdf_backend_version_major:` and `_pdf_backend_version_minor:`.)

6.3.5 Marked content

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

```
2725 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2726   { \_kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2727 \cs_new_protected:Npn \_pdf_backend_emc:
2728   { \_kernel_backend_literal_page:n { EMC } }
```

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

```
2729 </luatex | pdftex>
```

6.4 dvipdfmx backend

```
2730 <*dvipdfmx | xetex>
```

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

```
\_pdf_backend:x
2731 \cs_new_protected:Npx \_pdf_backend:n #1
2732   { \_kernel_backend_literal:n { pdf: #1 } }
2733 \cs_generate_variant:Nn \_pdf_backend:n { x }
```

(End definition for `_pdf_backend:n`.)

6.4.1 Catalogue entries

```
\_pdf_backend_catalog_gput:nn
\_pdf_backend_info_gput:nn
2734 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2735   { \_pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2736 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2737   { \_pdf_backend:n { docinfo << /#1 ~ #2 >> } }
```

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

6.4.2 Objects

`\g__pdf_backend_object_int` For tracking objects to allow finalisation.

```
\g__pdf_backend_object_prop
2738 \int_new:N \g__pdf_backend_object_int
2739 \prop_new:N \g__pdf_backend_object_prop
```

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

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

```
\_pdf_backend_object_ref:n
2740 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2
2741   {
2742     \int_gincr:N \g__pdf_backend_object_int
2743     \int_const:cn
2744     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2745     { \g__pdf_backend_object_int }
2746     \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2747   }
2748 \cs_new:Npn \_pdf_backend_object_ref:n #1
2749   { @pdf.obj \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } }
```

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

This is where we choose the actual type.

```

\__pdf_backend_object_write:nn
\__pdf_backend_object_write:nx
\__pdf_backend_object_write:nnn
\__pdf_backend_object_write_array:nn
\__pdf_backend_object_write_dict:nn
\__pdf_backend_object_write_fstream:nn
\__pdf_backend_object_write_stream:nn
\__pdf_backend_object_write_stream:nnnn
2750 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2751 {
2752   \exp_args:Nx \__pdf_backend_object_write:nnn
2753   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2754 }
2755 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2756 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2757 {
2758   \use:c { \__pdf_backend_object_write_ #1 :nn }
2759   { \__pdf_backend_object_ref:n {#2} } {#3}
2760 }
2761 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2762 {
2763   \__pdf_backend:x
2764   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2765 }
2766 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2767 {
2768   \__pdf_backend:x
2769   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2770 }
2771 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2772 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2773 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2774 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2775 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2776 {
2777   \__pdf_backend:x
2778   {
2779     #1 stream ~ #2 ~
2780     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2781   }
2782 }

```

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

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

```

\__pdf_backend_object_now:nn
\__pdf_backend_object_now:nx
2783 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2784 {
2785   \int_gincr:N \g__pdf_backend_object_int
2786   \exp_args:Nnx \use:c { \__pdf_backend_object_write_ #1 :nn }
2787   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2788   {#2}
2789 }
2790 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for `_pdf_backend_object_now:nn`.)

`__pdf_backend_object_last:`

```

2791 \cs_new:Npn \__pdf_backend_object_last:
2792 { @pdf.obj \int_use:N \g__pdf_backend_object_int }

```


(End definition for _pdf_backend_object_last:.)

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

```
2793 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2794 { @page #1 }
```

(End definition for _pdf_backend_pageobject_ref:n.)

6.4.3 Annotations

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

```
2795 \bool_new:N \g__pdf_landscape_bool
2796 \cs_if_exist:NT \landscape
2797 {
2798   \tl_put_right:Nn \landscape
2799     { \bool_gset_true:N \g__pdf_landscape_bool }
2800   \tl_put_left:Nn \endlandscape
2801     { \bool_gset_false:N \g__pdf_landscape_bool }
2802 }
```

(End definition for \g__pdf_landscape_bool.)

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

```
2803 \int_new:N \g_pdf_backend_annotation_int
```

(End definition for \g_pdf_backend_annotation_int.)

_pdf_backend_annotation:nnnn Simply pass the raw data through, just dealing with evaluation of dimensions. The only wrinkle is landscape: we have to adjust by hand.

```
2804 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2805 {
2806   \bool_if:NTF \g__pdf_landscape_bool
2807   {
2808     \box_move_up:nn {#2}
2809     {
2810       \vbox:n
2811       {
2812         \_pdf_backend_annotation_aux:nnnn
2813         { #2 + #3 } {#1} { Opt } {#4}
2814       }
2815     }
2816   }
2817   { \_pdf_backend_annotation_aux:nnnn {#1} {#2} {#3} {#4} }
2818 }
2819 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nnnn #1#2#3#4
2820 {
2821   \int_gincr:N \g_pdf_backend_object_int
2822   \int_gset_eq:NN \g_pdf_backend_annotation_int \g_pdf_backend_object_int
2823   \_pdf_backend:x
2824   {
2825     ann ~ @pdf.obj \int_use:N \g_pdf_backend_object_int \c_space_tl
2826     width ~ \dim_eval:n {#1} ~
2827     height ~ \dim_eval:n {#2} ~
```

```

2828         depth ~ \dim_eval:n {#3} ~
2829         <</Type/Annot #4 >>
2830     }
2831 }

```

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

_pdf_backend_annotation_last:

```

2832 \cs_new:Npn \_pdf_backend_annotation_last:
2833 { @pdf.obj \int_use:N \g_pdf_backend_annotation_int }

```

(End definition for _pdf_backend_annotation_last:.)

\g_pdf_backend_link_int To track annotations which are links.

```

2834 \int_new:N \g_pdf_backend_link_int

```

(End definition for \g_pdf_backend_link_int.)

_pdf_backend_link_begin_goto:nnw All created using the same internals.

_pdf_backend_link_begin_user:nnw

_pdf_backend_link_begin:n

_pdf_backend_link_end:

```

2835 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2836 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
2837 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2838 { \_pdf_backend_link_begin:n {#1#2} }
2839 \cs_new_protected:Npx \_pdf_backend_link_begin:n #1
2840 {
2841     \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2842     {
2843         \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int
2844     }
2845     \_pdf_backend:x
2846     {
2847         bann ~
2848         \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2849         {
2850             @pdf.lnk
2851             \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int
2852             \c_space_tl
2853         }
2854         <<
2855         /Type /Annot
2856         #1
2857         >>
2858     }
2859 }
2860 \cs_new_protected:Npn \_pdf_backend_link_end:
2861 { \_pdf_backend:n { eann } }

```

(End definition for _pdf_backend_link_begin_goto:nnw and others.)

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

```

2862 \cs_new:Npx \_pdf_backend_link_last:
2863 {
2864     \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
2865     {
2866         @pdf.lnk

```

```

2867         \exp_not:N \int_use:N \exp_not:N \g__pdf_backend_link_int
2868     }
2869 }

```

(End definition for __pdf_backend_link_last:.)

__pdf_backend_link_margin:n Pass to dvipdfmx.

```

2870 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2871 { \__kernel_backend_literal:x { dvipdfmx:config-g~ \dim_eval:n {#1} } }

```

(End definition for __pdf_backend_link_margin:n.)

```

\__pdf_backend_destination:nn
\__pdf_backend_destination:nnnn
\__pdf_backend_destination_aux:nnnn

```

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

```

2872 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2873 {
2874     \__pdf_backend:x
2875     {
2876         dest ~ ( \exp_not:n {#1} )
2877         [
2878             @thispage
2879             \str_case:nnF {#2}
2880             {
2881                 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2882                 { fit } { /Fit }
2883                 { fitb } { /FitB }
2884                 { fitbh } { /FitBH }
2885                 { fitbv } { /FitBV ~ @xpos }
2886                 { fith } { /FitH ~ @ypos }
2887                 { fitv } { /FitV ~ @xpos }
2888                 { fitr } { /Fit }
2889             }
2890             { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2891         ]
2892     }
2893 }
2894 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2895 {
2896     \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2897     { \dim_eval:n {#2} } {#1} {#3} {#4}
2898 }
2899 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2900 {
2901     \vbox_to_zero:n
2902     {
2903         \__kernel_kern:n {#4}
2904         \hbox:n
2905         {
2906             \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2907             \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2908         }
2909         \tex_vss:D

```

```

2910     }
2911     \__kernel_kern:n {#1}
2912     \vbox_to_zero:n
2913     {
2914         \__kernel_kern:n { -#3 }
2915         \hbox:n
2916         {
2917             \__pdf_backend:n
2918             {
2919                 dest ~ (#2)
2920                 [
2921                     @thispage
2922                     /FitR ~
2923                     @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2924                     @xpos ~ @ypos
2925                 ]
2926             }
2927         }
2928         \tex_vss:D
2929     }
2930     \__kernel_kern:n { -#1 }
2931 }

```

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

6.4.4 Structure

_pdf_backend_compresslevel:n
_pdf_backend_compress_objects:n

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

```

2932 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2933 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
2934 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2935 {
2936     \bool_if:nF {#1}
2937     { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2938 }

```

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

_pdf_backend_version_major_gset:n
_pdf_backend_version_minor_gset:n

We start with the assumption that the default is active.

