

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}{2022-02-07}{ }
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2022-02-07}{ }
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2022-02-07}{ }
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2022-02-07}{ }
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2022-02-07}{ }
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2022-02-07}{ }
25   {L3 backend support: XeTeX}
26 </xetex>
```

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

```
27 \cs_if_exist:NTF \_kernel_dependency_version_check:nn
28   {
29     \_kernel_dependency_version_check:nn {2021-02-18}
30     <dvipdfmx>    {l3backend-dvipdfmx.def}
31     <dvips>       {l3backend-dvips.def}
32     <dvisvgm>     {l3backend-dvisvgm.def}
33     <luatex>      {l3backend-luatex.def}
34     <pdftex>      {l3backend-pdftex.def}
35     <xetex>       {l3backend-xetex.def}
```

```

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

```

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

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

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

```

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

```

(End definition for `__kernel_backend_literal:e`.)

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

```

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

```

(End definition for `__kernel_backend_first_shipout:n`.)

1.1 dvips backend

```

60 <*dvips>

```

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

```

61 \cs_new_protected:Npn __kernel_backend_literal_postscript:n #1
62 { __kernel_backend_literal:n { ps:: #1 } }
63 \cs_generate_variant:Nn __kernel_backend_literal_postscript:n { x }

```

(End definition for `_kernel_backend_literal_postscript:n`.)

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

`_kernel_backend_postscript:x`

```

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

```

(End definition for `_kernel_backend_postscript:n`.)

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

```

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

```

`_kernel_backend_align_begin:`

`_kernel_backend_align_end:`

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

```

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

```

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

`_kernel_backend_scope_begin:`

`_kernel_backend_scope_end:`

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

```

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

```

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

```

87 </dvips>

```

1.2 LuaTeX and pdfTeX backends

88 `<*luatex | pdftex>`

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

_kernel_backend_literal_pdf:n
_kernel_backend_literal_pdf:x

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

```
89 \cs_new_protected:Npn \_kernel_backend_literal_pdf:n #1
90 {
91   <*luatex>
92     \tex_pdfextension:D literal
93   </luatex>
94   <*pdftex>
95     \tex_pdfliteral:D
96   </pdftex>
97   { \exp_not:n {#1} }
98 }
99 \cs_generate_variant:Nn \_kernel_backend_literal_pdf:n { x }
```

(End definition for _kernel_backend_literal_pdf:n.)

_kernel_backend_literal_page:n

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

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

(End definition for _kernel_backend_literal_page:n.)

_kernel_backend_scope_begin:

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

_kernel_backend_scope_end:

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

```

126 </pdfTeX>
127 }

```

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

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

```

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

```

(End definition for `_kernel_backend_matrix:n.`)

```

139 </luatex | pdfTeX>

```

1.3 dvipdfmx backend

```

140 <*dvipdfmx | xetex>

```

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

```

\_kernel_backend_literal_pdf:n
\_kernel_backend_literal_pdf:x

```

```

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

```

(End definition for `_kernel_backend_literal_pdf:n.`)

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

```

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

```

(End definition for `_kernel_backend_literal_page:n.`)

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

```

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

```

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

```

150 <@@=sys>

```

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

```

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

```

(End definition for `\c__kernel_sys_dvipdfmx_version_int`.)

```

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

```

1.4 dvisvgm backend

```

167 <*dvisvgm>

```

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

```

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

```

(End definition for `__kernel_backend_literal_svg:n`.)

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

```

171 \int_new:N \g__kernel_backend_scope_int
172 \int_new:N \l__kernel_backend_scope_int

```

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

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

```

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

```

```

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

```

(End definition for __kernel_backend_scope_begin: and others.)

```

208 </dvisvgm>
209 </package>

```

2 l3backend-box Implementation

```

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

```

2.1 dvips backend

```

212 <*dvips>

```

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

```

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

```

```

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

```

(End definition for __box_backend_clip:N.)

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

```

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

```

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

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

```

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

```



```

264     \__kernel_backend_scope_end:
265 }

(End definition for \__box_backend_scale:Nnn.)

266 </dvips>

```

2.2 LuaTeX and pdfTeX backends

```

267 <*luatex | pdftex>

```

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

```

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

```

(End definition for `__box_backend_clip:N`.)

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

```

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

```

```

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

```

(End definition for __box_backend_rotate:Nn and others.)

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

```

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

```

(End definition for __box_backend_scale:Nnn.)

323 </luatex | pdftex>

2.3 dvipdfmx/X_YTeX backend

324 <*dvipdfmx | xetex>

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

```

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

```

(End definition for `_box_backend_clip:N`.)

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

```

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

```

(End definition for `_box_backend_rotate:Nn` and `_box_backend_rotate_aux:Nn`.)

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

```

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

```

(End definition for `_box_backend_scale:Nnn`.)

367 `</dvipdfmx | xetex>`

2.4 dvisvgm backend

368 `<*dvisvgm>`

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

down using scopes to allow for the depth of the \TeX box and keep the reference point the same!

```

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

```

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

```

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

```

```

415     \box_use:N #1
416     \__kernel_backend_scope_end:
417   }
418   \int_new:N \g__kernel_clip_path_int
(End definition for \__box_backend_clip:N and \g__kernel_clip_path_int.)

```

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

```

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

```

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

```

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

```

3 l3backend-color Implementation

```

453 <*package>
454 <@@=color>

```

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

3.1 Collecting information from L^AT_EX 2_ε

3.1.1 dvips-style

```

455 <*dvisvgm | dvipdfmx | dvips | xetex>

```

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

```

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

```

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

```

477 </dvisvgm | dvipdfmx | dvips | xetex>

```

3.1.2 Lua_T_EX and pdf_T_EX

```

478 <*luatex | pdftex>

```

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

```

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

```

```

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

```

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

511 `</luatex | pdftex>`

3.2 The color stack

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

3.2.1 Common code

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

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

513 `\int_new:N \l__color_backend_stack_int`

(End definition for `\l__color_backend_stack_int`.)

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

3.2.2 dvipdfmx/X_YTeX

515 $\langle *dvipdfmx | xetex \rangle$

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

```

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

```

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

```

551 \cs_gset_protected:Npn \__kernel_backend_scope_end:
552 {
553   \__kernel_backend_literal:n { x:grestore }
554   \__kernel_backend_literal:x
555   {
556     pdfcolorstack ~
557     \int_use:N \g__color_backend_stack_int \c_space_tl current
558   }
559 }
560 }

```


(End definition for `_kernel_color_backend_stack_init:Nnn`, `\g_color_backend_stack_int`, and `\c_color_backend_main_stack_int`.)

Simple enough but needs a version check.

```

561 \int_compare:nNnF \c\_kernel_sys_dvipdfmx_version_int < { 20201111 }
562 {
563   \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
564   {
565     \__kernel_backend_literal:x
566     {
567       pdfcolorstack ~
568       \int_eval:n {#1} ~
569       push ~ (#2)
570     }
571   }
572   \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
573   \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
574   {
575     \__kernel_backend_literal:x
576     {
577       pdfcolorstack ~
578       \int_eval:n {#1} ~
579       pop
580     }
581   }
582 }

```

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

583 `</dvipdfmx | xetex>`

3.2.3 LuaTeX and pdfTeX

584 `<*luatex | pdftex>`

```

585 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
586 {
587   \int_const:Nn #1
588   {
589     <*luatex>
590     \tex_pdffeedback:D colorstackinit ~
591     </luatex>
592     <*pdftex>
593     \tex_pdfcolorstackinit:D
594     </pdftex>
595     \tl_if_blank:nF {#2} { #2 ~ }
596     {#3}
597   }
598 }

```

(End definition for `__kernel_color_backend_stack_init:Nnn`.)

```

\__kernel_color_backend_stack_push:nn
\__kernel_color_backend_stack_push:nx
\__kernel_color_backend_stack_pop:n

599 \cs_new_protected:Npn \__kernel_color_backend_stack_push:nn #1#2
600 {
601   \*luatex
602     \tex_pdfextension:D colorstack ~
603   \*pdfTeX
604     \tex_pdfcolorstack:D
605   \int_eval:n {#1} ~ push ~ {#2}
606 }
607 \cs_generate_variant:Nn \__kernel_color_backend_stack_push:nn { nx }
608 \cs_new_protected:Npn \__kernel_color_backend_stack_pop:n #1
609 {
610   \*luatex
611     \tex_pdfextension:D colorstack ~
612   \*pdfTeX
613     \tex_pdfcolorstack:D
614   \int_eval:n {#1} ~ pop \scan_stop:
615 }
616
(End definition for \__kernel_color_backend_stack_push:nn and \__kernel_color_backend_stack_
pop:n.)

620 \*luatex | pdfTeX

```

3.3 General color

3.3.1 dvips-style

```

621 \*dvips | dvisvgm

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_select:n
\__color_backend_reset:
color.sc

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

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

637 \*dvips | dvisvgm

```

3.3.2 LuaTeX and pdfTeX

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

\backslash l__color_backend_fill_tl
 \backslash l__color_backend_stroke_tl

639 \backslash tl_new:N \backslash l__color_backend_fill_tl
640 \backslash tl_new:N \backslash l__color_backend_stroke_tl

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

\backslash _color_backend_select_cmyk:n
 \backslash _color_backend_select_gray:n
 \backslash _color_backend_select_rgb:n
 \backslash _color_backend_select:nn
 \backslash _color_backend_reset:

Store the values then pass to the stack.

641 \backslash cs_new_protected:Npn \backslash _color_backend_select_cmyk:n #1
642 { \backslash _color_backend_select:nn { #1 ~ k } { #1 ~ K } }
643 \backslash cs_new_protected:Npn \backslash _color_backend_select_gray:n #1
644 { \backslash _color_backend_select:nn { #1 ~ g } { #1 ~ G } }
645 \backslash cs_new_protected:Npn \backslash _color_backend_select_rgb:n #1
646 { \backslash _color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
647 \backslash cs_new_protected:Npn \backslash _color_backend_select:nn #1#2
648 {
649 \backslash tl_set:Nn \backslash l__color_backend_fill_tl {#1}
650 \backslash tl_set:Nn \backslash l__color_backend_stroke_tl {#2}
651 \backslash _kernel_color_backend_stack_push:nn \backslash l__color_backend_stack_int { #1 ~ #2 }
652 }
653 \backslash cs_new_protected:Npn \backslash _color_backend_reset:
654 { \backslash _kernel_color_backend_stack_pop:n \backslash l__color_backend_stack_int }

(End definition for \backslash _color_backend_select_cmyk:n and others.)

655 \langle /dvipdfmx | luatex | pdftex | xetex \rangle

3.3.3 dvipdfmx/X_YTeX

656 \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).

\backslash _color_backend_select_cmyk:n
 \backslash _color_backend_select_gray:n
 \backslash _color_backend_select_rgb:n
 \backslash _color_backend_reset:

Push the data to the stack.

657 \backslash int_compare:nNnT \backslash c__kernel_sys_dvipdfmx_version_int < { 20201111 }
658 {
659 \backslash cs_gset_protected:Npn \backslash _color_backend_select_cmyk:n #1
660 { \backslash _kernel_backend_literal:n { pdf: bc ~ [#1] } }
661 \backslash cs_gset_eq:NN \backslash _color_backend_select_gray:n \backslash _color_backend_select_cmyk:n
662 \backslash cs_gset_eq:NN \backslash _color_backend_select_rgb:n \backslash _color_backend_select_cmyk:n
663 \backslash cs_gset_protected:Npn \backslash _color_backend_reset:
664 { \backslash _kernel_backend_literal:n { pdf: ec } }
665 }

(End definition for \backslash _color_backend_select_cmyk:n and others.)

666 \langle /dvipdfmx | xetex \rangle

3.4 Separations

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

```
667 < *dvipdfmx | luatex | pdftex | xetex | dvips >
```

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

```
\g_color_backend_colorant_prop
```

```
668 \prop_new:N \g__color_backend_colorant_prop
```

(End definition for \g__color_backend_colorant_prop.)

```
\_color_backend_devicen_colorants:n
```

```
\_color_backend_devicen_colorants:w
```

```
669 \cs_new:Npx \_color_backend_devicen_colorants:n #1
670 {
671   \exp_not:N \tl_if_blank:nF {#1}
672   {
673     \c_space_tl
674     << ~
675     /Colorants ~
676     << ~
677     \exp_not:N \_color_backend_devicen_colorants:w #1 ~
678     \exp_not:N \q_recursion_tail \c_space_tl
679     \exp_not:N \q_recursion_stop
680     >> ~
681   }
682 }
683 }
684 \cs_new:Npn \_color_backend_devicen_colorants:w #1 ~
685 {
686   \quark_if_recursion_tail_stop:n {#1}
687   \prop_if_in:NnT \g__color_backend_colorant_prop {#1}
688   {
689     #1 ~
690     \prop_item:Nn \g__color_backend_colorant_prop {#1} ~
691   }
692   \_color_backend_devicen_colorants:w
693 }
```

(End definition for _color_backend_devicen_colorants:n and _color_backend_devicen_colorants:w.)

```
694 < /dvipdfmx | luatex | pdftex | xetex | dvips >
```

```
695 < *dvips >
```

```
\_color_backend_select_separation:nn
```

```
\_color_backend_select_devicen:nn
```

```
696 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
697 { \_color_backend_select:n { separation ~ #1 ~ #2 } }
698 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
```

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

```
\_color_backend_select_iccbased:nn
```

No support.

```
699 \cs_new_protected:Npn \_color_backend_select_iccbased:nn #1#2 { }
```

(End definition for `_color_backend_select_iccbased:nn`.)