```

2939 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2940 {
2941     \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
2942     \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
2943 }
2944 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2945 {
2946     \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
2947     \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
2948 }

```

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

_pdf_backend_version_major:
_pdf_backend_version_minor:

We start with the assumption that the default is active.

```

2949 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2950 \cs_new:Npn \__pdf_backend_version_minor: { 5 }

```

(End definition for `_pdf_backend_version_major:` and `_pdf_backend_version_minor:`.)

6.4.5 Marked content

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

```

2951 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2952   { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2953 \cs_new_protected:Npn \_pdf_backend_emc:
2954   { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2955 </dviptfm | xetex>

```

6.5 dvisvgm backend

```

2956 <*dvisvgm>

```

6.5.1 Catalogue entries

`_pdf_backend_catalog_gput:nn`
`_pdf_backend_info_gput:nn`

No-op.

```

2957 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2 { }
2958 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2 { }

```

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

6.5.2 Objects

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

All no-ops here.

```

2959 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1#2 { }
2960 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
2961 \cs_new_protected:Npn \_pdf_backend_object_write:nn #1#2 { }
2962 \cs_new_protected:Npn \_pdf_backend_object_write:nx #1#2 { }
2963 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
2964 \cs_new_protected:Npn \_pdf_backend_object_now:nx #1#2 { }
2965 \cs_new:Npn \_pdf_backend_object_last: { }
2966 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1 { }

```

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

6.5.3 Structure

`_pdf_backend_compresslevel:n`
`_pdf_backend_compress_objects:n`

These are all no-ops.

```

2967 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1 { }
2968 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1 { }

```

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

`_pdf_backend_version_major_gset:n`
`_pdf_backend_version_minor_gset:n`

Data not available!

```

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

```

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

```

\__pdf_backend_version_major: Data not available!
\__pdf_backend_version_minor: 2971 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2972 \cs_new:Npn \__pdf_backend_version_minor: { -1 }

(End definition for \__pdf_backend_version_major: and \__pdf_backend_version_minor:.)

\__pdf_backend_bdc:nn More no-ops.
\__pdf_backend_emc: 2973 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2974 \cs_new_protected:Npn \__pdf_backend_emc: { }

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

2975 \</divisvgn>
2976 \</package>

```

7 l3backend-opacity Implementation

```

2977 \*package>
2978 \@@=opacity>

```

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

```

2979 \*dvips>

```

```

\__opacity_backend_select:n No stack so set values directly.
\__opacity_backend_select_aux:n 2980 \cs_new_protected:Npn \__opacity_backend_select:n #1
2981 {
2982   \exp_args:Nx \__opacity_backend_select_aux:n
2983   { \fp_eval:n { min(max(0,#1),1) } }
2984 }
2985 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
2986 {
2987   \__kernel_backend_postscript:n
2988   { #1 ~ .setfillconstantalpha ~ #1 ~ .setstrokeconstantalpha }
2989 }

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

```

```

\__opacity_backend_fill:n Similar to the above but with no stack and only adding to one or other of the entries.
\__opacity_backend_stroke:n 2990 \cs_new_protected:Npn \__opacity_backend_fill:n #1
\__opacity_backend:nn 2991 { \__opacity_backend:xn { \fp_eval:n { min(max(0,#1),1) } } { fill } }
\__opacity_backend:xn 2992 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
2993 { \__opacity_backend:xn { \fp_eval:n { min(max(0,#1),1) } } { stroke } }
2994 \cs_new_protected:Npn \__opacity_backend:nn #1#2
2995 {
2996   \__kernel_backend_postscript:n { #1 ~ .set #2 constantalpha }
2997 }
2998 \cs_generate_variant:Nn \__opacity_backend:nn { x }

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

2999 \</dvips>

```

3000 \langle *dvipdfmx | luatex | pdftex | xetex \rangle

\backslash c_opacity_backend_stack_int Set up a stack.

```
3001 \cs_if_exist:NT \pdfmanagement_add:nnn
3002 {
3003   \__kernel_color_backend_stack_init:Nnn \c_opacity_backend_stack_int
3004   { page ~ direct } { /opacity 1 ~ gs }
3005   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3006   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3007 }
```

(End definition for \backslash c_opacity_backend_stack_int.)

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

```
\l_opacity_backend_stroke_tl
3008 \tl_new:N \l_opacity_backend_fill_tl
3009 \tl_new:N \l_opacity_backend_stroke_tl
```

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

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

```
\_opacity_backend_select_aux:n
\_opacity_backend_reset:
3010 \cs_new_protected:Npn \_opacity_backend_select:n #1
3011 {
3012   \exp_args:Nx \_opacity_backend_select_aux:n
3013   { \fp_eval:n { min(max(0,#1),1) } }
3014 }
3015 \cs_new_protected:Npn \_opacity_backend_select_aux:n #1
3016 {
3017   \tl_set:Nn \l_opacity_backend_fill_tl {#1}
3018   \tl_set:Nn \l_opacity_backend_stroke_tl {#1}
3019   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3020   { opacity #1 }
3021   { << /ca ~ #1 /CA ~ #1 >> }
3022   \_opacity_backend_stack_push:nn \c_opacity_backend_stack_int
3023   { /opacity #1 ~ gs }
3024   \group_insert_after:N \_opacity_backend_reset:
3025 }
3026 \cs_if_exist:NF \pdfmanagement_add:nnn
3027 {
3028   \cs_gset_protected:Npn \_opacity_backend_select_aux:n #1 { }
3029 }
3030 \cs_new_protected:Npn \_opacity_backend_reset:
3031 { \_opacity_backend_stack_pop:n \c_opacity_backend_stack_int }
```

(End definition for \backslash _opacity_backend_select:n, \backslash _opacity_backend_select_aux:n, and \backslash _opacity_backend_reset:.)

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

```
\_opacity_backend_stroke:n
\_opacity_backend_fillstroke:nn
\_opacity_backend_fillstroke:xx
3032 \cs_new_protected:Npn \_opacity_backend_fill:n #1
3033 {
3034   \_opacity_backend_fill_stroke:xx
3035   { \fp_eval:n { min(max(0,#1),1) } }
3036   \l_opacity_backend_stroke_tl
3037 }
3038 \cs_new_protected:Npn \_opacity_backend_stroke:n #1
```

```

3039 {
3040   \_opacity_backend_fill_stroke:xx
3041   \l\_opacity_backend_fill_tl
3042   { \fp_eval:n { min(max(0,#1),1) } }
3043 }
3044 \cs_new_protected:Npn \_opacity_backend_fill_stroke:nn #1#2
3045 {
3046   \str_if_eq:nnTF {#1} {#2}
3047   { \_opacity_backend_select_aux:n {#1} }
3048   {
3049     \tl_set:Nn \l\_opacity_backend_fill_tl {#1}
3050     \tl_set:Nn \l\_opacity_backend_stroke_tl {#2}
3051     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3052     { opacity.fill #1 }
3053     { << /ca ~ #1 >> }
3054     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3055     { opacity.stroke #1 }
3056     { << /CA ~ #2 >> }
3057     \_opacity_backend_stack_push:nn \c\_opacity_backend_stack_int
3058     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3059     \group_insert_after:N \_opacity_backend_reset:
3060   }
3061 }
3062 \cs_generate_variant:Nn \_opacity_backend_fill_stroke:nn { xx }

```

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

```

3063 </dvipdfmx | luatex | pdftex | xetex>
3064 < *dvipdfmx | xdvipdfmx>

```

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

```

3065 \int_compare:nNnT \c_kernel_sys_dvipdfmx_version_int < { 20201111 }
3066 {
3067   \cs_gset_protected:Npn \_opacity_backend_select_aux:n #1
3068   {
3069     \tl_set:Nn \l\_opacity_backend_fill_tl {#1}
3070     \tl_set:Nn \l\_opacity_backend_stroke_tl {#1}
3071     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3072     { opacity #1 }
3073     { << /ca ~ #1 /CA ~ #1 >> }
3074     \_kernel_backend_literal_pdf:n { /opacity #1 ~ gs }
3075   }
3076   \cs_gset_protected:Npn \_opacity_backend_fill_stroke:nn #1#2
3077   {
3078     \str_if_eq:nnTF {#1} {#2}
3079     { \_opacity_backend_select_aux:n {#1} }
3080     {
3081       \tl_set:Nn \l\_opacity_backend_fill_tl {#1}
3082       \tl_set:Nn \l\_opacity_backend_stroke_tl {#2}
3083       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3084       { opacity.fill #1 }
3085       { << /ca ~ #1 >> }
3086       \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3087       { opacity.stroke #1 }

```



```

3088         { << /CA ~ #2 >> }
3089         \_kernel_backend_literal_pdf:n
3090         { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3091     }
3092 }
3093 }

```

(End definition for _opacity_backend_select:n.)

```

3094 </dvipdfmx | xdvipdfmx>

```

```

3095 <*dvisvgm>

```

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

```

\__opacity_backend_fill:n
\__opacity_backend_stroke:n
\__opacity_backend:nn
3096 \cs_new_protected:Npn \__opacity_backend_select:n #1
3097 { \__opacity_backend:nn {#1} { } }
3098 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3099 { \__opacity_backend:nn {#1} { fill- } }
3100 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3101 { \__opacity_backend:nn { {#1} } { stroke- } }
3102 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3103 { \_kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

```

(End definition for _opacity_backend_select:n and others.)

```

3104 </dvisvgm>

```

```

3105 </package>

```

8 l3backend-header Implementation

```

3106 <*dvips & header>

```

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

```

3107 /color.sc { } def

```

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

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

```

3108 TeXDict begin
3109 /TeXcolorseparation { setcolor } def
3110 end

```

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

pdf.globaldict A small global dictionary for backend use.

```

3111 true setglobal
3112 /pdf.globaldict 4 dict def
3113 false setglobal

```

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

pdf.cvs	Small utilities for PostScript manipulations. Conversion to DVI dimensions is done here
pdf.dvi.pt	to allow for Resolution. The total height of a rectangle (an array) needs a little maths,
pdf.pt.dvi	in contrast to simply extracting a value.
pdf.rect.ht	<pre> 3114 /pdf.cvs { 65534 string cvs } def 3115 /pdf.dvi.pt { 72.27 mul Resolution div } def 3116 /pdf.pt.dvi { 72.27 div Resolution mul } def 3117 /pdf.rect.ht { dup 1 get neg exch 3 get add } def </pre> <p>(End definition for pdf.cvs and others. These functions are documented on page ??.)</p>
pdf.linkmargin	Settings which are defined up-front in SDict.
pdf.linkdp.pad	<pre> 3118 /pdf.linkmargin { 1 pdf.pt.dvi } def </pre>
pdf.linkht.pad	<pre> 3119 /pdf.linkdp.pad { 0 } def 3120 /pdf.linkht.pad { 0 } def </pre> <p>(End definition for pdf.linkmargin, pdf.linkdp.pad, and pdf.linkht.pad. These functions are documented on page ??.)</p>
pdf.rect	Functions for marking the limits of an annotation/link, plus drawing the border. We
pdf.save.ll	separate links for generic annotations to support adding a margin and setting a minimal
pdf.save.ur	size.
pdf.save.linkll	<pre> 3121 /pdf.rect </pre>
pdf.save.linkur	<pre> 3122 { /Rect [pdf.llx pdf.lly pdf.urx pdf.ury] } def </pre>
pdf.llx	<pre> 3123 /pdf.save.ll </pre>
pdf.lly	<pre> 3124 { </pre>
pdf.urx	<pre> 3125 currentpoint </pre>
pdf.ury	<pre> 3126 /pdf.lly exch def 3127 /pdf.llx exch def 3128 } </pre>
	<pre> 3129 def 3130 /pdf.save.ur </pre>
	<pre> 3131 { </pre>
	<pre> 3132 currentpoint </pre>
	<pre> 3133 /pdf.ury exch def 3134 /pdf.urx exch def 3135 } </pre>
	<pre> 3136 def 3137 /pdf.save.linkll </pre>
	<pre> 3138 { </pre>
	<pre> 3139 currentpoint 3140 pdf.linkmargin add 3141 pdf.linkdp.pad add 3142 /pdf.lly exch def 3143 pdf.linkmargin sub 3144 /pdf.llx exch def 3145 } </pre>
	<pre> 3146 def 3147 /pdf.save.linkur </pre>
	<pre> 3148 { </pre>
	<pre> 3149 currentpoint 3150 pdf.linkmargin sub 3151 pdf.linkht.pad sub 3152 /pdf.ury exch def 3153 pdf.linkmargin add </pre>

```

3154     /pdf.urx exch def
3155   }
3156   def

```

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

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

```

pdf.dev.x 3157 /pdf.dest.anchor
pdf.dev.y 3158 {
pdf.tmpa 3159   currentpoint exch
pdf.tmpb 3160   pdf.dvi.pt 72 add
pdf.tmpc 3161   /pdf.dest.x exch def
pdf.tmpd 3162   pdf.dvi.pt
3163   vsize 72 sub exch sub
3164   /pdf.dest.y exch def
3165 }
3166 def
3167 /pdf.dest.point
3168 { pdf.dest.x pdf.dest.y } def
3169 /pdf.dest2device
3170 {
3171   /pdf.dest.y exch def
3172   /pdf.dest.x exch def
3173   matrix currentmatrix
3174   matrix defaultmatrix
3175   matrix invertmatrix
3176   matrix concatmatrix
3177   cvx exec
3178   /pdf.dev.y exch def
3179   /pdf.dev.x exch def
3180   /pdf.tmpd exch def
3181   /pdf.tmpc exch def
3182   /pdf.tmpb exch def
3183   /pdf.tmpa exch def
3184   pdf.dest.x pdf.tmpa mul
3185     pdf.dest.y pdf.tmpc mul add
3186     pdf.dev.x add
3187   pdf.dest.x pdf.tmpb mul
3188     pdf.dest.y pdf.tmpd mul add
3189     pdf.dev.y add
3190 }
3191 def

```

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

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

```

pdf.bordertracking.begin 3192 /pdf.bordertracking false def
pdf.bordertracking.end
pdf.leftboundary
pdf.rightboundary
pdf.brokenlink.rect
pdf.brokenlink.skip
pdf.brokenlink.dict
pdf.bordertracking.endpage
pdf.bordertracking.continue
pdf.originx
pdf.originy

```

```

3193 /pdf.bordertracking.begin
3194 {
3195     SDict /pdf.bordertracking true put
3196     SDict /pdf.leftboundary undef
3197     SDict /pdf.rightboundary undef
3198     /a where
3199     {
3200         /a
3201         {
3202             currentpoint pop
3203             SDict /pdf.rightboundary known dup
3204             {
3205                 SDict /pdf.rightboundary get 2 index lt
3206                 { not }
3207                 if
3208             }
3209             if
3210             { pop }
3211             { SDict exch /pdf.rightboundary exch put }
3212             ifelse
3213             moveto
3214             currentpoint pop
3215             SDict /pdf.leftboundary known dup
3216             {
3217                 SDict /pdf.leftboundary get 2 index gt
3218                 { not }
3219                 if
3220             }
3221             if
3222             { pop }
3223             { SDict exch /pdf.leftboundary exch put }
3224             ifelse
3225         }
3226         put
3227     }
3228     if
3229 }
3230 def
3231 /pdf.bordertracking.end
3232 {
3233     /a where { /a { moveto } put } if
3234     /x where { /x { 0 exch rmoveto } put } if
3235     SDict /pdf.leftboundary known
3236     { pdf.outerbox 0 pdf.leftboundary put }
3237     if
3238     SDict /pdf.rightboundary known
3239     { pdf.outerbox 2 pdf.rightboundary put }
3240     if
3241     SDict /pdf.bordertracking false put
3242 }
3243 def
3244 /pdf.bordertracking.endpage
3245 {
3246     pdf.bordertracking

```

```

3247 {
3248     pdf.bordertracking.end
3249     true setglobal
3250     pdf.globaldict
3251     /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3252     pdf.globaldict
3253     /pdf.brokenlink.skip pdf.baselineskip put
3254     pdf.globaldict
3255     /pdf.brokenlink.dict
3256     pdf.link.dict pdf.cvs put
3257     false setglobal
3258     mark pdf.link.dict cvx exec /Rect
3259     [
3260         pdf.llx
3261         pdf.lly
3262         pdf.outerbox 2 get pdf.linkmargin add
3263         currentpoint exch pop
3264         pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3265     ]
3266     /ANN pdf.pdfmark
3267 }
3268 if
3269 }
3270 def
3271 /pdf.bordertracking.continue
3272 {
3273     /pdf.link.dict pdf.globaldict
3274     /pdf.brokenlink.dict get def
3275     /pdf.outerbox pdf.globaldict
3276     /pdf.brokenlink.rect get def
3277     /pdf.baselineskip pdf.globaldict
3278     /pdf.brokenlink.skip get def
3279     pdf.globaldict dup dup
3280     /pdf.brokenlink.dict undef
3281     /pdf.brokenlink.skip undef
3282     /pdf.brokenlink.rect undef
3283     currentpoint
3284     /pdf.originy exch def
3285     /pdf.originx exch def
3286     /a where
3287     {
3288         /a
3289         {
3290             moveto
3291             SDict
3292             begin
3293                 currentpoint pdf.originy ne exch
3294                 pdf.originx ne or
3295                 {
3296                     pdf.save.linkll
3297                     /pdf.lly
3298                     pdf.lly pdf.outerbox 1 get sub def
3299                     pdf.bordertracking.begin
3300                 }

```