```
\_color_backend_separation_init:nnnn
\_color_backend_separation_init:nxxnn
\_color_backend_separation_init_aux:nnnnnn
\_color_backend_separation_init_DeviceCMYK:nnn
\_color_backend_separation_init_DeviceGray:nnn
\_color_backend_separation_init_DeviceRGB:nnn
\_color_backend_separation_init_Device:Nn
\_color_backend_separation_init:nnn
\_color_backend_separation_init_count:n
\_color_backend_separation_init_count:w
\_color_backend_separation_init:nnnn
\_color_backend_separation_init:w
\_color_backend_separation_init:n
\_color_backend_separation_init:nw
\_color_backend_separation_init_CIELAB:nnn
```

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

```
700 \cs_new_protected:Npx \_color_backend_separation_init:nnnnn #1#2#3#4#5
701 {
702   \bool_if:NT \g__kernel_backend_header_bool
703   {
704     \exp_args:Nx \_kernel_backend_first_shipout:n
705     {
706       \exp_not:N \_color_backend_separation_init_aux:nnnnnn
707       { \exp_not:N \int_use:N \g__color_model_int }
708       {#1} {#2} {#3} {#4} {#5}
709     }
710     \prop_gput:Nxx \exp_not:N \g__color_backend_colorant_prop
711     { / \exp_not:N \str_convert_pdfname:n {#1} }
712     {
713       << ~
714       /setcolorspace ~ {} ~
715       >> ~ begin ~
716       color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
717       end
718     }
719   }
720 }
721 \cs_generate_variant:Nn \_color_backend_separation_init:nnnnn { nxx }
722 \cs_new_protected:Npn \_color_backend_separation_init_aux:nnnnnn #1#2#3#4#5#6
723 {
724   \_kernel_backend_literal:e
725   {
726     !
727     TeXDict ~ begin ~
728     /color #1
729     {
730       [ ~
731       /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
732       [ ~ #3 ~ ] ~
733       {
734         \cs_if_exist_use:cF { \_color_backend_separation_init_ #3 :nnn }
735         { \_color_backend_separation_init:nnn }
736         {#4} {#5} {#6}
737       }
738       ] ~ setcolorspace
739     } ~ def ~
740     end
741   }
742 }
743 \cs_new:cpn { \_color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
744 { \_color_backend_separation_init_Device:Nn 4 {#3} }
745 \cs_new:cpn { \_color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
746 { \_color_backend_separation_init_Device:Nn 1 {#3} }
747 \cs_new:cpn { \_color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
```

```

748 { \_color_backend_separation_init_Device:Nn 2 {#3} }
749 \cs_new:Npn \_color_backend_separation_init_Device:Nn #1#2
750 {
751   #2 ~
752   \prg_replicate:nn {#1}
753   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
754   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
755 }

```

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.

```

756 \cs_new:Npn \_color_backend_separation_init:nnn #1#2#3
757 {
758   \exp_args:Ne \_color_backend_separation_init:nnnn
759   { \_color_backend_separation_init_count:n {#2} }
760   {#1} {#2} {#3}
761 }
762 \cs_new:Npn \_color_backend_separation_init_count:n #1
763 { \int_eval:n { 0 \_color_backend_separation_init_count:w #1 ~ \s_color_stop } }
764 \cs_new:Npn \_color_backend_separation_init_count:w #1 ~ #2 \s_color_stop
765 {
766   +1
767   \tl_if_blank:nF {#2}
768   { \_color_backend_separation_init_count:w #2 \s_color_stop }
769 }

```

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.

```

770 \cs_new:Npn \_color_backend_separation_init:nnnn #1#2#3#4
771 {
772   \_color_backend_separation_init:w #3 ~ \s_color_stop #4 ~ \s_color_stop
773   \prg_replicate:nn {#1}
774   {
775     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
776     \int_eval:n { 3 * #1 } ~ index ~ mul ~
777     2 ~ index ~ add ~
778     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
779   }
780   \int_step_function:nnnN {#1} { -1 } { 1 }
781   \_color_backend_separation_init:n
782   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
783   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
784   \tl_if_blank:nF {#2}

```

```

785     { \_color_backend_separation_init:nw {#1} #2 ~ \s_color_stop }
786   }
787   \cs_new:Npn \_color_backend_separation_init:w
788     #1 ~ #2 \s_color_stop #3 ~ #4 \s_color_stop
789   {
790     #1 ~ #3 ~ 0 ~
791     \tl_if_blank:nF {#2}
792     { \_color_backend_separation_init:w #2 \s_color_stop #4 \s_color_stop }
793   }
794   \cs_new:Npn \_color_backend_separation_init:n #1
795   { \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.

```

796   \cs_new:Npn \_color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s_color_stop
797   {
798     #2 ~ #3 ~
799     2 ~ index ~ 2 ~ index ~ lt ~
800     { ~ pop ~ exch ~ pop ~ } ~
801     { ~
802       2 ~ index ~ 1 ~ index ~ gt ~
803       { ~ exch ~ pop ~ exch ~ pop ~ } ~
804       { ~ pop ~ pop ~ } ~
805       ifelse ~
806     }
807     ifelse ~
808     #1 ~ 1 ~ roll ~
809     \tl_if_blank:nF {#4}
810     { \_color_backend_separation_init:nw {#1} #4 \s_color_stop }
811   }

```

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.

```

812   \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3
813   {
814     \_color_backend_separation_init:nxxxnn
815     {#2}
816     {
817       /CIEBasedABC ~
818       << ~
819       /RangeABC ~ [ ~ \c_color_model_range_CIELAB_tl \c_space_tl ] ~
820       /DecodeABC ~
821       [ ~
822         { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
823         { ~ 500 ~ div ~ } ~ bind ~
824         { ~ 200 ~ div ~ } ~ bind ~
825       ] ~
826       /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
827       /DecodeLMN ~
828       [ ~
829         { ~
830           dup ~ 6 ~ 29 ~ div ~ ge ~
831           { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
832           { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~

```

```

833         ifelse ~
834         0.9505 ~ mul ~
835     } ~ bind ~
836     { ~
837         dup ~ 6 ~ 29 ~ div ~ ge ~
838         { ~ dup ~ dup ~ mul ~ mul ~ } ~
839         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
840         ifelse ~
841     } ~ bind ~
842     { ~
843         dup ~ 6 ~ 29 ~ div ~ ge ~
844         { ~ dup ~ dup ~ mul ~ mul ~ } ~
845         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
846         ifelse ~
847         1.0890 ~ mul ~
848     } ~ bind
849 ] ~
850 /WhitePoint ~
851 [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
852 >>
853 }
854 { \c__color_model_range_CIELAB_tl }
855 { 100 ~ 0 ~ 0 }
856 {#3}
857 }

```

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

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

```

858 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3
859 {
860     \__kernel_backend_literal:e
861     {
862         !
863         TeXDict ~ begin ~
864         /color \int_use:N \g__color_model_int
865         {
866             [ ~
867                 /DeviceN ~
868                 [ ~ #1 ~ ] ~
869                 #2 ~
870                 { ~ #3 ~ } ~
871                 \_color_backend_devicen_colorants:n {#1}
872             ] ~ setcolorspace
873         } ~ def ~
874     end
875 }
876 }

```

(End definition for `_color_backend_devicen_init:nnn`.)

`_color_backend_iccbased_init:nmn` No support at present.

```

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

```


(End definition for `_color_backend_iccbased_init:nnn`.)

878 `</dvips>`

879 `<*dvisvgm>`

`_color_backend_select_separation:nn` No support at present.

`_color_backend_select_devicen:nn`

880 `\cs_new_protected:Npn _color_backend_select_separation:nn #1#2 { }`

881 `\cs_new_eq:NN _color_backend_select_devicen:nn _color_backend_select_separation:nn`

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

`_color_backend_separation_init:nnnnn` No support at present.

`_color_backend_separation_init_CIELAB:nnn`

882 `\cs_new_protected:Npn _color_backend_separation_init:nnnnn #1#2#3#4#5 { }`

883 `\cs_new_protected:Npn _color_backend_separation_init_CIELAB:nnnnnn #1#2#3 { }`

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

`_color_backend_select_iccbased:nn`

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

884 `\cs_new_protected:Npn _color_backend_select_iccbased:nn #1#2`

885 `{`

886 `_kernel_backend_literal_svg:x`

887 `{`

888 `<style>`

889 `@color-profile ~`

890 `\str_if_eq:nnTF {#2} { cmyk }`

891 `{ device-cmyk }`

892 `{ --color \int_use:N \g__color_model_int }`

893 `\c_space_tl`

894 `{`

895 `src:("#1")`

896 `}`

897 `</style>`

898 `}`

899 `}`

(End definition for `_color_backend_select_iccbased:nn`.)

900 `</dvisvgm>`

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

`_color_backend_select_separation:nn`

`_color_backend_select_devicen:nn`

`_color_backend_select_iccbased: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.

902 `\cs_new_protected:Npn _color_backend_select_separation:nn #1#2`

903 `{ _color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }`

904 `\cs_new_eq:NN _color_backend_select_devicen:nn _color_backend_select_separation:nn`

905 `\cs_new_eq:NN _color_backend_select_iccbased:nn _color_backend_select_separation:nn`

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

```

__color_backend_separation_init:nnnn
__color_backend_separation_init:nn
__color_backend_separation_init_CIELAB:nnn

```

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

```

906 \cs_new_protected:Npn __color_backend_separation_init:nnnn #1#2#3#4#5
907 {
908   \pdf_object_unnamed_write:nx { dict }
909   {
910     /FunctionType ~ 2
911     /Domain ~ [0 ~ 1]
912     \tl_if_blank:nF {#3} { /Range ~ [#3] }
913     /C0 ~ [#4] ~
914     /C1 ~ [#5] /N ~ 1
915   }
916   \exp_args:Nx __color_backend_separation_init:nn
917   { \str_convert_pdfname:n {#1} } {#2}
918   \bool_lazy_and:nnT
919   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
920   { \pdfmanagement_if_active_p: }
921   {
922     \use:x
923     {
924       \pdfmanagement_add:nnn
925       { Page / Resources / ColorSpace }
926       { color \int_use:N \g__color_model_int }
927       { \pdf_object_ref_last: }
928     }
929   }
930 }
931 \cs_new_protected:Npn __color_backend_separation_init:nn #1#2
932 {
933   \pdf_object_unnamed_write:nx { array }
934   { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
935   \prop_gput:Nnx \g__color_backend_colorant_prop { /#1 }
936   { \pdf_object_ref_last: }
937 }

```

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

```

938 \cs_new_protected:Npn __color_backend_separation_init_CIELAB:nnn #1#2#3
939 {
940   \pdf_object_if_exist:nF { __color_illuminant_CIELAB_ #1 }
941   {
942     \pdf_object_new:nn { __color_illuminant_CIELAB_ #1 } { array }
943     \pdf_object_write:nx { __color_illuminant_CIELAB_ #1 }
944     {
945       /Lab ~
946       <<
947       /WhitePoint ~
948       [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ]
949       /Range ~ [ \c__color_model_range_CIELAB_tl ]
950       >>
951     }
952   }
953   __color_backend_separation_init:nnnn

```

```

954     {#2}
955     { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
956     { \c__color_model_range_CIELAB_t1 }
957     { 100 ~ 0 ~ 0 }
958     {#3}
959 }

```

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

__color_backend_devicen_init:nnn
 __color_backend_devicen_init:w

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

```

960 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
961 {
962   \pdf_object_unnamed_write:nx { stream }
963   {
964     {
965       /FunctionType ~ 4 ~
966       /Domain ~
967       [ ~
968         \prg_replicate:nn
969         { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
970         { 0 ~ 1 ~ }
971       ] ~
972       /Range ~
973       [ ~
974         \str_case:nn {#2}
975         {
976           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
977           { /DeviceGray } { 0 ~ 1 }
978           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
979         } ~
980       ]
981     }
982     { {#3} }
983   }
984   \pdf_object_unnamed_write:nx { array }
985   {
986     /DeviceN ~
987     [ ~ #1 ~ ] ~
988     #2 ~
989     \pdf_object_ref_last:
990     \__color_backend_devicen_colorants:n {#1}
991   }
992   \bool_lazy_and:nnT
993   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
994   { \pdfmanagement_if_active_p: }
995   {
996     \use:x
997     {
998       \pdfmanagement_add:nnn
999       { Page / Resources / ColorSpace }
1000       { color \int_use:N \g__color_model_int }
1001       { \pdf_object_ref_last: }

```

```

1002     }
1003   }
1004 }
1005 \cs_new:Npn \__color_backend_devicen_init:w #1 ~ #2 \s__color_stop
1006 {
1007   + 1
1008   \tl_if_blank:nF {#2}
1009   { \__color_backend_devicen_init:w #2 \s__color_stop }
1010 }

```

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

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

```

1011 \cs_new_protected:Npn \__color_backend_iccbased_init:nnn #1#2#3
1012 {
1013   \pdf_object_if_exist:nF { __color_icc_ #1 }
1014   {
1015     \pdf_object_new:nn { __color_icc_ #1 } { fstream }
1016     \pdf_object_write:nx { __color_icc_ #1 }
1017     {
1018       {
1019         /N ~ \exp_not:n { #2 } ~
1020         \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
1021       }
1022       {#1}
1023     }
1024   }
1025   \pdf_object_unnamed_write:nx { array }
1026   { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
1027   \cs_if_exist:NT \pdfmanagement_add:nnn
1028   {
1029     \use:x
1030     {
1031       \pdfmanagement_add:nnn { Page / Resources / ColorSpace }
1032       { color \int_use:N \g__color_model_int }
1033       { ~ \pdf_object_ref_last: }
1034     }
1035   }
1036 }

```

(End definition for __color_backend_iccbased_init:nnn.)

__color_backend_iccbased_device:nnn This is very similar to setting up a color space: the only part we skip is adding it to the page resources.

```

1037 \cs_new_protected:Npn \__color_backend_iccbased_device:nnn #1#2#3
1038 {
1039   \pdf_object_if_exist:nF { __color_icc_ #1 }
1040   {
1041     \pdf_object_new:nn { __color_icc_ #1 } { fstream }
1042     \pdf_object_write:nn { __color_icc_ #1 }
1043     {
1044       { /N ~ #3 }
1045       {#1}
1046     }
1047   }

```

```

1047     }
1048     \pdf_object_unnamed_write:nx { array }
1049     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
1050     \cs_if_exist:NT \pdfmanagement_add:nnn
1051     {
1052         \use:x
1053         {
1054             \pdfmanagement_add:nnn
1055             { Page / Resources / ColorSpace }
1056             { Default #2 }
1057             { \pdf_object_ref_last: }
1058         }
1059     }
1060 }

```

(End definition for __color_backend_iccbased_device:nnn.)

```

1061 </dvipdfmx | luatex | pdftex | xetex>
1062 <*dvipdfmx | xetex>

```

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

```

1063 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
1064 {
1065     \cs_gset_protected:Npn \__color_backend_select_separation:nn #1#2 { }
1066     \cs_gset_eq:NN \__color_backend_select_devicen:nn
1067     \__color_backend_select_separation:nn
1068 }

```

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

```

1069 </dvipdfmx | xetex>

```

3.5 Fill and stroke color

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

```

1070 <*dvipdfmx | luatex | pdftex | xetex>

```

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

```

1071 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1072 { \__color_backend_fill:n { #1 ~ k } }
1073 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1074 { \__color_backend_fill:n { #1 ~ g } }
1075 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1076 { \__color_backend_fill:n { #1 ~ rg } }
1077 \cs_new_protected:Npn \__color_backend_fill:n #1
1078 {
1079     \tl_set:Nn \l__color_backend_fill_tl {#1}

```

```

1080 \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1081 { #1 ~ \l__color_backend_stroke_tl }
1082 \group_insert_after:N \__color_backend_reset:
1083 }
1084 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1085 { \__color_backend_stroke:n { #1 ~ K } }
1086 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1087 { \__color_backend_stroke:n { #1 ~ G } }
1088 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1089 { \__color_backend_stroke:n { #1 ~ RG } }
1090 \cs_new_protected:Npn \__color_backend_stroke:n #1
1091 {
1092   \tl_set:Nn \l__color_backend_stroke_tl {#1}
1093   \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
1094   { \l__color_backend_fill_tl \c_space_tl #1 }
1095   \group_insert_after:N \__color_backend_reset:
1096 }

```

(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
1097 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
1098 { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
1099 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1100 { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
1101 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1102 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

1103 </dvipdfmx | luatex | pdftex | xetex>
1104 <*dvipdfmx | xetex>

```

Deal with older (x)dvipdfmx.