```

3301         if
3302         end
3303     }
3304     put
3305 }
3306 if
3307 /x where
3308 {
3309     /x
3310     {
3311         0 exch rmoveto
3312         SDict
3313         begin
3314         currentpoint
3315         pdf.originy ne exch pdf.originx ne or
3316         {
3317             pdf.save.link11
3318             /pdf.lly
3319             pdf.lly pdf.outerbox 1 get sub def
3320             pdf.bordertracking.begin
3321         }
3322         if
3323         end
3324     }
3325     put
3326 }
3327 if
3328 }
3329 def

```

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

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

```

3330 /pdf.breaklink
3331 {
3332     pop
3333     counttomark 2 mod 0 eq
3334     {
3335         counttomark /pdf.count exch def
3336         {
3337             pdf.count 0 eq { exit } if
3338             counttomark 2 roll
3339             1 index /Rect eq
3340             {
3341                 dup 4 array copy
3342                 dup dup
3343                 1 get
3344                 pdf.outerbox pdf.rect.ht
3345                 pdf.linkmargin 2 mul add sub
3346                 3 exch put

```

```

3347     dup
3348     pdf.outerbox 2 get
3349     pdf.linkmargin add
3350     2 exch put
3351   dup dup
3352     3 get
3353     pdf.outerbox pdf.rect.ht
3354     pdf.linkmargin 2 mul add add
3355     1 exch put
3356   /pdf.currentrect exch def
3357   pdf.breaklink.write
3358   {
3359     pdf.currentrect
3360     dup
3361     pdf.outerbox 0 get
3362     pdf.linkmargin sub
3363     0 exch put
3364     dup
3365     pdf.outerbox 2 get
3366     pdf.linkmargin add
3367     2 exch put
3368     dup dup
3369     1 get
3370     pdf.baselineskip add
3371     1 exch put
3372     dup dup
3373     3 get
3374     pdf.baselineskip add
3375     3 exch put
3376     /pdf.currentrect exch def
3377     pdf.breaklink.write
3378   }
3379   1 index 3 get
3380   pdf.linkmargin 2 mul add
3381   pdf.outerbox pdf.rect.ht add
3382   2 index 1 get sub
3383   pdf.baselineskip div round cvi 1 sub
3384   exch
3385   repeat
3386   pdf.currentrect
3387   dup
3388     pdf.outerbox 0 get
3389     pdf.linkmargin sub
3390     0 exch put
3391   dup dup
3392     1 get
3393     pdf.baselineskip add
3394     1 exch put
3395   dup dup
3396     3 get
3397     pdf.baselineskip add
3398     3 exch put
3399   dup 2 index 2 get 2 exch put
3400   /pdf.currentrect exch def

```

```

3401         pdf.breaklink.write
3402         SDict /pdf.pdfmark.good false put
3403         exit
3404     }
3405     { pdf.count 2 sub /pdf.count exch def }
3406     ifelse
3407 }
3408 loop
3409 }
3410 if
3411 /ANN
3412 }
3413 def
3414 /pdf.breaklink.write
3415 {
3416     counttomark 1 sub
3417     index /_objdef eq
3418     {
3419         counttomark -2 roll
3420         dup wcheck
3421         {
3422             readonly
3423             counttomark 2 roll
3424         }
3425         { pop pop }
3426         ifelse
3427     }
3428     if
3429     counttomark 1 add copy
3430     pop pdf.currentrect
3431     /ANN pdfmark
3432 }
3433 def

```

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

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

```

3434 /pdf.pdfmark
3435 {
3436     SDict /pdf.pdfmark.good true put
3437     dup /ANN eq
3438     {
3439         pdf.pdfmark.store
3440         pdf.pdfmark.dict
3441         begin
3442             Subtype /Link eq
3443             currentdict /Rect known and
3444             SDict /pdf.outerbox known and
3445             SDict /pdf.baselineskip known and
3446             {

```



```

3447         Rect 3 get
3448         pdf.linkmargin 2 mul add
3449         pdf.outerbox pdf.rect.ht add
3450         Rect 1 get sub
3451         pdf.baselineskip div round cvi 0 gt
3452         { pdf.breaklink }
3453         if
3454     }
3455     if
3456     end
3457     SDict /pdf.outerbox undef
3458     SDict /pdf.baselineskip undef
3459     currentdict /pdf.pdfmark.dict undef
3460 }
3461 if
3462 pdf.pdfmark.good
3463 { pdfmark }
3464 { cleartomark }
3465 ifelse
3466 }
3467 def
3468 /pdf.pdfmark.store
3469 {
3470     /pdf.pdfmark.dict 65534 dict def
3471     counttomark 1 add copy
3472     pop
3473     {
3474         dup mark eq
3475         {
3476             pop
3477             exit
3478         }
3479         {
3480             pdf.pdfmark.dict
3481             begin def end
3482         }
3483         ifelse
3484     }
3485     loop
3486 }
3487 def

```

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

```

3488 </dvips & header>

```

Index

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

- _ 146
- A**
- \AtBeginDvi 59, 60, 516, 517, 661, 662
- B**
- bool commands:
 - \bool_gset_false:N 1158, 1177, 1200, 1222, 1238, 1339, 1576, 1612, 2177, 2223, 2801
 - \bool_gset_true:N 1156, 1225, 1337, 1591, 2170, 2176, 2799
 - \bool_if:NTF 57, 659, 1168, 1172, 1188, 1191, 1195, 1206, 1213, 1217, 1229, 1233, 1350, 1355, 1360, 1550, 1595, 1708, 1743, 1853, 1895, 2165, 2180, 2185, 2190, 2806
 - \bool_if:nTF 2399, 2659, 2936
 - \bool_lazy_or:nnTF 1735, 1888
 - \bool_new:N 1159, 1226, 1340, 1592, 2153, 2154, 2795
 - \bool_set_false:N 1718, 1820, 1913, 1977
- box commands:
 - \box_dp:N 216, 218, 266, 268, 323, 325, 372, 374, 376, 378, 2202, 2235, 2236, 2261
 - \box_ht:N 218, 268, 325, 376, 378, 1755, 1950, 2207, 2246, 2247, 2263
 - \box_if_empty:NTF 2296
 - \box_move_down:nn 2127, 2202
 - \box_move_up:nn 2129, 2207, 2808
 - \box_new:N 2012, 2117, 2118
 - \box_set_dp:Nn 1675
 - \box_set_ht:Nn 1674
 - \box_set_wd:Nn 280, 1673
 - \box_use:N 223, 241, 255, 271, 298, 312, 328, 344, 356, 407, 421, 440, 1290, 1485, 1676, 2158
 - \box_wd:N 217, 225, 267, 273, 324, 330, 373, 375, 1754, 1949
- box internal commands:
 - __box_backend_clip:N 205, 260, 317, 361
 - \l__box_backend_cos_fp 275
 - __box_backend_rotate:Nn 227, 275, 332, 411
 - __box_backend_rotate_aux:Nn 227, 275, 332
 - __box_backend_scale:Nnn 244, 303, 347, 424
 - \l__box_backend_sin_fp 275
 - \g__box_clip_path_int 361
- C**
- char commands:
 - \char_set_catcode_space:n 146
- clist commands:
 - \clist_map_function:nN ... 1246, 1370
 - \clist_map_function:nn 1619
- color internal commands:
 - __color_backend:nnn 1044
 - __color_backend_cmyk:w 1045
 - __color_backend_devicen_init:n 893
 - __color_backend_devicen_init:nnn 807, 893
 - __color_backend_devicen_init:w 893
 - __color_backend_fill:n 951, 978, 1008, 1026, 1033
 - __color_backend_fill_cmyk:n 951, 985, 1008, 1033
 - __color_backend_fill_devicen:nn 977, 999, 1025, 1095
 - __color_backend_fill_gray:n 951, 985, 1008, 1033
 - __color_backend_fill_rgb:n 951, 985, 1008, 1033
 - __color_backend_fill_separation:nn 977, 985, 1025, 1095
 - \l__color_backend_fill_tl 621, 631, 959, 974
 - \c__color_backend_main_stack_int 508
 - __color_backend_pickup:N .. 448, 471
 - __color_backend_pickup:w 14, 448, 471
 - __color_backend_reset: 603, 623, 640, 962, 975, 985, 1017, 1042
 - __color_backend_rgb:w 1068
 - __color_backend_select:n 603, 655, 833
 - __color_backend_select:nn . 623, 834
 - __color_backend_select_cmyk:n .. 603, 623, 640
 - __color_backend_select_devicen:nn 654, 827, 833, 943

_color_backend_select_gray:n	603, 623, 640	_color_backend_stroke_rgb:n	951, 1008, 1044
_color_backend_select_rgb:n	603, 623, 640	_color_backend_stroke_rgb:w	1044
_color_backend_select_separation:nn	654, 827, 833, 943	_color_backend_stroke_separation:nn	977, 985, 1025, 1095
_color_backend_separation-init:n	657, 836, 917, 940	\l_color_backend_stroke_tl	621, 632, 961, 972
_color_backend_separation-init:nnn	657	\g_color_model_int	677, 813, 856, 928
_color_backend_separation-init:nnnn	657	\c_color_model_range_CIELAB_tl	768, 803, 877, 884
_color_backend_separation-init:nnnnn	657, 829, 836	color.sc	603, 3107
_color_backend_separation-init:nw	657	cs commands:	
_color_backend_separation-init:w	657	\cs_generate_variant:Nn	49, 53, 56, 91, 130, 135, 162, 193, 199, 553, 590, 670, 1105, 1300, 1494, 1867, 1924, 1940, 2016, 2053, 2112, 2595, 2623, 2733, 2755, 2790, 2998, 3062
_color_backend_separation-init_/DeviceCMYK:nnn	657	\cs_gset:Npx	2941, 2946
_color_backend_separation-init_/DeviceGray:nnn	657	\cs_gset_eq:NN	647, 648, 946, 992, 993, 999, 1001, 1003
_color_backend_separation-init_/DeviceRGB:nnn	657	\cs_gset_protected:Npn	642, 649, 861, 890, 933, 945, 987, 994, 996, 998, 3028, 3067, 3076
_color_backend_separation-init_aux:nnnnn	657	\cs_if_exist:NTF	27, 59, 449, 472, 516, 530, 661, 860, 888, 932, 2292, 2684, 2710, 2796, 3001, 3026
_color_backend_separation-init_CIELAB:nnn	657, 829, 836	\cs_if_exist_use:NTF	38, 683
_color_backend_separation-init_CIELAB:nnnnnn	830	\cs_new:Npn	692, 694, 696, 698, 705, 711, 713, 719, 736, 743, 745, 934, 1251, 1375, 1623, 1953, 1962, 2006, 2031, 2113, 2115, 2148, 2317, 2411, 2412, 2565, 2596, 2597, 2715, 2748, 2791, 2793, 2832, 2949, 2950, 2960, 2965, 2966, 2971, 2972
_color_backend_separation-init_count:n	657	\cs_new:Npx	2432, 2467, 2624, 2635, 2702, 2862
_color_backend_separation-init_count:w	657	\cs_new_eq:NN	46, 656, 835, 940, 981, 982, 1029, 1030, 1097, 1098, 1104, 1299, 1305, 1306, 1493, 1500, 1685, 1714, 1765, 1766, 1808, 1816, 1838, 1909, 1966, 1973, 2005, 2158
_color_backend_separation-init_Device:Nn	657	\cs_new_protected:Npn	47, 51, 54, 64, 70, 75, 77, 81, 92, 102, 111, 120, 133, 136, 138, 140, 160, 165, 174, 184, 194, 205, 227, 229, 244, 260, 275, 277, 303, 317, 332, 334, 347, 361, 411, 424, 448, 466, 471, 479, 509, 544, 554, 566, 580, 591, 603, 605, 607, 609, 617, 623, 625, 627, 629, 636, 654, 671, 761, 807, 827, 828, 829, 830, 833, 836, 862, 866, 893, 951, 953, 955, 957, 964, 966, 968, 970, 977, 979, 1008, 1010, 1012, 1014, 1019, 1021, 1023, 1025,
\g_color_backend_stack_int	508		
\l_color_backend_stack_int	505, 532, 538, 633, 637, 960, 973		
_color_backend_stack_pop:n	542, 580, 637		
_color_backend_stack_push:nn	542, 580, 633, 960, 973		
_color_backend_stroke:n	951, 980, 985		
_color_backend_stroke_cmyk:n	951, 1008, 1044		
_color_backend_stroke_cmyk:w	1044		
_color_backend_stroke_devicen:nn	977, 1003, 1025, 1095		
_color_backend_stroke_gray:n	951, 1008, 1044		
_color_backend_stroke_gray-aux:n	1044		

1027, 1033, 1035, 1037, 1039, 1044,
 1046, 1057, 1065, 1067, 1069, 1095,
 1096, 1106, 1111, 1116, 1118, 1120,
 1128, 1136, 1145, 1155, 1157, 1160,
 1162, 1179, 1184, 1202, 1224, 1227,
 1240, 1253, 1258, 1260, 1262, 1264,
 1266, 1268, 1270, 1272, 1277, 1301,
 1303, 1307, 1312, 1317, 1327, 1336,
 1338, 1341, 1343, 1345, 1347, 1352,
 1357, 1362, 1364, 1377, 1382, 1384,
 1386, 1388, 1390, 1392, 1394, 1396,
 1407, 1432, 1444, 1456, 1468, 1475,
 1495, 1501, 1506, 1511, 1522, 1532,
 1542, 1544, 1546, 1548, 1579, 1581,
 1586, 1588, 1590, 1593, 1614, 1625,
 1638, 1640, 1642, 1644, 1646, 1648,
 1650, 1652, 1654, 1662, 1686, 1700,
 1715, 1727, 1732, 1760, 1772, 1785,
 1795, 1810, 1817, 1825, 1836, 1840,
 1843, 1858, 1868, 1903, 1910, 1916,
 1922, 1925, 1932, 1941, 1946, 1954,
 1967, 1974, 1980, 1982, 1984, 1995,
 2014, 2017, 2019, 2023, 2033, 2054,
 2059, 2064, 2069, 2079, 2084, 2092,
 2120, 2125, 2157, 2159, 2161, 2163,
 2168, 2183, 2188, 2225, 2254, 2273,
 2282, 2319, 2326, 2352, 2357, 2385,
 2397, 2409, 2410, 2413, 2415, 2419,
 2443, 2445, 2447, 2458, 2478, 2488,
 2511, 2525, 2535, 2546, 2567, 2598,
 2646, 2657, 2663, 2691, 2725, 2727,
 2734, 2736, 2740, 2750, 2756, 2761,