```

\__color_backend_fill_cmyk:n
\__color_backend_fill_gray:n
\__color_backend_fill_rgb:n
\__color_backend_reset:
\__color_backend_stroke:n
\__color_backend_fill_separation:nn
\__color_backend_stroke_separation:nn
1105 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
1106 {
1107   \cs_gset_protected:Npn \__color_backend_fill_cmyk:n #1
1108   {
1109     \__kernel_backend_literal:n { pdf: bc ~ [#1] }
1110     \group_insert_after:N \__color_backend_reset:
1111   }
1112   \cs_gset_eq:NN \__color_backend_fill_gray:n \__color_backend_fill_cmyk:n
1113   \cs_gset_eq:NN \__color_backend_fill_rgb:n \__color_backend_fill_cmyk:n
1114   \cs_gset_protected:Npn \__color_backend_reset:
1115   { \__kernel_backend_literal:n { pdf: ec } }
1116   \cs_gset_protected:Npn \__color_backend_stroke:n #1
1117   { \__kernel_backend_literal:n {#1} }
1118   \cs_gset_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1119   \cs_gset_eq:NN \__color_backend_fill_devicen:nn
1120   \__color_backend_fill_separation:nn
1121   \cs_gset_eq:NN \__color_backend_stroke_separation:nn
1122   \__color_backend_fill_separation:nn
1123   \cs_gset_eq:NN \__color_backend_stroke_devicen:nn
1124   \__color_backend_stroke_separation:nn
1125 }

```

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

1126 `</dvipdfmx | xetex>`

1127 `<*dvips>`

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

```

1128 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1129 { \_color_backend_fill:n { cmyk ~ #1 } }
1130 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1131 { \_color_backend_fill:n { gray ~ #1 } }
1132 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1133 { \_color_backend_fill:n { rgb ~ #1 } }
1134 \cs_new_protected:Npn \_color_backend_fill:n #1
1135 {
1136   \_kernel_backend_literal:n { color~push~ #1 }
1137   \group_insert_after:N \_color_backend_reset:
1138 }
1139 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1140 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1141 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1142 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1143 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1144 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }

```

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

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
\_color_backend_fill_devicen:nn
\_color_backend_stroke_devicen:nn
1145 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1146 { \_color_backend_fill:n { separation ~ #1 ~ #2 } }
1147 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1148 { \_kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1149 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1150 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

1151 `</dvips>`

1152 `<*dvisvgm>`

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

```

1153 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1154 { \_color_backend_fill:n { cmyk ~ #1 } }
1155 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1156 { \_color_backend_fill:n { gray ~ #1 } }
1157 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1158 { \_color_backend_fill:n { rgb ~ #1 } }
1159 \cs_new_protected:Npn \_color_backend_fill:n #1
1160 {
1161   \_kernel_backend_literal:n { color~push~ #1 }
1162   \group_insert_after:N \_color_backend_reset:
1163 }

```

(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).

```

\__color_backend_stroke_cmyk:n
\__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
1164 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1165 { \__color_backend_cmyk:w #1 \s__color_stop }
1166 \cs_new_protected:Npn \__color_backend_stroke_cmyk:w
1167 #1 ~ #2 ~ #3 ~ #4 \s__color_stop
1168 {
1169   \use:x
1170   {
1171     \__color_backend:nnn
1172     { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1173     { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1174     { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }
1175   }
1176 }
1177 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1178 {
1179   \use:x
1180   {
1181     \__color_backend_stroke_gray_aux:n
1182     { \fp_eval:n { 100 * (#1) } }
1183   }
1184 }
1185 \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1186 { \__color_backend:nnn {#1} {#1} {#1} }
1187 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1188 { \__color_backend_rgb:w #1 \s__color_stop }
1189 \cs_new_protected:Npn \__color_backend_stroke_rgb:w
1190 #1 ~ #2 ~ #3 \s__color_stop
1191 {
1192   \use:x
1193   {
1194     \__color_backend:nnn
1195     { \fp_eval:n { 100 * (#1) } }
1196     { \fp_eval:n { 100 * (#2) } }
1197     { \fp_eval:n { 100 * (#3) } }
1198   }
1199 }
1200 \cs_new_protected:Npx \__color_backend:nnn #1#2#3
1201 {
1202   \__kernel_backend_scope:n
1203   {
1204     stroke =
1205     "
1206     rgb
1207     (
1208       #1 \c_percent_str ,
1209       #2 \c_percent_str ,
1210       #3 \c_percent_str
1211     )
1212     "
1213   }
1214 }

```


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

At present, these are no-ops.

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
1215 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
1216 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2 { }
1217 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1218 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn
(End definition for \_color_backend_fill_separation:nn and others.)
1219 </dvisvgm>
1220 </package>

```

4 l3backend-draw Implementation

```

1221 <*package>
1222 <@@=draw>

```

4.1 dvips backend

```

1223 <*dvips>

```

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

```

\_draw_backend_literal:n
\_draw_backend_literal:x
1224 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_postscript:n
1225 \cs_generate_variant:Nn \_draw_backend_literal:n { x }

```

(End definition for `_draw_backend_literal:n`.)

```

\_draw_backend_begin:
  \_draw_backend_end:

```

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

```

1226 \cs_new_protected:Npn \_draw_backend_begin:
1227 {
1228   \_kernel_backend_literal:n { ps::[begin] }
1229   \_draw_backend_literal:n { @beginspecial }
1230 }
1231 \cs_new_protected:Npn \_draw_backend_end:
1232 {
1233   \_draw_backend_literal:n { @endspecial }
1234   \_kernel_backend_literal:n { ps::[end] }
1235 }

```

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

```

\_draw_backend_scope_begin:
\_draw_backend_scope_end:

```

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

```

1236 \cs_new_protected:Npn \_draw_backend_scope_begin:
1237 { \_draw_backend_literal:n { save } }
1238 \cs_new_protected:Npn \_draw_backend_scope_end:
1239 { \_draw_backend_literal:n { restore } }

```

(End definition for `_draw_backend_scope_begin:` and `_draw_backend_scope_end:`.)

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

```

1240 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1241 {
1242   \_draw_backend_literal:x
1243   {
1244     \dim_to_decimal_in_bp:n {#1} ~
1245     \dim_to_decimal_in_bp:n {#2} ~ moveto
1246   }
1247 }
1248 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1249 {
1250   \_draw_backend_literal:x
1251   {
1252     \dim_to_decimal_in_bp:n {#1} ~
1253     \dim_to_decimal_in_bp:n {#2} ~ lineto
1254   }
1255 }
1256 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1257 {
1258   \_draw_backend_literal:x
1259   {
1260     \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1261     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1262     moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1263   }
1264 }
1265 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1266 {
1267   \_draw_backend_literal:x
1268   {
1269     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1270     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1271     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1272     curveto
1273   }
1274 }

```

(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`

```

1275 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1276 { \bool_gset_true:N \g__draw_draw_eor_bool }
1277 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1278 { \bool_gset_false:N \g__draw_draw_eor_bool }
1279 \bool_new:N \g__draw_draw_eor_bool

```

(End definition for `_draw_backend_evenodd_rule:`, `_draw_backend_nonzero_rule:`, and `\g__draw_draw_eor_bool`.)

`__draw_backend_closepath:` Unlike PDF, PostScript doesn't track separate colors for strokes and other elements. It is
`__draw_backend_stroke:` also desirable to have the `clip` keyword after a stroke or fill. To achieve those outcomes,
`__draw_backend_closestroke:` there is some work to do. For color, the stroke color is simple but the fill one has to be
`__draw_backend_fill:` inserted by hand. For clipping, the required ordering is achieved using a TeX switch.
`__draw_backend_fillstroke:` All of the operations end with a new path instruction as they do not terminate (again in
`__draw_backend_clip:` contrast to PDF).
`__draw_backend_discardpath:`
`\g__draw_draw_clip_bool`

```

1280 \cs_new_protected:Npn \__draw_backend_closepath:
1281 { \__draw_backend_literal:n { closepath } }
1282 \cs_new_protected:Npn \__draw_backend_stroke:
1283 {
1284   \__draw_backend_literal:n { gsave }
1285   \__draw_backend_literal:n { color.sc }
1286   \__draw_backend_literal:n { stroke }
1287   \__draw_backend_literal:n { grestore }
1288   \bool_if:NT \g__draw_draw_clip_bool
1289   {
1290     \__draw_backend_literal:x
1291     {
1292       \bool_if:NT \g__draw_draw_eor_bool { eo }
1293       clip
1294     }
1295   }
1296   \__draw_backend_literal:n { newpath }
1297   \bool_gset_false:N \g__draw_draw_clip_bool
1298 }
1299 \cs_new_protected:Npn \__draw_backend_closestroke:
1300 {
1301   \__draw_backend_closepath:
1302   \__draw_backend_stroke:
1303 }
1304 \cs_new_protected:Npn \__draw_backend_fill:
1305 {
1306   \__draw_backend_literal:x
1307   {
1308     \bool_if:NT \g__draw_draw_eor_bool { eo }
1309     fill
1310   }
1311   \bool_if:NT \g__draw_draw_clip_bool
1312   {
1313     \__draw_backend_literal:x
1314     {
1315       \bool_if:NT \g__draw_draw_eor_bool { eo }
1316       clip
1317     }
1318   }
1319   \__draw_backend_literal:n { newpath }
1320   \bool_gset_false:N \g__draw_draw_clip_bool
1321 }
1322 \cs_new_protected:Npn \__draw_backend_fillstroke:
1323 {
1324   \__draw_backend_literal:x
1325   {
1326     \bool_if:NT \g__draw_draw_eor_bool { eo }

```

```

1327         fill
1328     }
1329     \__draw_backend_literal:n { gsave }
1330     \__draw_backend_literal:n { color.sc }
1331     \__draw_backend_literal:n { stroke }
1332     \__draw_backend_literal:n { grestore }
1333     \bool_if:NT \g__draw_draw_clip_bool
1334     {
1335         \__draw_backend_literal:x
1336         {
1337             \bool_if:NT \g__draw_draw_eor_bool { eo }
1338             clip
1339         }
1340     }
1341     \__draw_backend_literal:n { newpath }
1342     \bool_gset_false:N \g__draw_draw_clip_bool
1343 }
1344 \cs_new_protected:Npn \__draw_backend_clip:
1345 { \bool_gset_true:N \g__draw_draw_clip_bool }
1346 \bool_new:N \g__draw_draw_clip_bool
1347 \cs_new_protected:Npn \__draw_backend_discardpath:
1348 {
1349     \bool_if:NT \g__draw_draw_clip_bool
1350     {
1351         \__draw_backend_literal:x
1352         {
1353             \bool_if:NT \g__draw_draw_eor_bool { eo }
1354             clip
1355         }
1356     }
1357     \__draw_backend_literal:n { newpath }
1358     \bool_gset_false:N \g__draw_draw_clip_bool
1359 }

```

(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:
1360 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1361 {
1362     \__draw_backend_literal:x
1363     {
1364         [
1365             \exp_args:Nf \use:n
1366             { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1367         ] ~
1368         \dim_to_decimal_in_bp:n {#2} ~ setdash
1369     }
1370 }
1371 \cs_new:Npn \__draw_backend_dash:n #1
1372 { ~ \dim_to_decimal_in_bp:n {#1} }
1373 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1374 {
1375     \__draw_backend_literal:x
1376     { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }

```

```

1377 }
1378 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1379 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1380 \cs_new_protected:Npn \__draw_backend_cap_butt:
1381 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1382 \cs_new_protected:Npn \__draw_backend_cap_round:
1383 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1384 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1385 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1386 \cs_new_protected:Npn \__draw_backend_join_miter:
1387 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1388 \cs_new_protected:Npn \__draw_backend_join_round:
1389 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1390 \cs_new_protected:Npn \__draw_backend_join_bevel:
1391 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

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

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

```

1392 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1393 {
1394   \__draw_backend_literal:n
1395   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1396 }

```

(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).

```

1397 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1398 {
1399   \__draw_backend_literal:n { @endspecial }
1400   \__draw_backend_literal:n { [end] }
1401   \__draw_backend_literal:n { [begin] }
1402   \__draw_backend_literal:n { save }
1403   \__draw_backend_literal:n { currentpoint }
1404   \__draw_backend_literal:n { currentpoint~translate }
1405   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1406   \__draw_backend_cm:nnnn { #2 } { #3 } { #4 } { #5 }
1407   \__draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1408   \__draw_backend_literal:n { neg~exch~neg~exch~translate }

```

```

1409     \__draw_backend_literal:n { [end] }
1410     \hbox_overlap_right:n { \box_use:N #1 }
1411     \__draw_backend_literal:n { [begin] }
1412     \__draw_backend_literal:n { restore }
1413     \__draw_backend_literal:n { [end] }
1414     \__draw_backend_literal:n { [begin] }
1415     \__draw_backend_literal:n { @beginspecial }
1416 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```

1417 </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.

```

1418 <*dvipdfmx | luatex | pdftex | xetex>

```

4.2.1 Drawing

__draw_backend_literal:n Pass data through using a dedicated interface.