 2766, 2771, 2773, 2775, 2783, 2804,
 2819, 2835, 2837, 2860, 2870, 2872,
 2894, 2899, 2932, 2934, 2939, 2944,
 2951, 2953, 2957, 2958, 2959, 2961,
 2962, 2963, 2964, 2967, 2968, 2969,
 2970, 2973, 2974, 2980, 2985, 2990,
 2992, 2994, 3010, 3015, 3030, 3032,
 3038, 3044, 3096, 3098, 3100, 3102
 \cs_new_protected:Npx
 512, 657, 1080, 2674, 2731, 2839
 \cs_set:Npn 144
 \cs_set_eq:NN 2313, 2314
 \cs_set_protected:Npn 451, 474

D

dim commands:

\dim_eval:n 2123, 2355,
 2427, 2428, 2429, 2486, 2521, 2522,
 2523, 2826, 2827, 2828, 2871, 2897
 \dim_max:nn 2233, 2244
 \dim_set:Nn ... 1754, 1755, 1949, 1950
 \dim_to_decimal:n .. 372, 373, 374,

375, 376, 378, 1504, 1509, 1515,
 1516, 1517, 1518, 1527, 1528, 1529,
 1620, 1639, 2000, 2001, 2231, 2242,
 2260, 2261, 2262, 2263, 2267, 2323
 \dim_to_decimal_in_bp:n
 216, 217, 218, 266, 267, 268,
 323, 324, 325, 1124, 1125, 1132,
 1133, 1140, 1141, 1149, 1150, 1151,
 1248, 1252, 1256, 1310, 1315, 1321,
 1322, 1323, 1331, 1332, 1372, 1376,
 1380, 1624, 1691, 1692, 1693, 1694,
 1830, 1831, 1832, 1833, 1882, 1883,
 1884, 1885, 1989, 1990, 1991, 1992

draw internal commands:

__draw_align_currentpoint:... .. 34
 __draw_backend_add_to_path:n ...
 1501, 1547
 __draw_backend_begin:
 1106, 1301, 1495
 __draw_backend_box_use:Nnnnn ...
 30, 1277, 1475, 1662
 __draw_backend_cap_but:
 1240, 1364, 1614
 __draw_backend_cap_rectangle: ..
 1240, 1364, 1614
 __draw_backend_cap_round:
 1240, 1364, 1614
 __draw_backend_clip: 1160, 1341, 1546
 __draw_backend_closepath:
 1160, 1341, 1546
 __draw_backend_closestroke: ...
 1160, 1341, 1546
 __draw_backend_cm:nnnn 1272, 1285,
 1286, 1287, 1396, 1479, 1654, 1665
 __draw_backend_cm_aux:nnnn .. 1396
 __draw_backend_cm_decompose:nnnnN
 1402, 1431
 __draw_backend_cm_decompose_-
 auxi:nnnnN 1431
 __draw_backend_cm_decompose_-
 auxii:nnnnN 1431
 __draw_backend_cm_decompose_-
 auxiii:nnnnN 1431
 __draw_backend_curveto:nnnnnn ..
 1120, 1307, 1501
 __draw_backend_dash:n
 1240, 1364, 1614
 __draw_backend_dash_aux:nn .. 1614
 __draw_backend_dash_pattern:nn ..
 1240, 1364, 1614
 __draw_backend_discardpath: ...
 1160, 1341, 1546
 __draw_backend_end: 1106, 1301, 1495

<code>__draw_backend_evenodd_rule: . . .</code>	<code>\g__draw_draw_path_tl</code>
<code> 1155, 1336, 1542</code>	<code> . . 1501, 1557, 1573, 1575, 1602, 1611</code>
<code>__draw_backend_fill: 1160, 1341, 1546</code>	<code>\g__draw_path_int 1561, 1578</code>
<code>__draw_backend_fillstroke:</code>	
<code> 1160, 1341, 1546</code>	E
<code>__draw_backend_join_bevel:</code>	<code>\endlandscape 2800</code>
<code> 1240, 1364, 1614</code>	<code>\errmessage 38</code>
<code>__draw_backend_join_miter:</code>	<code>\evensidemargin 2200</code>
<code> 1240, 1364, 1614</code>	exp commands:
<code>__draw_backend_join_round:</code>	<code>\exp_after:wN 151, 457, 1960</code>
<code> 1240, 1364, 1614</code>	<code>\exp_args:Ne 707, 2354, 2896</code>
<code>__draw_backend_lineto:nn</code>	<code>\exp_args:Nf 1245, 1369, 2122</code>
<code> 1120, 1307, 1501</code>	<code>\exp_args:NNf 228, 276, 333</code>
<code>__draw_backend_linewidth:n</code>	<code>\exp_args:Nnx 2109, 2786</code>
<code> 1240, 1364, 1614</code>	<code>\exp_args:NV 453</code>
<code>__draw_backend_literal:n</code>	<code>\exp_args:Nx 1778, 1799,</code>
<code> 1104, 1109, 1113, 1117,</code>	<code>2066, 2081, 2196, 2752, 2982, 3012</code>
<code>1119, 1122, 1130, 1138, 1147, 1161,</code>	<code>\exp_last_unbraced:Nx 462, 476</code>
<code>1164, 1165, 1166, 1167, 1170, 1176,</code>	<code>\exp_not:N . 514, 515, 517, 518, 523,</code>
<code>1186, 1193, 1199, 1204, 1209, 1210,</code>	<code>525, 662, 665, 2434, 2436, 2439,</code>
<code>1211, 1212, 1215, 1221, 1231, 1237,</code>	<code>2469, 2471, 2474, 2626, 2628, 2631,</code>
<code>1242, 1255, 1259, 1261, 1263, 1265,</code>	<code>2637, 2639, 2642, 2679, 2680, 2686,</code>
<code>1267, 1269, 1271, 1274, 1279, 1280,</code>	<code>2687, 2706, 2711, 2843, 2851, 2867</code>
<code>1281, 1282, 1283, 1284, 1288, 1289,</code>	<code>\exp_not:n . . 48, 89, 100, 128, 2057,</code>
<code>1291, 1292, 1293, 1294, 1295, 1299,</code>	<code>2062, 2348, 2581, 2582, 2596, 2597,</code>
<code>1309, 1314, 1319, 1329, 1342, 1344,</code>	<code>2609, 2610, 2764, 2769, 2780, 2876</code>
<code>1346, 1349, 1354, 1359, 1363, 1366,</code>	<code>\ExplBackendFileDate 1</code>
<code>1379, 1383, 1385, 1387, 1389, 1391,</code>	
<code>1393, 1395, 1493, 1553, 1572, 1598</code>	F
<code>__draw_backend_miterlimit:n . . .</code>	file commands:
<code> 1240, 1364, 1614</code>	<code>\file_compare_timestamp:nNnTF . 1787</code>
<code>__draw_backend_moveto:nn</code>	<code>\file_parse_full_name:nNNN 1774, 1797</code>
<code> 1120, 1307, 1501</code>	fp commands:
<code>__draw_backend_nonzero_rule: . . .</code>	<code>\fp_compare:nNnTF</code>
<code> 1155, 1336, 1542</code>	<code> . 235, 282, 288, 340, 1412, 1425, 1470</code>
<code>__draw_backend_path:n 1546</code>	<code>\fp_eval:n . 228, 237, 250, 251, 276,</code>
<code>__draw_backend_rectangle:nnnn . .</code>	<code>293, 308, 310, 333, 342, 353, 354,</code>
<code> 1120, 1307, 1501</code>	<code>418, 433, 434, 1052, 1053, 1054,</code>
<code>__draw_backend_scope:n 1543, 1545,</code>	<code>1062, 1075, 1076, 1077, 1414, 1419,</code>
<code>1565, 1605, 1627, 1639, 1641, 1643,</code>	<code>1420, 1427, 1437, 1438, 1439, 1440,</code>
<code>1645, 1647, 1649, 1651, 1653, 1656</code>	<code>1449, 1450, 1451, 1452, 1461, 1462,</code>
<code>__draw_backend_scope_begin: . . .</code>	<code>1463, 1464, 2345, 2508, 2890, 2983,</code>
<code> 1116, 1302, 1305</code>	<code>2991, 2993, 3013, 3035, 3042, 3103</code>
<code>__draw_backend_scope_end:</code>	<code>\fp_new:N 301, 302</code>
<code> 1116, 1304, 1305</code>	<code>\fp_set:Nn 281, 284</code>
<code>__draw_backend_stroke:</code>	<code>\fp_use:N 287, 291, 296</code>
<code> 1160, 1341, 1546</code>	<code>\fp_zero:N 283</code>
<code>\g__draw_clip_path_int</code>	<code>\c_zero_fp 235, 282, 288, 340, 1412, 1425</code>
<code> . . 1552, 1555, 1568, 1597, 1600, 1608</code>	
<code>\g__draw_draw_clip_bool . . 1160, 1546</code>	G
<code>\g__draw_draw_eor_bool</code>	graphics commands:
<code> . . . 1155, 1172, 1188, 1195, 1206,</code>	<code>\graphics_bb_restore:nTF . 1729, 1943</code>
<code>1217, 1233, 1336, 1350, 1355, 1360</code>	<code>\graphics_bb_save:n 1758, 1951</code>
<code>\g__draw_draw_path_int 1546</code>	<code>\l_graphics_decodearray_tl</code>
	<code> 1706, 1707,</code>

1717, 1737, 1741, 1742, 1819, 1851,
 1852, 1890, 1893, 1894, 1912, 1976
 \graphics_extract_bb:n
 1814, 1821, 1971, 1978
 \l_graphics_interpolate_bool ...
 1708, 1718, 1736, 1743,
 1820, 1853, 1889, 1895, 1913, 1977
 \l_graphics_llx_dim
 1691, 1830, 1882, 1989
 \l_graphics_lly_dim
 1692, 1831, 1883, 1990
 \l_graphics_name_tl 1792
 \l_graphics_page_int
 1702, 1722, 1723, 1747,
 1748, 1812, 1849, 1850, 1876, 1877,