```

\__draw_backend_literal:x
1419 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_pdf:n
1420 \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:
1421 \cs_new_protected:Npn \__draw_backend_begin:
1422 { \__draw_backend_scope_begin: }
1423 \cs_new_protected:Npn \__draw_backend_end:
1424 { \__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:
1425 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1426 \cs_new_eq:NN \__draw_backend_scope_end: \__kernel_backend_scope_end:

```

(End definition for __draw_backend_scope_begin: and __draw_backend_scope_end:.)

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

```

\__draw_backend_lineto:nn
\__draw_backend_moveto:nnnnnn
\__draw_backend_rectangle:nnnn
1427 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1428 {
1429     \__draw_backend_literal:x
1430     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1431 }
1432 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1433 {
1434     \__draw_backend_literal:x
1435     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1436 }
1437 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1438 {

```

```

1439 \__draw_backend_literal:x
1440 {
1441   \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1442   \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1443   \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1444   c
1445 }
1446 }
1447 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1448 {
1449   \__draw_backend_literal:x
1450   {
1451     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1452     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1453     re
1454   }
1455 }

```

(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.

```

1456 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1457 { \bool_gset_true:N \g__draw_draw_eor_bool }
1458 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1459 { \bool_gset_false:N \g__draw_draw_eor_bool }
1460 \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.

```

1461 \cs_new_protected:Npn \__draw_backend_closepath:
1462 { \__draw_backend_literal:n { h } }
1463 \cs_new_protected:Npn \__draw_backend_stroke:
1464 { \__draw_backend_literal:n { S } }
1465 \cs_new_protected:Npn \__draw_backend_closestroke:
1466 { \__draw_backend_literal:n { s } }
1467 \cs_new_protected:Npn \__draw_backend_fill:
1468 {
1469   \__draw_backend_literal:x
1470   { f \bool_if:NT \g__draw_draw_eor_bool * }
1471 }
1472 \cs_new_protected:Npn \__draw_backend_fillstroke:
1473 {
1474   \__draw_backend_literal:x
1475   { B \bool_if:NT \g__draw_draw_eor_bool * }
1476 }
1477 \cs_new_protected:Npn \__draw_backend_clip:
1478 {
1479   \__draw_backend_literal:x
1480   { W \bool_if:NT \g__draw_draw_eor_bool * }
1481 }
1482 \cs_new_protected:Npn \__draw_backend_discardpath:
1483 { \__draw_backend_literal:n { n } }

```

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

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_but:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1484 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1485 {
1486   \__draw_backend_literal:x
1487   {
1488     [
1489       \exp_args:Nf \use:n
1490       { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1491     ] ~
1492     \dim_to_decimal_in_bp:n {#2} ~ d
1493   }
1494 }
1495 \cs_new:Npn \__draw_backend_dash:n #1
1496 { ~ \dim_to_decimal_in_bp:n {#1} }
1497 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1498 {
1499   \__draw_backend_literal:x
1500   { \dim_to_decimal_in_bp:n {#1} ~ w }
1501 }
1502 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1503 { \__draw_backend_literal:x { #1 ~ M } }
1504 \cs_new_protected:Npn \__draw_backend_cap_but:
1505 { \__draw_backend_literal:n { 0 ~ J } }
1506 \cs_new_protected:Npn \__draw_backend_cap_round:
1507 { \__draw_backend_literal:n { 1 ~ J } }
1508 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1509 { \__draw_backend_literal:n { 2 ~ J } }
1510 \cs_new_protected:Npn \__draw_backend_join_miter:
1511 { \__draw_backend_literal:n { 0 ~ j } }
1512 \cs_new_protected:Npn \__draw_backend_join_round:
1513 { \__draw_backend_literal:n { 1 ~ j } }
1514 \cs_new_protected:Npn \__draw_backend_join_bevel:
1515 { \__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!

```

1516 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1517 {
1518   < *luatex | pdftex >
1519   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1520   < /luatex | pdftex >
1521   < *dvipdfmx | xetex >
1522   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1523   \__draw_backend_cm_aux:nnnn
1524   < /dvipdfmx | xetex >

```



```

1525 }
1526 <*/dvipdfmx | xetex>
1527 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1528 {
1529   \__kernel_backend_literal:x
1530   {
1531     x:rotate~
1532     \fp_compare:nNnTF {#1} = \c_zero_fp
1533     { 0 }
1534     { \fp_eval:n { round ( -#1 , 5 ) } }
1535   }
1536   \__kernel_backend_literal:x
1537   {
1538     x:scale~
1539     \fp_eval:n { round ( #2 , 5 ) } ~
1540     \fp_eval:n { round ( #3 , 5 ) }
1541   }
1542   \__kernel_backend_literal:x
1543   {
1544     x:rotate~
1545     \fp_compare:nNnTF {#4} = \c_zero_fp
1546     { 0 }
1547     { \fp_eval:n { round ( -#4 , 5 ) } }
1548   }
1549 }
1550 </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.

```

1551 <*dvipdfmx|xetex>
1552 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1553 {
1554   \use:x
1555   {
1556     \__draw_backend_cm_decompose_auxi:nnnnN
1557     { \fp_eval:n { (#1 + #4) / 2 } }
1558     { \fp_eval:n { (#1 - #4) / 2 } }
1559     { \fp_eval:n { (#3 + #2) / 2 } }
1560     { \fp_eval:n { (#3 - #2) / 2 } }
1561   }
1562   #5
1563 }
1564 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1565 {
1566   \use:x
1567   {
1568     \__draw_backend_cm_decompose_auxii:nnnnN
1569     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1570     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1571     { \fp_eval:n { atand ( #3 , #2 ) } }
1572     { \fp_eval:n { atand ( #4 , #1 ) } }
1573   }
1574   #5
1575 }
1576 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1577 {
1578   \use:x
1579   {
1580     \__draw_backend_cm_decompose_auxiii:nnnnN
1581     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1582     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1583     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1584     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1585   }
1586   #5
1587 }
1588 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1589 {
1590   \fp_compare:nNnTF { abs ( #2 ) } > { abs ( #3 ) }
1591   { #5 {#1} {#2} {#3} {#4} }
1592   { #5 {#1} {#3} {#2} {#4} }
1593 }
1594 </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.

```

1595 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1596 {
1597   \__kernel_backend_scope_begin:
1598   <*luatex | pdftex>
1599   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1600   </luatex | pdftex>
1601   <*dvipdfmx | xetex>
1602   \__kernel_backend_literal:n
1603   { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1604   </dvipdfmx | xetex>
1605   \hbox_overlap_right:n { \box_use:N #1 }
1606   <*dvipdfmx | xetex>
1607   \__kernel_backend_literal:n { pdf:etrans }
1608   </dvipdfmx | xetex>
1609   \__kernel_backend_scope_end:
1610 }

(End definition for \__draw_backend_box_use:Nnnnn.)

1611 </dvipdfmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1612 <*dvisvgm>

```

The same as the more general literal call.

```

\__draw_backend_literal:n
\__draw_backend_literal:x
1613 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1614 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for __draw_backend_literal:n.)

Use the backend-level scope mechanisms.

```

\__draw_backend_scope_begin:
\__draw_backend_scope_end:
1615 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1616 \cs_new_eq:NN \__draw_backend_scope_end: \__kernel_backend_scope_end:

```

(End definition for __draw_backend_scope_begin: and __draw_backend_scope_end:.)

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

```

\__draw_backend_begin:
\__draw_backend_end:
1617 \cs_new_protected:Npn \__draw_backend_begin:
1618 {
1619   \__kernel_backend_scope_begin:
1620   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1621 }
1622 \cs_new_eq:NN \__draw_backend_end: \__kernel_backend_scope_end:

```

(End definition for __draw_backend_begin: and __draw_backend_end:.)

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_moveto:nn
\__draw_backend_lineto:nn
\__draw_backend_rectangle:nnnn
\__draw_backend_curveto:nnnnnn
\__draw_backend_add_to_path:n
\g__draw_backend_path_tl
1623 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1624 {

```

```

1625 \__draw_backend_add_to_path:n
1626 { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1627 }
1628 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1629 {
1630 \__draw_backend_add_to_path:n
1631 { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1632 }
1633 \cs_new_protected:Npn \__draw_backend_rectangle:nnnn #1#2#3#4
1634 {
1635 \__draw_backend_add_to_path:n
1636 {
1637 M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1638 h ~ \dim_to_decimal:n {#3} ~
1639 v ~ \dim_to_decimal:n {#4} ~
1640 h ~ \dim_to_decimal:n { -#3 } ~
1641 Z
1642 }
1643 }
1644 \cs_new_protected:Npn \__draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1645 {
1646 \__draw_backend_add_to_path:n
1647 {
1648 C ~
1649 \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1650 \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1651 \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1652 }
1653 }
1654 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1655 {
1656 \tl_gset:Nx \g__draw_backend_path_tl
1657 {
1658 \g__draw_backend_path_tl
1659 \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1660 #1
1661 }
1662 }
1663 \tl_new:N \g__draw_backend_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: 1664 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1665 { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1666 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1667 { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

(End definition for __draw_backend_evenodd_rule: and __draw_backend_nonzero_rule:.)

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

```

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

```

```

1668 \cs_new_protected:Npn \__draw_backend_closepath:
1669 { \__draw_backend_add_to_path:n { Z } }
1670 \cs_new_protected:Npn \__draw_backend_path:n #1
1671 {
1672   \bool_if:NTF \g__draw_draw_clip_bool
1673   {
1674     \int_gincr:N \g__kernel_clip_path_int
1675     \__draw_backend_literal:x
1676     {
1677       < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1678       { ?nl }
1679       <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1680       < /clipPath > { ? nl }
1681       <
1682       use~xlink:href =
1683       "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1684       #1
1685       />
1686     }
1687     \__kernel_backend_scope:x
1688     {
1689       clip-path =
1690       "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int )"
1691     }
1692   }
1693   {
1694     \__draw_backend_literal:x
1695     { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1696   }
1697   \tl_gclear:N \g__draw_backend_path_tl
1698   \bool_gset_false:N \g__draw_draw_clip_bool
1699 }
1700 \int_new:N \g__draw_backend_path_int
1701 \cs_new_protected:Npn \__draw_backend_stroke:
1702 { \__draw_backend_path:n { style="fill:none" } }
1703 \cs_new_protected:Npn \__draw_backend_closestroke:
1704 {
1705   \__draw_backend_closepath:
1706   \__draw_backend_stroke:
1707 }
1708 \cs_new_protected:Npn \__draw_backend_fill:
1709 { \__draw_backend_path:n { style="stroke:none" } }
1710 \cs_new_protected:Npn \__draw_backend_fillstroke:
1711 { \__draw_backend_path:n { } }
1712 \cs_new_protected:Npn \__draw_backend_clip:
1713 { \bool_gset_true:N \g__draw_draw_clip_bool }
1714 \bool_new:N \g__draw_draw_clip_bool
1715 \cs_new_protected:Npn \__draw_backend_discardpath:
1716 {
1717   \bool_if:NT \g__draw_draw_clip_bool
1718   {
1719     \int_gincr:N \g__kernel_clip_path_int
1720     \__draw_backend_literal:x
1721     {

```

```

1722         < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1723         { ?nl }
1724         <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1725         < /clipPath >
1726     }
1727     \__kernel_backend_scope:x
1728     {
1729         clip-path =
1730         "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1731     }
1732 }
1733 \tl_gclear:N \g__draw_path_tl
1734 \bool_gset_false:N \g__draw_draw_clip_bool
1735 }

```

(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:
1736 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1737 {
1738     \use:x
1739     {
1740         \__draw_backend_dash_aux:nn
1741         { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1742         { \dim_to_decimal:n {#2} }
1743     }
1744 }
1745 \cs_new:Npn \__draw_backend_dash:n #1
1746 { , \dim_to_decimal_in_bp:n {#1} }
1747 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1748 {
1749     \__kernel_backend_scope:x
1750     {
1751         stroke-dasharray =
1752         "
1753         \tl_if_empty:nTF {#1}
1754         { none }
1755         { \use_none:n #1 }
1756         " ~
1757         stroke-offset=" #2 "
1758     }
1759 }
1760 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1761 { \__kernel_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1762 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1763 { \__kernel_backend_scope:x { stroke-miterlimit=" #1 " } }
1764 \cs_new_protected:Npn \__draw_backend_cap_but:
1765 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1766 \cs_new_protected:Npn \__draw_backend_cap_round:
1767 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1768 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1769 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1770 \cs_new_protected:Npn \__draw_backend_join_miter:

```

```

1771 { \_kernel_backend_scope:n { stroke-linejoin="miter" } }
1772 \cs_new_protected:Npn \_draw_backend_join_round:
1773 { \_kernel_backend_scope:n { stroke-linejoin="round" } }
1774 \cs_new_protected:Npn \_draw_backend_join_bevel:
1775 { \_kernel_backend_scope:n { stroke-linejoin="bevel" } }

```

(End definition for _draw_backend_dash_pattern:nn and others.)

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

```

1776 \cs_new_protected:Npn \_draw_backend_cm:nnnn #1#2#3#4
1777 {
1778   \_kernel_backend_scope:n
1779   {
1780     transform =
1781     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1782   }
1783 }

```

(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.

```

1784 \cs_new_protected:Npn \_draw_backend_box_use:Nnnnn #1#2#3#4#5
1785 {
1786   \_kernel_backend_scope_begin:
1787   \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1788   \_kernel_backend_literal_svg:n
1789   {
1790     < g~
1791     stroke="none"~
1792     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1793     >
1794   }
1795   \box_set_wd:Nn #1 { Opt }
1796   \box_set_ht:Nn #1 { Opt }
1797   \box_set_dp:Nn #1 { Opt }
1798   \box_use:N #1
1799   \_kernel_backend_literal_svg:n { </g> }
1800   \_kernel_backend_scope_end:
1801 }

```

(End definition for _draw_backend_box_use:Nnnnn.)

```

1802 </dvisvgm>
1803 </package>

```

5 l3backend-graphics Implementation

```

1804 <*package>
1805 <@@=graphics>

```

5.1 dvips backend

```

1806 <*dvips>

```

`_graphics_backend_getbb_eps:n` Simply use the generic function.

```

1807 \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.
1808 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
1809 {
1810   \_kernel_backend_literal:x
1811   {
1812     PSfile = #1 \c_space_tl
1813     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
1814     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1815     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1816     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1817   }
1818 }
(End definition for \_graphics_backend_include_eps:n.)
1819 </dvips>

```

5.2 LuaTeX and pdfTeX backends

1820 <*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.

```

1821 \tl_new:N \l__graphics_graphics_attr_tl
(End definition for \l__graphics_graphics_attr_tl.)

```

`_graphics_backend_getbb_jpg: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.

```

1822 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1823 {
1824   \int_zero:N \l_graphics_page_int
1825   \tl_clear:N \l_graphics_pagebox_tl
1826   \tl_set:Nx \l__graphics_graphics_attr_tl
1827   {
1828     \tl_if_empty:NF \l_graphics_decodearray_tl
1829     { :D \l_graphics_decodearray_tl }
1830     \bool_if:NT \l_graphics_interpolate_bool
1831     { :I }
1832   }
1833   \tl_clear:N \l__graphics_graphics_attr_tl
1834   \_graphics_backend_getbb_auxi:n {#1}
1835 }
1836 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n

```



```

1837 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1838 {
1839   \tl_clear:N \l_graphics_decodearray_tl
1840   \bool_set_false:N \l_graphics_interpolate_bool
1841   \tl_set:Nx \l__graphics_graphics_attr_tl
1842   {
1843     : \l_graphics_pagebox_tl
1844     \int_compare:nNnT \l_graphics_page_int > 1
1845     { :P \int_use:N \l_graphics_page_int }
1846   }
1847   \__graphics_backend_getbb_auxi:n {#1}
1848 }
1849 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1850 {
1851   \graphics_bb_restore:xF { #1 \l__graphics_graphics_attr_tl }
1852   { \__graphics_backend_getbb_auxii:n {#1} }
1853 }

```

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.

```

1854 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1855 {
1856   \tex_immediate:D \tex_pdfximage:D
1857   \bool_lazy_or:nnT
1858   { \l_graphics_interpolate_bool }
1859   { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
1860   {
1861     attr ~
1862     {
1863       \tl_if_empty:NF \l_graphics_decodearray_tl
1864       { /Decode~[ \l_graphics_decodearray_tl ] }
1865       \bool_if:NT \l_graphics_interpolate_bool
1866       { /Interpolate~true }
1867     }
1868   }
1869   \int_compare:nNnT \l_graphics_page_int > 0
1870   { page ~ \int_use:N \l_graphics_page_int }
1871   \tl_if_empty:NF \l_graphics_pagebox_tl
1872   { \l_graphics_pagebox_tl }
1873   {#1}
1874   \hbox_set:Nn \l__graphics_internal_box
1875   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1876   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1877   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1878   \int_const:cn { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1879   { \tex_the:D \tex_pdflastximage:D }
1880   \graphics_bb_save:x { #1 \l__graphics_graphics_attr_tl }
1881 }

```

(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.

```

1882 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1883 {
1884   \tex_pdfrefximage:D
1885   \int_use:c { c__graphics_graphics_ #1 \l__graphics_graphics_attr_tl _int }
1886 }
1887 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1888 \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
1889 \sys_if_shell:T
1890 {
1891   \str_new:N \l__graphics_backend_dir_str
1892   \str_new:N \l__graphics_backend_name_str
1893   \str_new:N \l__graphics_backend_ext_str
1894   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1895   {
1896     \file_parse_full_name:nNNN {#1}
1897     \l__graphics_backend_dir_str
1898     \l__graphics_backend_name_str
1899     \l__graphics_backend_ext_str
1900     \exp_args:Nx \__graphics_backend_getbb_eps:nn
1901     {
1902       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1903       -converted-to.pdf
1904     }
1905     {#1}
1906   }
1907   \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1908   {
1909     \file_compare_timestamp:nNnT {#2} > {#1}
1910     {
1911       \sys_shell_now:n
1912       { repstopdf ~ #2 ~ #1 }
1913     }
1914     \tl_set:Nn \l__graphics_name_tl {#1}
1915     \__graphics_backend_getbb_pdf:n {#1}
1916   }
1917   \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1918   {
1919     \file_parse_full_name:nNNN {#1}
1920     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1921     \exp_args:Nx \__graphics_backend_include_pdf:n
1922     {
1923       \l__graphics_backend_name_str - \str_tail:N \l__graphics_backend_ext_str
1924       -converted-to.pdf
1925     }
1926   }
1927 }

```

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

1928 `</luatex | pdftex>`

5.3 dvipdfmx backend

1929 `<*dvipdfmx | xetex>`

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

```

\graphics_backend_getbb_eps:n 1930 \cs_new_eq:NN \_graphics_backend_getbb_eps:n \graphics_read_bb:n
\graphics_backend_getbb_jpg:n 1931 <*dvipdfmx>
\graphics_backend_getbb_pdf:n 1932 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
\graphics_backend_getbb_png:n 1933 {
1934   \int_zero:N \l_graphics_page_int
1935   \tl_clear:N \l_graphics_pagebox_tl
1936   \graphics_extract_bb:n {#1}
1937 }
1938 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1939 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1940 {
1941   \tl_clear:N \l_graphics_decodearray_tl
1942   \bool_set_false:N \l_graphics_interpolate_bool
1943   \graphics_extract_bb:n {#1}
1944 }
1945 </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.

1946 `\int_new:N \g__graphics_track_int`

(End definition for `\g__graphics_track_int`.)

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

```

\_graphics_backend_include_eps:n 1947 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
\_graphics_backend_include_jpg:n 1948 {
\_graphics_backend_include_pdf:n 1949   \__kernel_backend_literal:x
\_graphics_backend_include_png:n 1950   {
\_graphics_backend_include_auxi:nn 1951     PSfile = #1 \c_space_tl
\_graphics_backend_include_auxii:nnn 1952     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
\_graphics_backend_include_auxiii:nnnn 1953     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
1954     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
1955     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
1956   }
1957 }
1958 \cs_new_protected:Npn \_graphics_backend_include_jpg:n #1
1959 { \_graphics_backend_include_auxi:nn {#1} { image } }
1960 \cs_new_eq:NN \_graphics_backend_include_png:n \_graphics_backend_include_jpg:n
1961 <*dvipdfmx>
1962 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
1963 { \_graphics_backend_include_auxi:nn {#1} { epdf } }
1964 </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.

```

1965 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1966 {
1967   \__graphics_backend_include_auxii:xnn
1968   {
1969     \tl_if_empty:NF \l_graphics_pagebox_tl
1970     { : \l_graphics_pagebox_tl }
1971     \int_compare:nNnT \l_graphics_page_int > 1
1972     { :P \int_use:N \l_graphics_page_int }
1973     \tl_if_empty:NF \l_graphics_decodearray_tl
1974     { :D \l_graphics_decodearray_tl }
1975     \bool_if:NT \l_graphics_interpolate_bool
1976     { :I }
1977   }
1978   {#1} {#2}
1979 }
1980 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1981 {
1982   \int_if_exist:cTF { c__graphics_graphics_ #2#1 _int }
1983   {
1984     \__kernel_backend_literal:x
1985     { pdf:usexobj~@graphic \int_use:c { c__graphics_graphics_ #2#1 _int } }
1986   }
1987   { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1988 }
1989 \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!

```

1990 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1991 {
1992   \int_gincr:N \g__graphics_track_int
1993   \int_const:cn { c__graphics_graphics_ #1#2 _int } { \g__graphics_track_int }
1994   \__kernel_backend_literal:x
1995   {
1996     pdf:#3~
1997     @graphic \int_use:c { c__graphics_graphics_ #1#2 _int } ~
1998     \int_compare:nNnT \l_graphics_page_int > 1
1999     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
2000     \tl_if_empty:NF \l_graphics_pagebox_tl
2001     {
2002       pagebox ~ \l_graphics_pagebox_tl \c_space_tl
2003       bbox ~
2004       \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
2005       \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2006       \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2007       \dim_to_decimal_in_bp:n \l_graphics_ury_dim \c_space_tl
2008     }
2009     (#1)
2010     \bool_lazy_or:nnT

```

```

2011         { \l_graphics_interpolate_bool }
2012         { ! \tl_if_empty_p:N \l_graphics_decodearray_tl }
2013         {
2014             <<
2015             \tl_if_empty:NF \l_graphics_decodearray_tl
2016             { /Decode~[ \l_graphics_decodearray_tl ] }
2017             \bool_if:NT \l_graphics_interpolate_bool
2018             { /Interpolate~true> }
2019             >>
2020         }
2021     }
2022 }

```

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

```

2023 </dvipdfmx | xetex>

```

5.4 X_YTeX backend

```

2024 <*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
2025 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2026 {
2027     \int_zero:N \l_graphics_page_int
2028     \tl_clear:N \l_graphics_pagebox_tl
2029     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2030 }
2031 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
2032 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2033 {
2034     \tl_clear:N \l_graphics_decodearray_tl
2035     \bool_set_false:N \l_graphics_interpolate_bool
2036     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2037 }
2038 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2039 {
2040     \int_compare:nNnTF \l_graphics_page_int > 1
2041     { \__graphics_backend_getbb_auxii:VnN \l_graphics_page_int {#1} #2 }
2042     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2043 }
2044 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2045 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2046 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2047 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2048 {
2049     \tl_if_empty:NTF \l_graphics_pagebox_tl
2050     { \__graphics_backend_getbb_auxiv:VnNnn \l_graphics_pagebox_tl }
2051     { \__graphics_backend_getbb_auxv:nNnn }
2052     {#1} #2 {#3} {#4}

```

```

2053 }
2054 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nNnn #1#2#3#4#5
2055 {
2056   \use:x
2057   {
2058     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2059     { #5 ~ \__graphics_backend_getbb_pagebox:w #1 }
2060   }
2061 }
2062 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nNnn { V }
2063 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2064 {
2065   \graphics_bb_restore:nF {#1#3}
2066   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2067 }
2068 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2069 {
2070   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
2071   \dim_set:Nn \l_graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2072   \dim_set:Nn \l_graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2073   \graphics_bb_save:n {#1#3}
2074 }
2075 \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.

```

2076 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2077 {
2078   \tex_XeTeXpdf file:D
2079   \__graphics_backend_include_pdf_quote:w #1 "#1" \s__graphics_stop \c_space_tl
2080   \int_compare:nNnT \l_graphics_page_int > 0
2081     { page ~ \int_use:N \l_graphics_page_int \c_space_tl }
2082   \exp_after:wN \__graphics_backend_getbb_pagebox:w \l_graphics_pagebox_tl
2083 }
2084 \cs_new:Npn \__graphics_backend_include_pdf_quote:w #1 " #2 " #3 \s__graphics_stop
2085 { " #2 " }

```

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

```
2086 </xetex>
```

5.5 dvisvgm backend

```
2087 <*dvisvgm>
```

__graphics_backend_getbb_eps:n

Simply use the generic function.

```
2088 \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`

```

2089 \cs_new_protected:Npn __graphics_backend_getbb_jpg:n #1
2090 {
2091   \int_zero:N \l_graphics_page_int
2092   \tl_clear:N \l_graphics_pagebox_tl
2093   \graphics_extract_bb:n {#1}
2094 }
2095 \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

```

2096 \cs_new_protected:Npn __graphics_backend_getbb_pdf:n #1
2097 {
2098   \tl_clear:N \l_graphics_decodearray_tl
2099   \bool_set_false:N \l_graphics_interpolate_bool
2100   \graphics_extract_bb:n {#1}
2101 }

```

(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
`__graphics_backend_include_pdf:n` is the same as the `dvips` code.)
`__graphics_backend_include:nn`

```

2102 \cs_new_protected:Npn __graphics_backend_include_eps:n #1
2103 { __graphics_backend_include:nn { PSfile } {#1} }
2104 \cs_new_protected:Npn __graphics_backend_include_pdf:n #1
2105 { __graphics_backend_include:nn { pdffile } {#1} }
2106 \cs_new_protected:Npn __graphics_backend_include:nn #1#2
2107 {
2108   \__kernel_backend_literal:x
2109   {
2110     #1 = #2 \c_space_tl
2111     llx = \dim_to_decimal_in_bp:n \l_graphics_llx_dim \c_space_tl
2112     lly = \dim_to_decimal_in_bp:n \l_graphics_lly_dim \c_space_tl
2113     urx = \dim_to_decimal_in_bp:n \l_graphics_urx_dim \c_space_tl
2114     ury = \dim_to_decimal_in_bp:n \l_graphics_ury_dim
2115   }
2116 }

```

(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
`__graphics_backend_include_jpg:n` more complex approach, needed if clipping, *etc.*, is covered at the graphic backend level).
`__graphics_backend_include_bitmap_quote:w` 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.

```

2117 \cs_new_protected:Npn __graphics_backend_include_png:n #1
2118 {
2119   \__kernel_backend_literal:x
2120   {
2121     dvisvgm:img~
2122     \dim_to_decimal:n { \l_graphics_ury_dim } ~
2123     \dim_to_decimal:n { \l_graphics_ury_dim } ~

```

```

2124         \_graphics_backend_include_bitmap_quote:w #1 " #1 " \s__graphics_stop
2125     }
2126 }
2127 \cs_new_eq:NN \_graphics_backend_include_jpg:n \_graphics_backend_include_png:n
2128 \cs_new:Npn \_graphics_backend_include_bitmap_quote:w #1 " #2 " #3 \s__graphics_stop
2129 { " #2 " }

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

2130 </dvisvgm>
2131 </package>

```

6 l3backend-pdf Implementation

```

2132 <*package>
2133 <@=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

2134 \box_new:N \l__pdf_internal_box

(End definition for \l__pdf_internal_box.)

```

6.2 dvips backend

```

2135 <*dvips>

\_pdf_backend_pdfmark:n Used often enough it should be a separate function.
\_pdf_backend_pdfmark:x
2136 \cs_new_protected:Npn \_pdf_backend_pdfmark:n #1
2137 { \_kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2138 \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
2139 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2140 { \_pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2141 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2142 { \_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.