 1905, 1918, 1919, 1958, 1959, 1969
 \l_graphics_pagebox_tl
 51, 1703, 1721,
 1749, 1750, 1813, 1847, 1848, 1878,
 1880, 1906, 1927, 1928, 1960, 1970
 \graphics_read_bb:n . 1685, 1808, 1966
 \l_graphics_urx_dim
 .. 1693, 1754, 1832, 1884, 1949, 1991
 \l_graphics_ury_dim .. 1694, 1755,
 1833, 1885, 1950, 1992, 2000, 2001
 graphics internal commands:
 \l__graphics_backend_dir_str . 1767
 \l__graphics_backend_ext_str . 1767
 __graphics_backend_getbb_auxi:n
 1700
 __graphics_backend_getbb_-
 auxi:nN 1903
 __graphics_backend_getbb_-
 auxii:n 1700
 __graphics_backend_getbb_-
 auxii:nnN 1903
 __graphics_backend_getbb_-
 auxiii:nNnn 1903
 __graphics_backend_getbb_-
 auxiv:nnNnn 1903
 __graphics_backend_getbb_-
 auxv:nNnn 1903
 __graphics_backend_getbb_-
 auxvi:nNnn 1944, 1946
 __graphics_backend_getbb_eps:n .
 1685, 1767, 1808, 1966
 __graphics_backend_getbb_eps:nm
 1767
 __graphics_backend_getbb_eps:nn
 1778, 1785
 __graphics_backend_getbb_jpg:n .
 1700, 1808, 1903, 1967
 __graphics_backend_getbb_-
 pagebox:w 1903, 1960
 __graphics_backend_getbb_pdf:n .
 1700, 1793, 1808, 1903, 1974
 __graphics_backend_getbb_png:n .
 1700, 1808, 1903, 1967
 __graphics_backend_include:nn 1980
 __graphics_backend_include_-
 auxi:nn 1825
 __graphics_backend_include_-
 auxii:nnn 1825
 __graphics_backend_include_-
 auxiii:nnn 1825
 __graphics_backend_include_-
 bitmap_quote:w 1954, 1995
 __graphics_backend_include_-
 eps:n 1686, 1767, 1825, 1980
 __graphics_backend_include_-
 jpg:n 1760, 1825, 1995
 __graphics_backend_include_-
 pdf:n .. 1760, 1799, 1825, 1954, 1980
 __graphics_backend_include_pdf_-
 quote:w 1957, 1962
 __graphics_backend_include_-
 png:n 1760, 1825, 1995
 \l__graphics_backend_name_str . 1767
 \l__graphics_graphics_attr_tl ...
 1699, 1704,
 1711, 1719, 1729, 1756, 1758, 1763
 \l__graphics_internal_box
 .. 1752, 1754, 1755, 1948, 1949, 1950
 \g__graphics_track_int
 1824, 1870, 1871
 group commands:
 \group_begin: 143, 171, 190
 \group_end: 156, 179
 \group_insert_after:N 615, 634, 645,
 962, 975, 990, 1017, 1042, 3024, 3059

H

 hbox commands:
 \hbox:n 2128, 2131,
 2203, 2209, 2362, 2369, 2904, 2915
 \hbox_overlap_right:n 223,
 255, 271, 312, 328, 356, 440, 1290, 1485
 \hbox_set:Nn .. 1752, 1948, 2195, 2227
 \hbox_set:Nw 2178
 \hbox_set_end: 2193
 \hbox_unpack:N 2314

I

 int commands:
 \int_compare:nNnTF
 508, 542, 640, 943, 985, 1722, 1747,
 1849, 1876, 1918, 1958, 2286, 2387,
 2677, 2705, 2841, 2848, 2864, 3065

<code>\int_const:Nn</code>	149, 155, 515, 568, 1756, 1871, 2026, 2555, 2743
<code>\int_eval:n</code>	549, 559, 588, 599, 703, 712, 725, 727, 731, 744, 2655, 2680, 2687, 2700, 2933, 2941, 2946
<code>\int_gincr:N</code>	197, 363, 514, 1552, 1597, 1870, 2025, 2094, 2138, 2212, 2742, 2785, 2821, 2843
<code>\int_gset:Nn</code>	172, 191, 2275
<code>\int_gset_eq:NN</code>	180, 2139, 2213, 2822
<code>\int_if_exist:NTF</code>	1860
<code>\int_if_odd:nTF</code>	2198
<code>\int_new:N</code>	163, 164, 410, 505, 511, 1578, 1824, 2021, 2119, 2150, 2152, 2738, 2803, 2834
<code>\int_set:Nn</code>	532
<code>\int_set_eq:NN</code>	168, 187, 538, 2287
<code>\int_step_function:nnnN</code>	729
<code>\int_use:N</code>	365, 396, 523, 533, 677, 813, 856, 928, 1555, 1561, 1568, 1600, 1608, 1723, 1748, 1763, 1850, 1863, 1875, 1877, 1959, 2032, 2097, 2110, 2114, 2142, 2149, 2217, 2318, 2566, 2576, 2749, 2787, 2792, 2825, 2833, 2851, 2867
<code>\int_value:w</code>	2434, 2469, 2626, 2637, 2655
<code>\int_zero:N</code>	1702, 1812, 1905, 1969
K	
kernel internal commands:	
<code>__kernel_backend_align_begin: . .</code>	64, 208, 232, 247
<code>__kernel_backend_align_end: . .</code>	64, 222, 240, 254
<code>\g__kernel_backend_header_bool . .</code>	57, 659
<code>__kernel_backend_literal:n .</code> 46, 52, 55, 62, 66, 73, 76, 78, 134, 137, 139, 141, 161, 337, 350, 520, 546, 556, 611, 618, 644, 650, 673, 809, 989, 995, 997, 1016, 1041, 1108, 1114, 1409, 1416, 1422, 1482, 1487, 1688, 1827, 1862, 1872, 1986, 1997, 2732, 2871, 2933, 2937, 2942, 2947	
<code>__kernel_backend_literal_page:n</code>	92, 136, 2726, 2728, 2952, 2954
<code>__kernel_backend_literal_pdf:n .</code>	81, 133, 263, 320, 1299, 3074, 3089
<code>__kernel_backend_literal_- postscript:n</code>	51, 67, 68, 72, 209, 210, 212, 213, 221, 233, 248, 1104, 2389, 2401
<code>__kernel_backend_literal_svg:n .</code>	160, 167, 178, 186, 196, 364, 366, 383, 1493, 1666, 1677
<code>__kernel_backend_matrix:n</code>	120, 285, 306, 1399
<code>__kernel_backend_postscript:n . .</code>	54, 613, 1020, 1022, 1024, 1028, 2015, 2071, 2086, 2128, 2134, 2171, 2203, 2210, 2214, 2228, 2256, 2300, 2307, 2313, 2321, 2328, 2362, 2369, 2987, 2996
<code>__kernel_backend_scope:n</code>	165, 393, 398, 1082, 1498, 3103
<code>__kernel_backend_scope_begin: . .</code>	75, 102, 138, 165, 207, 231, 246, 262, 279, 305, 319, 336, 349, 1305, 1477, 1497, 1664
<code>__kernel_backend_scope_begin:n .</code>	165, 385, 413, 426
<code>__kernel_backend_scope_end: . . .</code>	75, 102, 138, 165, 224, 242, 256, 272, 299, 313, 329, 345, 357, 408, 422, 441, 1306, 1489, 1500, 1678
<code>\g__kernel_backend_scope_int . . .</code>	163, 170, 172, 177, 181, 189, 191, 197
<code>\l__kernel_backend_scope_int . . .</code>	163, 169, 182, 188
<code>__kernel_color_backend_stack_- init:Nnn</code>	508, 566, 3003
<code>__kernel_dependency_version_- check:Nn</code>	1
<code>__kernel_dependency_version_- check:nn</code>	27, 29
<code>__kernel_kern:n</code>	2133, 2135, 2361, 2365, 2368, 2372, 2903, 2911, 2914, 2930
<code>\c__kernel_sys_dvipdfmx_version_- int</code>	143, 508, 542, 640, 943, 985, 2841, 2848, 2864, 3065
L	
<code>\landscape</code>	2796, 2798
M	
math commands:	
<code>\c_math_toggle_token</code>	2181, 2191
<code>\MessageBreak</code>	40
mode commands:	
<code>\mode_if_horizontal:TF</code>	2277, 2284
<code>\mode_if_math:TF</code>	2175
O	
<code>\oddsidemargin</code>	2199
opacity internal commands:	
<code>__opacity_backend:nn</code>	2990, 3096

<code>__opacity_backend_fill:n</code>	<code>__pdf_backend_destination:nn</code> ...
..... 2990 , 3032 , 3096 2326 , 2488 , 2872
<code>__opacity_backend_fill_stroke:nn</code>	<code>__pdf_backend_destination:nnnn</code> .
..... 3034 , 3040 , 3044 , 3062 , 3076 2326 , 2488 , 2872
<code>\l__opacity_backend_fill_tl</code>	<code>__pdf_backend_destination_</code>
.. 3008 , 3017 , 3041 , 3049 , 3069 , 3081	aux:nnnn 2326 , 2872
<code>__opacity_backend_fillstroke:nn</code>	<code>__pdf_backend_emc:</code>
..... 3032 2413 , 2725 , 2951 , 2973
<code>__opacity_backend_reset:</code> 3010 , 3059	<code>__pdf_backend_info_gput:nn</code>
<code>__opacity_backend_select:n</code> 2017 , 2525 , 2734 , 2957
..... 2980 , 3010 , 3065 , 3096	<code>__pdf_backend_link:nw</code> 2159
<code>__opacity_backend_select_aux:n</code> .	<code>__pdf_backend_link_aux:nw</code> ... 2159
..... 2980 , 3010 , 3047 , 3067 , 3079	<code>__pdf_backend_link_begin:n</code> .. 2835
<code>\c__opacity_backend_stack_int</code> ...	<code>__pdf_backend_link_begin:nnnw</code> 2443
..... 3001 , 3022 , 3031 , 3057	<code>__pdf_backend_link_begin:nw</code> ...
<code>__opacity_backend_stack_pop:n</code> 3031 2160 , 2162 , 2163
<code>__opacity_backend_stack_push:nn</code>	<code>__pdf_backend_link_begin_aux:nw</code>
..... 3022 , 3057 2166 , 2168
<code>__opacity_backend_stroke:n</code>	<code>__pdf_backend_link_begin_</code>
..... 2990 , 3032 , 3096	goto:nnw 2159 , 2443 , 2835
<code>\l__opacity_backend_stroke_tl</code> ...	<code>__pdf_backend_link_begin_</code>
.. 3008 , 3018 , 3036 , 3050 , 3070 , 3082	user:nnw 2159 , 2443 , 2835
P	
pdf commands:	
<code>\pdf_object_if_exist:nTF</code> 868	<code>\g__pdf_backend_link_bool</code>
<code>\pdf_object_last:</code> .. 850 , 857 , 922 , 929 2154 , 2165 , 2170 , 2185 , 2223
<code>\pdf_object_new:nn</code> 870	<code>\g__pdf_backend_link_dict_tl</code> ...
<code>\pdf_object_now:nn</code> 2151 , 2173 , 2218
..... 838 , 860 , 864 , 888 , 895 , 932	<code>__pdf_backend_link_end:</code>
<code>\pdf_object_ref:n</code> 883 2159 , 2443 , 2835
<code>\pdf_object_write:nn</code> 871	<code>__pdf_backend_link_end_aux:</code> . 2159
pdf internal commands:	<code>\g__pdf_backend_link_int</code>
<code>__pdf_backend:n</code> 2731 , 2150 , 2213 ,
2735 , 2737 , 2763 , 2768 , 2777 , 2823 ,	2217 , 2318 , 2834 , 2843 , 2851 , 2867
2845 , 2861 , 2874 , 2906 , 2907 , 2917	<code>__pdf_backend_link_last:</code>
<code>__pdf_backend_annotation:nnnn</code> 2317 , 2467 , 2862
..... 2120 , 2419 , 2804	<code>__pdf_backend_link_margin:n</code> ...
<code>__pdf_backend_annotation_</code> 2319 , 2478 , 2870
aux:nnnn 2122 , 2125 , 2804	<code>\g__pdf_backend_link_math_bool</code> ..
<code>\g__pdf_backend_annotation_int</code> 2153 , 2176 , 2177 , 2180 , 2190
.. 2119 , 2139 , 2149 , 2803 , 2822 , 2833	<code>__pdf_backend_link_minima:</code> .. 2159
<code>__pdf_backend_annotation_last:</code> .	<code>__pdf_backend_link_outertbox:n</code> 2159
..... 2148 , 2432 , 2832	<code>\g__pdf_backend_link_sf_int</code>
<code>__pdf_backend_bdc:nn</code> 2152 , 2275 , 2286 , 2287
..... 2413 , 2725 , 2951 , 2973	<code>__pdf_backend_link_sf_restore:</code> 2159
<code>__pdf_backend_catalog_gput:nn</code> ..	<code>__pdf_backend_link_sf_save:</code> . 2159
..... 2017 , 2525 , 2734 , 2957	<code>\l__pdf_backend_model_box</code> . 2118 ,
<code>__pdf_backend_compress_objects:n</code>	2195 , 2227 , 2235 , 2246 , 2261 , 2263
..... 2385 , 2646 , 2932 , 2967	<code>__pdf_backend_objcompresslevel:n</code>
<code>__pdf_backend_compresslevel:n</code> 2646
..... 2385 , 2646 , 2932 , 2967	<code>\g__pdf_backend_object_int</code>
<code>\l__pdf_backend_content_box</code> 2117 , 2021 , 2025 , 2028 ,
2178 , 2202 , 2205 , 2207 , 2236 , 2247	2094 , 2097 , 2110 , 2114 , 2138 , 2139 ,
	2142 , 2212 , 2213 , 2738 , 2742 , 2745 ,
	2785 , 2787 , 2792 , 2821 , 2822 , 2825

_pdf_backend_object_last:	pdf.bordertracking.continue	3192
..... 2113, 2624, 2791, 2959	pdf.bordertracking.end	3192
_pdf_backend_object_new:nn . . .	pdf.bordertracking.endpage	3192
..... 2023, 2546, 2740, 2959	pdf.breaklink	3330
_pdf_backend_object_now:nn . . .	pdf.breaklink.write	3330
..... 2092, 2598, 2783, 2959	pdf.brokenlink.dict	3192
_g_pdf_backend_object_prop	pdf.brokenlink.rect	3192
..... 2021, 2029, 2040, 2050,	pdf.brokenlink.skip	3192
2545, 2563, 2579, 2738, 2746, 2753	pdf.count	3330
_pdf_backend_object_ref:n 2023,	pdf.currentrect	3330
2037, 2051, 2546, 2740, 2759, 2959	pdf.cvs	3114
_pdf_backend_object_write:nn . .	pdf.dest.anchor	3157
..... 2033, 2567, 2750, 2959	pdf.dest.point	3157
_pdf_backend_object_write:nnn 2750	pdf.dest.x	3157
_pdf_backend_object_write_-	pdf.dest.y	3157
array:nn 2033, 2750	pdf.dest2device	3157
_pdf_backend_object_write_-	pdf.dev.x	3157
dict:nn 2033, 2750	pdf.dev.y	3157
_pdf_backend_object_write_-	pdf.dvi.pt	3114
fstream:nn 2033, 2750	pdf.globaldict	3111
_pdf_backend_object_write_-	pdf.leftboundary	3192
fstream:nnn 2067, 2069	pdf.link.dict	2159
_pdf_backend_object_write_-	pdf.linkdp.pad	2159, 3118
stream:nn 2033, 2750	pdf.linkht.pad	2159, 3118
_pdf_backend_object_write_-	pdf.linkmargin	3118
stream:nnn 2033	pdf.llx	2159, 3121
_pdf_backend_object_write_-	pdf.lly	2159, 3121
stream:nnnn 2750	pdf.originx	3192
_pdf_backend_pageobject_ref:n .	pdf.originy	3192
..... 2115, 2635, 2793, 2959	pdf.outerbox	2159, 3434
_pdf_backend_pdfmark:n	pdf.pdfmark	3434
2014, 2018, 2020, 2035, 2056, 2061,	pdf.pdfmark.dict	3434
2095, 2140, 2329, 2373, 2414, 2416	pdf.pdfmark.good	3434
_pdf_backend_version_major: . .	pdf.pt.dvi	3114
.. 2411, 2702, 2941, 2942, 2949, 2971	pdf.rect	3121
_pdf_backend_version_major_-	pdf.rect.ht	3114
gset:n 2409, 2674, 2939, 2969	pdf.rightboundary	3192
_pdf_backend_version_minor: . .	pdf.save.linkll	3121
.. 2411, 2702, 2946, 2947, 2949, 2971	pdf.save.linkur	3121
_pdf_backend_version_minor_-	pdf.save.ll	3121
gset:n 2409, 2674, 2939, 2969	pdf.save.ur	3121
_pdf_breaklink_pdfmark_tl . . .	pdf.tmpa	3157
..... 2155, 2220, 2312	pdf.tmpb	3157
_pdf_breaklink_postscript:n . .	pdf.tmpc	3157
..... 2157, 2204, 2206, 2313	pdf.tmpd	3157
_pdf_breaklink_usebox:N	pdf.urx	3121
..... 2158, 2205, 2314	pdf.ury	2159, 3121
_pdf_exp_not_i:nn . 2567, 2613, 2618	pdfmanagement commands:	
_pdf_exp_not_ii:nn 2567, 2614, 2619	_pdfmanagement_add:nnn	
_pdf_internal_box 2012 854, 926, 3001, 3005, 3019,	
_pdf_landscape_bool . . . 2795, 2806	3026, 3051, 3054, 3071, 3083, 3086	
pdf.baselineskip 2159, 3434	prg commands:	
pdf.bordertracking 3192	_prg_replicate:nn	
pdf.bordertracking.begin 3192 176, 701, 722, 732, 901	

<code>\tl_gclear:N</code>	1575, 1611	2032, 2556, 2566, 2577, 2744, 2749
<code>\tl_gset:Nn</code>	1534, 2173	<code>\tl_use:N</code> 800, 876
<code>\tl_if_blank:nTF</code>		
	525, 576, 716, 733, 740, 758, 842, 937	
<code>\tl_if_empty:N</code>	1537, 1706, 1741,	
	1749, 1847, 1851, 1878, 1893, 1927	
<code>\tl_if_empty:nTF</code>	1631	
<code>\tl_if_empty_p:N</code>	1737, 1890	
<code>\tl_if_head_is_space:nTF</code>	453	
<code>\tl_new:N</code>	621,	
	622, 1541, 1699, 2151, 2155, 3008, 3009	
<code>\tl_put_left:Nn</code>	2800	
<code>\tl_put_right:Nn</code>	2294, 2798	
<code>\tl_set:Nn</code>	455, 467, 483, 486, 489,	
	493, 496, 631, 632, 959, 972, 1704,	
	1719, 1792, 2156, 2312, 3017, 3018,	
	3049, 3050, 3069, 3070, 3081, 3082	
<code>\tl_to_str:n</code>	2027,	

U

use commands:

<code>\use:N</code>	43, 2049, 2109, 2758, 2786
<code>\use:n</code>	61, 457, 493, 518, 663,
	852, 924, 1049, 1059, 1072, 1245,
	1369, 1434, 1446, 1458, 1616, 1934
<code>\use_none:n</code>	1631, 1633, 2290

V

<code>\value</code>	2198
vbox commands:	
<code>\vbox:n</code>	2810
<code>\vbox_set:Nn</code>	2298
<code>\vbox_to_zero:n</code>	2359, 2366, 2901, 2912
<code>\vbox_unpack_drop:N</code>	2306