```
2143 \int_new:N \g__pdf_backend_object_int
2144 \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.

```
2145 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2146 {
2147   \int_gincr:N \g__pdf_backend_object_int
2148   \int_const:cn
2149     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2150     { \g__pdf_backend_object_int }
2151   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2152 }
2153 \cs_new:Npn \__pdf_backend_object_ref:n #1
2154 { { 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:nmn

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

```
2155 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2156 {
2157   \__pdf_backend_pdfmark:x
2158   {
2159     /_objdef ~ \__pdf_backend_object_ref:n {#1}
2160     /type
2161     \str_case_e:nn
2162       { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2163       {
2164         { array } { /array }
2165         { dict } { /dict }
2166         { fstream } { /stream }
2167         { stream } { /stream }
2168       }
2169     /OBJ
2170   }
2171   \use:c
2172     { __pdf_backend_object_write_ \prop_item:Nn \g__pdf_backend_object_prop {#1} :nn }
2173     { \__pdf_backend_object_ref:n {#1} } {#2}
2174 }
2175 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2176 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2177 {
2178   \__pdf_backend_pdfmark:x
2179   { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2180 }
2181 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2182 {
2183   \__pdf_backend_pdfmark:x
2184   { #1 << \exp_not:n {#2} >> /PUT }
2185 }
2186 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
```

```

2187 {
2188   \exp_args:Nx
2189   \__pdf_backend_object_write_fstream:nnn {#1} #2
2190 }
2191 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2192 {
2193   \__kernel_backend_postscript:n
2194   {
2195     SDict ~ begin ~
2196     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2197     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2198     end
2199   }
2200 }
2201 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2202 {
2203   \exp_args:Nx
2204   \__pdf_backend_object_write_stream:nnn {#1} #2
2205 }
2206 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2207 {
2208   \__kernel_backend_postscript:n
2209   {
2210     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2211     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2212   }
2213 }

```

(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.

```

2214 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2215 {
2216   \int_gincr:N \g__pdf_backend_object_int
2217   \__pdf_backend_pdfmark:x
2218   {
2219     /objdef ~ { pdf.obj \int_use:N \g__pdf_backend_object_int }
2220     /type
2221     \str_case:nn
2222     {#1}
2223     {
2224       { array } { /array }
2225       { dict } { /dict }
2226       { fstream } { /stream }
2227       { stream } { /stream }
2228     }
2229     /OBJ
2230   }
2231   \exp_args:Nnx \use:c { __pdf_backend_object_write_ #1 :nn }
2232   { { pdf.obj \int_use:N \g__pdf_backend_object_int } } {#2}
2233 }
2234 \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.

```
2235 \cs_new:Npn \__pdf_backend_object_last:
2236 { { 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.

```
2237 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2238 { { 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.

```
2239 \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.

```
2240 \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.

```
2241 \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.

```
2242 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2243 {
2244   \exp_args:Nf \__pdf_backend_annotation_aux:nnnn
2245     { \dim_eval:n {#1} } {#2} {#3} {#4}
2246 }
2247 \cs_new_protected:Npn \__pdf_backend_annotation_aux:nnnn #1#2#3#4
2248 {
2249   \box_move_down:nn {#3}
2250   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } } }
2251   \box_move_up:nn {#2}
2252   {
2253     \hbox:n
2254     {
2255       \__kernel_kern:n {#1}
2256       \__kernel_backend_postscript:n { pdf.save.ur }
2257       \__kernel_kern:n { -#1 }
2258     }
2259   }
2260 }
```

```

2259     }
2260     \int_gincr:N \g__pdf_backend_object_int
2261     \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2262     \__pdf_backend_pdfmark:x
2263     {
2264         /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2265         pdf.rect
2266         #4 ~
2267         /ANN
2268     }
2269 }

```

(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.

```

2270 \cs_new:Npn \__pdf_backend_annotation_last:
2271 { { 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.

```

2272 \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.

```

2273 \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.

```

2274 \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.

```

2275 \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.

```

2276 \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.

```

2277 \tl_new:N \l__pdf_breaklink_pdfmark_tl
2278 \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.

```

2279 \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.

2280 `\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`.

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

Taking the idea of `evenboxes` from `hydvips`, 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 `hydvips` 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 `hydvips`.

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

```
2281 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2282 {
2283   \_pdf_backend_link_begin:nw
2284   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2285 }
2286 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2287 { \_pdf_backend_link_begin:nw {#1#2} }
2288 \cs_new_protected:Npn \_pdf_backend_link_begin:nw #1
2289 {
2290   \bool_if:NF \g__pdf_backend_link_bool
2291   { \_pdf_backend_link_begin_aux:nw {#1} }
2292 }
```

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

```
2293 \cs_new_protected:Npn \_pdf_backend_link_begin_aux:nw #1
2294 {
2295   \bool_gset_true:N \g__pdf_backend_link_bool
2296   \_kernel_backend_postscript:n
2297   { /pdf.link.dict ( #1 ) def }
2298   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2299   \_pdf_backend_link_sf_save:
2300   \mode_if_math:TF
2301   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2302   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2303   \hbox_set:Nw \l__pdf_backend_content_box
2304   \_pdf_backend_link_sf_restore:
2305   \bool_if:NT \g__pdf_backend_link_math_bool
2306   { \c_math_toggle_token }
2307 }
2308 \cs_new_protected:Npn \_pdf_backend_link_end:
```

```

2309 {
2310   \bool_if:NT \g__pdf_backend_link_bool
2311   { \__pdf_backend_link_end_aux: }
2312 }
2313 \cs_new_protected:Npn \__pdf_backend_link_end_aux:
2314 {
2315   \bool_if:NT \g__pdf_backend_link_math_bool
2316   { \c_math_toggle_token }
2317   \__pdf_backend_link_sf_save:
2318   \hbox_set_end:
2319   \__pdf_backend_link_minima:
2320   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2321   \exp_args:Nx \__pdf_backend_link_outerbox:n
2322   {
2323     \int_if_odd:nTF { \value { page } }
2324     { \oddsidemargin }
2325     { \evensidemargin }
2326   }
2327   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2328   { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2329   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2330   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2331   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2332   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2333   {
2334     \hbox:n
2335     { \__kernel_backend_postscript:n { pdf.save.linkur } }
2336   }
2337   \int_gincr:N \g__pdf_backend_object_int
2338   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2339   \__kernel_backend_postscript:x
2340   {
2341     mark
2342     /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2343     \g__pdf_backend_link_dict_tl \c_space_tl
2344     pdf.rect
2345     /ANN ~ \l__pdf_breaklink_pdfmark_tl
2346   }
2347   \__pdf_backend_link_sf_restore:
2348   \bool_gset_false:N \g__pdf_backend_link_bool
2349 }
2350 \cs_new_protected:Npn \__pdf_backend_link_minima:
2351 {
2352   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2353   \__kernel_backend_postscript:x
2354   {
2355     /pdf.linkdp.pad ~
2356     \dim_to_decimal:n
2357     {
2358       \dim_max:nn
2359       {
2360         \box_dp:N \l__pdf_backend_model_box
2361         - \box_dp:N \l__pdf_backend_content_box
2362       }

```

```

2363         { Opt }
2364     } ~
2365     pdf.pt.dvi ~ def
2366 /pdf.linkht.pad ~
2367 \dim_to_decimal:n
2368 {
2369     \dim_max:nn
2370     {
2371         \box_ht:N \l__pdf_backend_model_box
2372         - \box_ht:N \l__pdf_backend_content_box
2373     }
2374     { Opt }
2375 } ~
2376 pdf.pt.dvi ~ def
2377 }
2378 }
2379 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2380 {
2381     \__kernel_backend_postscript:x
2382     {
2383         /pdf.outerbox
2384         [
2385             \dim_to_decimal:n {#1} ~
2386             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2387             \dim_to_decimal:n { #1 + \textwidth } ~
2388             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2389         ]
2390         [ exch { pdf.pt.dvi } forall ] def
2391     /pdf.baselineskip ~
2392     \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2393     { pdf.pt.dvi ~ def }
2394     { pop ~ pop }
2395     ifelse
2396 }
2397 }
2398 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2399 {
2400     \int_gset:Nn \g__pdf_backend_link_sf_int
2401     {
2402         \mode_if_horizontal:TF
2403         { \tex_spacefactor:D }
2404         { 0 }
2405     }
2406 }
2407 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2408 {
2409     \mode_if_horizontal:T
2410     {
2411         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2412         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2413     }
2414 }

```

(End definition for __pdf_backend_link_begin_goto:nnw 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.

```

2415 \use_none:n
2416 {
2417   \cs_if_exist:NT \@makecol@hook
2418   {
2419     \tl_put_right:Nn \@makecol@hook
2420     {
2421       \box_if_empty:NF \@cclv
2422       {
2423         \vbox_set:Nn \@cclv
2424         {
2425           \__kernel_backend_postscript:n
2426           {
2427             pdf.globaldict /pdf.brokenlink.rect ~ known
2428             { pdf.bordertracking.continue }
2429             if
2430             }
2431             \vbox_unpack_drop:N \@cclv
2432             \__kernel_backend_postscript:n
2433             { pdf.bordertracking.endpage }
2434           }
2435         }
2436       }
2437       \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2438       \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2439       \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2440     }
2441   }

```

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

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

```

2442 \cs_new:Npn \__pdf_backend_link_last:
2443 { { 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.

```

2444 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2445 {
2446   \__kernel_backend_postscript:x
2447   {
2448     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2449   }
2450 }

```

(End definition for `__pdf_backend_link_margin:n`)

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


```

2451 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2452 {
2453   \__kernel_backend_postscript:n { pdf.dest.anchor }
2454   \__pdf_backend_pdfmark:x
2455   {
2456     /View
2457     [
2458       \str_case:nnF {#2}
2459       {
2460         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2461         { fit } { /Fit }
2462         { fitb } { /FitB }
2463         { fitbh } { /FitBH ~ pdf.dest.y }
2464         { fitbv } { /FitBV ~ pdf.dest.x }
2465         { fith } { /FitH ~ pdf.dest.y }
2466         { fitv } { /FitV ~ pdf.dest.x }
2467         { fitr } { /Fit }
2468       }
2469       {
2470         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2471       }
2472     ]
2473     /Dest ( \exp_not:n {#1} ) cvn
2474     /DEST
2475   }
2476 }
2477 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2478 {
2479   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2480   { \dim_eval:n {#2} } {#1} {#3} {#4}
2481 }
2482 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2483 {
2484   \vbox_to_zero:n
2485   {
2486     \__kernel_kern:n {#4}
2487     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2488     \tex_vss:D
2489   }
2490   \__kernel_kern:n {#1}
2491   \vbox_to_zero:n
2492   {
2493     \__kernel_kern:n { -#3 }
2494     \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2495     \tex_vss:D
2496   }
2497   \__kernel_kern:n { -#1 }
2498   \__pdf_backend_pdfmark:n
2499   {
2500     /View
2501     [
2502       /FitR ~
2503       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2504       pdf.urx ~ pdf.ury ~ pdf.dest2device

```

```

2505     ]
2506     /Dest ( #2 ) cvn
2507     /DEST
2508   }
2509 }

```

(End definition for `_pdf_backend_destination:nn`, `_pdf_backend_destination:nnnn`, and `_pdf_backend_destination_aux:nnnn`.)

6.2.4 Structure

Doable for the usual ps2pdf method.

```

\_pdf_backend_compresslevel:n
\_pdf_backend_compress_objects:n
2510 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2511 {
2512   \int_compare:nNnT {#1} = 0
2513   {
2514     \__kernel_backend_literal_postscript:n
2515     {
2516       /setdistillerparams ~ where
2517       { pop << /CompressPages ~ false >> setdistillerparams }
2518       if
2519     }
2520   }
2521 }
2522 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2523 {
2524   \bool_if:nF {#1}
2525   {
2526     \__kernel_backend_literal_postscript:n
2527     {
2528       /setdistillerparams ~ where
2529       { pop << /CompressStreams ~ false >> setdistillerparams }
2530       if
2531     }
2532   }
2533 }

```

(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
2534 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2535 {
2536   \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2537 }
2538 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2539 {
2540   \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2541 }

```

(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:
2542 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2543 \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

Simple wrappers.

```

2544 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2545   { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2546 \cs_new_protected:Npn \__pdf_backend_emc:
2547   { \__pdf_backend_pdfmark:n { /EMC } }

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

2548 </dvips>

```

6.3 LuaTeX and pdfTeX backend

2549 <*luatex | pdftex>

6.3.1 Annotations

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

```

2550 \cs_new_protected:Npn \__pdf_backend_annotation:nnnn #1#2#3#4
2551   {
2552     <*luatex>
2553       \tex_pdfextension:D annot ~
2554     </luatex>
2555     <*pdftex>
2556       \tex_pdfannot:D
2557     </pdftex>
2558       width ~ \dim_eval:n {#1} ~
2559       height ~ \dim_eval:n {#2} ~
2560       depth ~ \dim_eval:n {#3} ~
2561       {#4}
2562   }

```

(End definition for __pdf_backend_annotation:nnnn.)

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.

```

2563 \cs_new:Npx \__pdf_backend_annotation_last:
2564   {
2565     \exp_not:N \int_value:w
2566     <*luatex>
2567       \exp_not:N \tex_pdffeedback:D lastannot ~
2568     </luatex>
2569     <*pdftex>
2570       \exp_not:N \tex_pdflastannot:D
2571     </pdftex>
2572     \c_space_tl 0 ~ R
2573   }

```

(End definition for __pdf_backend_annotation_last:.)

Links are all created using the same internals.

```

2574 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2
2575   { \__pdf_backend_link_begin:nnnw {#1} { goto~name } {#2} }
2576 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2

```

```

2577 { \_pdf_backend_link_begin:nnnw {#1} { user } {#2} }
2578 \cs_new_protected:Npn \_pdf_backend_link_begin:nnnw #1#2#3
2579 {
2580   \*luatex
2581   \tex_pdfextension:D startlink ~
2582   \*pdftex
2583   \tex_pdfstartlink:D
2584   \*pdftex
2585   attr {#1}
2586   #2 {#3}
2587 }
2588 \cs_new_protected:Npn \_pdf_backend_link_end:
2589 {
2590   \*luatex
2591   \tex_pdfextension:D endlink \scan_stop:
2592   \*pdftex
2593   \tex_pdfendlink:D
2594   \*pdftex
2595 }
2596
2597

```

(End definition for _pdf_backend_link_begin_goto:nnw and others.)

_pdf_backend_link_last: Formatted for direct use.

```

2598 \cs_new:Npx \_pdf_backend_link_last:
2599 {
2600   \exp_not:N \int_value:w
2601   \*luatex
2602   \exp_not:N \tex_pdffeedback:D lastlink ~
2603   \*pdftex
2604   \exp_not:N \tex_pdflastlink:D
2605   \*pdftex
2606   \c_space_tl 0 ~ R
2607 }
2608

```

(End definition for _pdf_backend_link_last:.)

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

```

2609 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2610 {
2611   \*luatex
2612   \tex_pdfvariable:D linkmargin
2613   \*pdftex
2614   \tex_pdflinkmargin:D
2615   \*pdftex
2616   \dim_eval:n {#1} \scan_stop:
2617 }
2618

```

(End definition for _pdf_backend_link_margin:n.)

`_pdf_backend_destination:nn`
`_pdf_backend_destination:nnnn`

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

```

2619 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2620 {
2621   \*luatex
2622     \tex_pdfextension:D dest ~
2623   \*pdfTeX
2624     \tex_pdfdest:D
2625   \*pdfTeX
2626     name {#1}
2627     \str_case:nnF {#2}
2628     {
2629       { xyz } { xyz }
2630       { fit } { fit }
2631       { fitb } { fitb }
2632       { fitbh } { fitbh }
2633       { fitbv } { fitbv }
2634       { fith } { fith }
2635       { fitv } { fitv }
2636       { fitr } { fitr }
2637     }
2638     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2639     \scan_stop:
2640   }
2641 }
2642 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2643 {
2644   \*luatex
2645     \tex_pdfextension:D dest ~
2646   \*pdfTeX
2647     \tex_pdfdest:D
2648   \*pdfTeX
2649     name {#1}
2650     fitr ~
2651     width \dim_eval:n {#2} ~
2652     height \dim_eval:n {#3} ~
2653     depth \dim_eval:n {#4} \scan_stop:
2654   }
2655 }

```

(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`

```

2656 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2657 {
2658   \*luatex
2659     \tex_pdfextension:D catalog
2660   \*pdfTeX
2661     \tex_pdfcatalog:D
2662   \*pdfTeX
2663   }

```

```

2664     { / #1 ~ #2 }
2665   }
2666   \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2667   {
2668     <*luatex>
2669     \tex_pdfextension:D info
2670   </luatex>
2671   <*pdftex>
2672     \tex_pdfinfo:D
2673   </pdftex>
2674     { / #1 ~ #2 }
2675   }

```

(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.

```

2676 \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
2677 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2678 {
2679   <*luatex>
2680   \tex_pdfextension:D obj ~
2681 </luatex>
2682 <*pdftex>
2683   \tex_pdfobj:D
2684 </pdftex>
2685   reserveobjnum ~
2686   \int_const:cn
2687   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2688 <*luatex>
2689   { \tex_pdffeedback:D lastobj }
2690 </luatex>
2691 <*pdftex>
2692   { \tex_pdflastobj:D }
2693 </pdftex>
2694   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2695 }
2696 \cs_new:Npn \__pdf_backend_object_ref:n #1
2697 { \int_use:c { c__pdf_backend_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

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

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

```

\__pdf_backend_object_write:nx
\__pdf_exp_not_i:nn
\__pdf_exp_not_ii:nn
2698 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2699 {
2700 <*luatex>
2701   \tex_immediate:D \tex_pdfextension:D obj ~
2702 </luatex>
2703 <*pdftex>
2704   \tex_immediate:D \tex_pdfobj:D

```

```

2705 </pdfTeX>
2706     useobjnum ~
2707     \int_use:c
2708     { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2709     \str_case_e:nn
2710     { \prop_item:Nn \g__pdf_backend_object_prop {#1} }
2711     {
2712       { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2713       { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2714       { fstream }
2715       {
2716         stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2717         file ~ { \__pdf_exp_not_ii:nn #2 }
2718       }
2719       { stream }
2720       {
2721         stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2722         { \__pdf_exp_not_ii:nn #2 }
2723       }
2724     }
2725   }
2726   \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2727   \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2728   \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
2729 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2730 {
2731   <*luatex>
2732   \tex_immediate:D \tex_pdfextension:D obj ~
2733   </luatex>
2734   <*pdfTeX>
2735   \tex_immediate:D \tex_pdfobj:D
2736   </pdfTeX>
2737   \str_case:nn
2738   {#1}
2739   {
2740     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2741     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2742     { fstream }
2743     {
2744       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2745       file ~ { \__pdf_exp_not_ii:nn #2 }
2746     }
2747     { stream }
2748     {
2749       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2750       { \__pdf_exp_not_ii:nn #2 }
2751     }
2752   }
2753 }
2754 \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.

```

2755 \cs_new:Npx \_pdf_backend_object_last:
2756 {
2757   \exp_not:N \int_value:w
2758   <*luatex>
2759   \exp_not:N \tex_pdffeedback:D lastobj ~
2760   </luatex>
2761   <*pdftex>
2762   \exp_not:N \tex_pdflastobj:D
2763   </pdftex>
2764   \c_space_tl 0 ~ R
2765 }

```

(End definition for _pdf_backend_object_last:.)

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

```

2766 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2767 {
2768   \exp_not:N \int_value:w
2769   <*luatex>
2770   \exp_not:N \tex_pdffeedback:D pageref
2771   </luatex>
2772   <*pdftex>
2773   \exp_not:N \tex_pdfpageref:D
2774   </pdftex>
2775   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2776 }

```

(End definition for _pdf_backend_pageobject_ref:n.)

6.3.4 Structure

_pdf_backend_compresslevel:n Simply pass data to the engine.

```

\_pdf_backend_compresslevel:n
\_pdf_backend_compress_objects:n
\_pdf_backend_objcompresslevel:n
2777 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2778 {
2779   \tex_global:D
2780   <*luatex>
2781   \tex_pdfvariable:D compresslevel
2782   </luatex>
2783   <*pdftex>
2784   \tex_pdfcompresslevel:D
2785   </pdftex>
2786   \int_value:w \int_eval:n {#1} \scan_stop:
2787 }
2788 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2789 {
2790   \bool_if:nTF {#1}
2791   { \_pdf_backend_objcompresslevel:n { 2 } }
2792   { \_pdf_backend_objcompresslevel:n { 0 } }
2793 }
2794 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2795 {

```



```

2796     \tex_global:D
2797 <*luatex>
2798     \tex_pdfvariable:D objcompresslevel
2799 </luatex>
2800 <*pdftex>
2801     \tex_pdfobjcompresslevel:D
2802 </pdftex>
2803     #1 \scan_stop:
2804 }

(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

```

2805 \cs_new_protected:Npx \__pdf_backend_version_major_gset:n #1
2806 {
2807 <*luatex>
2808     \int_compare:nNnT \tex_luatexversion:D > { 106 }
2809     {
2810         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2811         \exp_not:N \int_eval:n {#1} \scan_stop:
2812     }
2813 </luatex>
2814 <*pdftex>
2815     \cs_if_exist:NT \tex_pdfmajorversion:D
2816     {
2817         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2818         \exp_not:N \int_eval:n {#1} \scan_stop:
2819     }
2820 </pdftex>
2821 }
2822 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2823 {
2824     \tex_global:D
2825 <*luatex>
2826     \tex_pdfvariable:D minorversion
2827 </luatex>
2828 <*pdftex>
2829     \tex_pdfminorversion:D
2830 </pdftex>
2831     \int_eval:n {#1} \scan_stop:
2832 }

```

(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:

```

2833 \cs_new:Npx \__pdf_backend_version_major:
2834 {
2835 <*luatex>
2836     \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2837     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2838     { 1 }
2839 </luatex>
2840 <*pdftex>
2841     \cs_if_exist:NTF \tex_pdfmajorversion:D

```

```

2842     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2843     { 1 }
2844   \</pdfTeX>
2845 }
2846 \cs_new:Npn \__pdf_backend_version_minor:
2847 {
2848   \tex_the:D
2849   \*luatex
2850   \tex_pdfvariable:D minorversion
2851   \</luatex>
2852   \*pdfTeX
2853   \tex_pdfminorversion:D
2854   \</pdfTeX>
2855 }

```

(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>.

```

2856 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2857 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2858 \cs_new_protected:Npn \__pdf_backend_emc:
2859 { \__kernel_backend_literal_page:n { EMC } }

```

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

```

2860 \</luatex | pdfTeX>

```

6.4 dvipdfmx backend

```

2861 \*dvipdfmx | xetex>

```

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

```

\__pdf_backend:x
2862 \cs_new_protected:Npx \__pdf_backend:n #1
2863 { \__kernel_backend_literal:n { pdf: #1 } }
2864 \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
2865 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2866 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2867 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2868 { \__pdf_backend:n { docinfo << /#1 ~ #2 >> } }

```

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

6.4.2 Objects

For tracking objects to allow finalisation.

```
2869 \int_new:N \g__pdf_backend_object_int
2870 \prop_new:N \g__pdf_backend_object_prop
```

(End definition for \g__pdf_backend_object_int and \g__pdf_backend_object_prop.)

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

```
2871 \cs_new_protected:Npn \__pdf_backend_object_new:nn #1#2
2872 {
2873   \int_gincr:N \g__pdf_backend_object_int
2874   \int_const:cn
2875   { c__pdf_backend_object_ \tl_to_str:n {#1} _int }
2876   { \g__pdf_backend_object_int }
2877   \prop_gput:Nnn \g__pdf_backend_object_prop {#1} {#2}
2878 }
2879 \cs_new:Npn \__pdf_backend_object_ref:n #1
2880 { @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.

```
2881 \cs_new_protected:Npn \__pdf_backend_object_write:nn #1#2
2882 {
2883   \exp_args:Nx \__pdf_backend_object_write:nnn
2884   { \prop_item:Nn \g__pdf_backend_object_prop {#1} } {#1} {#2}
2885 }
2886 \cs_generate_variant:Nn \__pdf_backend_object_write:nn { nx }
2887 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2888 {
2889   \use:c { __pdf_backend_object_write_ #1 :nn }
2890   { \__pdf_backend_object_ref:n {#2} } {#3}
2891 }
2892 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2893 {
2894   \__pdf_backend:x
2895   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2896 }
2897 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2898 {
2899   \__pdf_backend:x
2900   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2901 }
2902 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2903 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2904 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2905 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2906 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2907 {
2908   \__pdf_backend:x
2909   {
2910     #1 stream ~ #2 ~
2911     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2912   }
```

```
2913 }
```

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

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

```
\_pdf_backend_object_now:nx
2914 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2915 {
2916   \int_gincr:N \g__pdf_backend_object_int
2917   \exp_args:Nnx \use:c { \_pdf_backend_object_write_ #1 :nn }
2918   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2919   {#2}
2920 }
2921 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }
```

(End definition for `_pdf_backend_object_now:nn`.)

`_pdf_backend_object_last:`

```
2922 \cs_new:Npn \_pdf_backend_object_last:
2923 { @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.

```
2924 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2925 { @page #1 }
```

(End definition for `_pdf_backend_pageobject_ref:n`.)

6.4.3 Annotations

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

```
2926 \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.

```
2927 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2928 {
2929   \int_gincr:N \g__pdf_backend_object_int
2930   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2931   \_pdf_backend:x
2932   {
2933     ann ~ @pdf.obj \int_use:N \g__pdf_backend_object_int \c_space_tl
2934     width ~ \dim_eval:n {#1} ~
2935     height ~ \dim_eval:n {#2} ~
2936     depth ~ \dim_eval:n {#3} ~
2937     << /Type /Annot #4 >>
2938   }
2939 }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

`_pdf_backend_annotation_last:`

```
2940 \cs_new:Npn \_pdf_backend_annotation_last:
2941 { @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.

2942 \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

2943 \cs_new_protected:Npn _pdf_backend_link_begin_goto:nnw #1#2

2944 { _pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D (#2) >> } }

_pdf_backend_link_begin:n

2945 \cs_new_protected:Npn _pdf_backend_link_begin_user:nnw #1#2

2946 { _pdf_backend_link_begin:n {#1#2} }

2947 \cs_new_protected:Npx _pdf_backend_link_begin:n #1

2948 {

2949 \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }

2950 {

2951 \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int

2952 }

2953 _pdf_backend:x

2954 {

2955 bann ~

2956 \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }

2957 {

2958 @pdf.lnk

2959 \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int

2960 \c_space_tl

2961 }

2962 <<

2963 /Type /Annot

2964 #1

2965 >>

2966 }

2967 }

2968 \cs_new_protected:Npn _pdf_backend_link_end:

2969 { _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.

2970 \cs_new:Npx _pdf_backend_link_last:

2971 {

2972 \int_compare:nNnF \c_kernel_sys_dvipdfmx_version_int < { 20201111 }

2973 {

2974 @pdf.lnk

2975 \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int

2976 }

2977 }

(End definition for _pdf_backend_link_last:.)

_pdf_backend_link_margin:n Pass to dvipdfmx.

2978 \cs_new_protected:Npn _pdf_backend_link_margin:n #1

2979 { _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.

```

2980 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2981 {
2982   \_pdf_backend:x
2983   {
2984     dest ~ ( \exp_not:n {#1} )
2985     [
2986       @thispage
2987       \str_case:nnF {#2}
2988       {
2989         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2990         { fit } { /Fit }
2991         { fitb } { /FitB }
2992         { fitbh } { /FitBH }
2993         { fitbv } { /FitBV ~ @xpos }
2994         { fith } { /FitH ~ @ypos }
2995         { fitv } { /FitV ~ @xpos }
2996         { fitr } { /Fit }
2997       }
2998       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2999     ]
3000   }
3001 }
3002 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
3003 {
3004   \exp_args:Ne \_pdf_backend_destination_aux:nnnn
3005   { \dim_eval:n {#2} } {#1} {#3} {#4}
3006 }
3007 \cs_new_protected:Npn \_pdf_backend_destination_aux:nnnn #1#2#3#4
3008 {
3009   \vbox_to_zero:n
3010   {
3011     \_kernel_kern:n {#4}
3012     \hbox:n
3013     {
3014       \_pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
3015       \_pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
3016     }
3017     \tex_vss:D
3018   }
3019   \_kernel_kern:n {#1}
3020   \vbox_to_zero:n
3021   {
3022     \_kernel_kern:n { -#3 }
3023     \hbox:n
3024     {
3025       \_pdf_backend:n
3026       {
3027         dest ~ (#2)
3028         [
3029           @thispage

```

```

3030             /FitR ~
3031             @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
3032             @xpos ~ @ypos
3033         ]
3034     }
3035 }
3036 \tex_vss:D
3037 }
3038 \__kernel_kern:n { -#1 }
3039 }

```

(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.

```

3040 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
3041 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
3042 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
3043 {
3044   \bool_if:nF {#1}
3045   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
3046 }

```

(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.

```

3047 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
3048 {
3049   \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
3050   \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
3051 }
3052 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
3053 {
3054   \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
3055   \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
3056 }

```

(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.

```

3057 \cs_new:Npn \__pdf_backend_version_major: { 1 }
3058 \cs_new:Npn \__pdf_backend_version_minor: { 5 }

```

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

6.4.5 Marked content

`__pdf_backend_bdc:nn`
`__pdf_backend_emc:`

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

```

3059 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
3060 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
3061 \cs_new_protected:Npn \__pdf_backend_emc:
3062 { \__kernel_backend_literal_page:n { EMC } }

```

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

3063 </dvipdfmx | xetex>

6.5 dvisvgm backend

3064 <*dvisvgm>

6.5.1 Catalogue entries

No-op.

_pdf_backend_catalog_gput:nn
_pdf_backend_info_gput:nn

3065 \cs_new_protected:Npn _pdf_backend_catalog_gput:nn #1#2 { }

3066 \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

All no-ops here.

_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

3067 \cs_new_protected:Npn _pdf_backend_object_new:nn #1#2 { }

3068 \cs_new:Npn _pdf_backend_object_ref:n #1 { }

3069 \cs_new_protected:Npn _pdf_backend_object_write:nn #1#2 { }

3070 \cs_new_protected:Npn _pdf_backend_object_write:nx #1#2 { }

3071 \cs_new_protected:Npn _pdf_backend_object_now:nn #1#2 { }

3072 \cs_new_protected:Npn _pdf_backend_object_now:nx #1#2 { }

3073 \cs_new:Npn _pdf_backend_object_last: { }

3074 \cs_new:Npn _pdf_backend_pageobject_ref:n #1 { }

(End definition for _pdf_backend_object_new:nn and others.)

6.5.3 Structure

These are all no-ops.

_pdf_backend_compresslevel:n
_pdf_backend_compress_objects:n

3075 \cs_new_protected:Npn _pdf_backend_compresslevel:n #1 { }

3076 \cs_new_protected:Npn _pdf_backend_compress_objects:n #1 { }

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

Data not available!

_pdf_backend_version_major_gset:n
_pdf_backend_version_minor_gset:n

3077 \cs_new_protected:Npn _pdf_backend_version_major_gset:n #1 { }

3078 \cs_new_protected:Npn _pdf_backend_version_minor_gset:n #1 { }

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

Data not available!

_pdf_backend_version_major:
_pdf_backend_version_minor:

3079 \cs_new:Npn _pdf_backend_version_major: { -1 }

3080 \cs_new:Npn _pdf_backend_version_minor: { -1 }

(End definition for _pdf_backend_version_major: and _pdf_backend_version_minor:.)

More no-ops.

_pdf_backend_bdc:nn
 _pdf_backend_emc:

3081 \cs_new_protected:Npn _pdf_backend_bdc:nn #1#2 { }

3082 \cs_new_protected:Npn _pdf_backend_emc: { }

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

3083 </dvisvgm>

3084 </package>

7 l3backend-opacity Implementation

```

3085 <*package>
3086 <@@=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.

```

3087 <*dvips>

```

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

```

3088 \cs_new_protected:Npn \__opacity_backend_select:n #1
3089 {
3090   \exp_args:Nx \__opacity_backend_select_aux:n
3091   { \fp_eval:n { min(max(0,#1),1) } }
3092 }
3093 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3094 {
3095   \__opacity_backend:nnn {#1} { fill } { ca }
3096   \__opacity_backend:nnn {#1} { stroke } { CA }
3097 }
3098 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3099 {
3100   \__opacity_backend:xnn
3101   { \fp_eval:n { min(max(0,#1),1) } }
3102   { fill }
3103   { ca }
3104 }
3105 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3106 {
3107   \__opacity_backend:xnn
3108   { \fp_eval:n { min(max(0,#1),1) } }
3109   { stroke }
3110   { CA }
3111 }
3112 \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3113 {
3114   \__kernel_backend_postscript:n
3115   {
3116     product ~ (Ghostscript) ~ search
3117     {
3118       pop ~ pop ~ pop ~
3119       #1 ~ .set #2 constantalpha
3120     }
3121     {
3122       pop ~
3123       mark ~
3124       /#3 ~ #1

```

```

3125         /SetTransparency ~
3126         pdfmark
3127     }
3128     ifelse
3129 }
3130 }
3131 \cs_generate_variant:Nn \_opacity_backend:nnn { x }

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

3132 </dvips>
3133 <*dvipdfmx | luatex | pdftex | xetex>

```

\c__opacity_backend_stack_int Set up a stack.

```

3134 \bool_lazy_and:nnT
3135 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3136 { \pdfmanagement_if_active_p:}
3137 {
3138     \_kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3139     { page ~ direct } { /opacity 1 ~ gs }
3140     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3141     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3142 }

(End definition for \c__opacity_backend_stack_int.)

```

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

```

\l__opacity_backend_stroke_tl
3143 \tl_new:N \l__opacity_backend_fill_tl
3144 \tl_new:N \l__opacity_backend_stroke_tl

```

(End definition for \l__opacity_backend_fill_tl and \l__opacity_backend_stroke_tl.)

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

```

\_opacity_backend_select_aux:n
\_opacity_backend_reset:
3145 \cs_new_protected:Npn \_opacity_backend_select:n #1
3146 {
3147     \exp_args:Nx \_opacity_backend_select_aux:n
3148     { \fp_eval:n { min(max(0,#1),1) } }
3149 }
3150 \cs_new_protected:Npn \_opacity_backend_select_aux:n #1
3151 {
3152     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3153     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3154     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3155     { opacity #1 }
3156     { << /ca ~ #1 /CA ~ #1 >> }
3157     \_kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3158     { /opacity #1 ~ gs }
3159     \group_insert_after:N \_opacity_backend_reset:
3160 }
3161 \bool_lazy_and:nnF
3162 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3163 { \pdfmanagement_if_active_p:}
3164 {
3165     \cs_gset_protected:Npn \_opacity_backend_select_aux:n #1 { }
3166 }

```

```

3167 \cs_new_protected:Npn \__opacity_backend_reset:
3168 { \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int }

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

```

```

\__opacity_backend_fill:n
\__opacity_backend_stroke:n
  \__opacity_backend_fillstroke:nn
  \__opacity_backend_fillstroke:xx

```

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.

```

3169 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3170 {
3171   \__opacity_backend_fill_stroke:xx
3172   { \fp_eval:n { min(max(0,#1),1) } }
3173   \l__opacity_backend_stroke_tl
3174 }
3175 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3176 {
3177   \__opacity_backend_fill_stroke:xx
3178   \l__opacity_backend_fill_tl
3179   { \fp_eval:n { min(max(0,#1),1) } }
3180 }
3181 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3182 {
3183   \str_if_eq:nnTF {#1} {#2}
3184   { \__opacity_backend_select_aux:n {#1} }
3185   {
3186     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3187     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3188     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3189     { opacity.fill #1 }
3190     { << /ca ~ #1 >> }
3191     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3192     { opacity.stroke #1 }
3193     { << /CA ~ #2 >> }
3194     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3195     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3196     \group_insert_after:N \__opacity_backend_reset:
3197   }
3198 }
3199 \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.)

```

3200 </dviPDFmx | luatex | pdftex | xetex>
3201 <*dviPDFmx | xdvipdfmx>

```

```
\__opacity_backend_select:n
```

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

```

3202 \int_compare:nNnT \c__kernel_sys_dvipdfmx_version_int < { 20201111 }
3203 {
3204   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1
3205   {
3206     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3207     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3208     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3209     { opacity #1 }

```

```

3210         { << /ca ~ #1 /CA ~ #1 >> }
3211         \__kernel_backend_literal_pdf:n { /opacity #1 ~ gs }
3212     }
3213     \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3214     {
3215         \str_if_eq:nnTF {#1} {#2}
3216         { \__opacity_backend_select_aux:n {#1} }
3217         {
3218             \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3219             \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3220             \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3221             { opacity.fill #1 }
3222             { << /ca ~ #1 >> }
3223             \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3224             { opacity.stroke #1 }
3225             { << /CA ~ #2 >> }
3226             \__kernel_backend_literal_pdf:n
3227             { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3228         }
3229     }
3230 }

```

(End definition for __opacity_backend_select:n.)

```

3231 </dvipdfmx | xdvipdfmx>

```

```

3232 <*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
3233 \cs_new_protected:Npn \__opacity_backend_select:n #1
3234 { \__opacity_backend:nn {#1} { } }
3235 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3236 { \__opacity_backend:nn {#1} { fill- } }
3237 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3238 { \__opacity_backend:nn { {#1} } { stroke- } }
3239 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3240 { \__kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

```

(End definition for __opacity_backend_select:n and others.)

```

3241 </dvisvgm>

```

```

3242 </package>

```

8 l3backend-header Implementation

```

3243 <*dvips & header>

```

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

```

3244 /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.

```

3245 TeXDict begin
3246 /TeXcolorseparation { setcolor } def
3247 end

```

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

pdf.globaldict A small global dictionary for backend use.

```

3248 true setglobal
3249 /pdf.globaldict 4 dict def
3250 false setglobal

```

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

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

```

3251 /pdf.cvs { 65534 string cvs } def
3252 /pdf.dvi.pt { 72.27 mul Resolution div } def
3253 /pdf.pt.dvi { 72.27 div Resolution mul } def
3254 /pdf.rect.ht { dup 1 get neg exch 3 get add } def

```

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

pdf.linkmargin Settings which are defined up-front in SDict.

```

3255 /pdf.linkmargin { 1 pdf.pt.dvi } def
3256 /pdf.linkdp.pad { 0 } def
3257 /pdf.linkht.pad { 0 } def

```

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

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

```

3258 /pdf.rect
3259 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
3260 /pdf.save.ll
3261 {
3262     currentpoint
3263     /pdf.lly exch def
3264     /pdf.llx exch def
3265 }
3266 def
3267 /pdf.save.ur
3268 {
3269     currentpoint
3270     /pdf.ury exch def
3271     /pdf.urx exch def
3272 }
3273 def
3274 /pdf.save.linkll
3275 {

```

```

3276     currentpoint
3277     pdf.linkmargin add
3278     pdf.linkdp.pad add
3279     /pdf.lly exch def
3280     pdf.linkmargin sub
3281     /pdf.llx exch def
3282   }
3283   def
3284 /pdf.save.linkur
3285   {
3286     currentpoint
3287     pdf.linkmargin sub
3288     pdf.linkht.pad sub
3289     /pdf.ury exch def
3290     pdf.linkmargin add
3291     /pdf.urx exch def
3292   }
3293   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 3294 /pdf.dest.anchor
pdf.dev.y 3295   {
pdf.tmpa 3296     currentpoint exch
pdf.tmpb 3297     pdf.dvi.pt 72 add
pdf.tmpc 3298     /pdf.dest.x exch def
pdf.tmpd 3299     pdf.dvi.pt
3300     vsize 72 sub exch sub
3301     /pdf.dest.y exch def
3302   }
3303   def
3304 /pdf.dest.point
3305   { pdf.dest.x pdf.dest.y } def
3306 /pdf.dest2device
3307   {
3308     /pdf.dest.y exch def
3309     /pdf.dest.x exch def
3310     matrix currentmatrix
3311     matrix defaultmatrix
3312     matrix invertmatrix
3313     matrix concatmatrix
3314     cvx exec
3315     /pdf.dev.y exch def
3316     /pdf.dev.x exch def
3317     /pdf.tmpd exch def
3318     /pdf.tmpc exch def
3319     /pdf.tmpb exch def
3320     /pdf.tmpa exch def
3321     pdf.dest.x pdf.tmpa mul

```

```

3322     pdf.dest.y pdf.tmpc mul add
3323     pdf.dev.x add
3324 pdf.dest.x pdf.tmpb mul
3325     pdf.dest.y pdf.tmpd mul add
3326     pdf.dev.y add
3327 }
3328 def

```

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

```

pdf.bordertracking To know where a breakable link can go, we need to track the boundary rectangle. That
pdf.bordertracking.begin can be done by hooking into a and x operations: those names have to be retained. The
pdf.bordertracking.end boundary is stored at the end of the operation. Special effort is needed at the start and
pdf.leftboundary end of pages (or rather galleys), such that everything works properly.
pdf.rightboundary
pdf.brokenlink.rect 3329 /pdf.bordertracking false def
pdf.brokenlink.skip 3330 /pdf.bordertracking.begin
pdf.brokenlink.dict 3331 {
pdf.bordertracking.endpage 3332   SDict /pdf.bordertracking true put
pdf.bordertracking.continue 3333   SDict /pdf.leftboundary undef
pdf.originx 3334   SDict /pdf.rightboundary undef
pdf.originy 3335   /a where
3336   {
3337     /a
3338     {
3339       currentpoint pop
3340       SDict /pdf.rightboundary known dup
3341       {
3342         SDict /pdf.rightboundary get 2 index lt
3343         { not }
3344         if
3345       }
3346       if
3347       { pop }
3348       { SDict exch /pdf.rightboundary exch put }
3349       ifelse
3350       moveto
3351       currentpoint pop
3352       SDict /pdf.leftboundary known dup
3353       {
3354         SDict /pdf.leftboundary get 2 index gt
3355         { not }
3356         if
3357       }
3358       if
3359       { pop }
3360       { SDict exch /pdf.leftboundary exch put }
3361       ifelse
3362     }
3363     put
3364   }
3365   if
3366 }
3367 def
3368 /pdf.bordertracking.end

```

```

3369 {
3370   /a where { /a { moveto } put } if
3371   /x where { /x { 0 exch rmoveto } put } if
3372   SDict /pdf.leftboundary known
3373     { pdf.outerbox 0 pdf.leftboundary put }
3374   if
3375   SDict /pdf.rightboundary known
3376     { pdf.outerbox 2 pdf.rightboundary put }
3377   if
3378   SDict /pdf.bordertracking false put
3379 }
3380 def
3381 /pdf.bordertracking.endpage
3382 {
3383   pdf.bordertracking
3384   {
3385     pdf.bordertracking.end
3386     true setglobal
3387     pdf.globaldict
3388       /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3389     pdf.globaldict
3390       /pdf.brokenlink.skip pdf.baselineskip put
3391     pdf.globaldict
3392       /pdf.brokenlink.dict
3393         pdf.link.dict pdf.cvs put
3394     false setglobal
3395     mark pdf.link.dict cvx exec /Rect
3396     [
3397       pdf.llx
3398       pdf.lly
3399       pdf.outerbox 2 get pdf.linkmargin add
3400       currentpoint exch pop
3401       pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3402     ]
3403     /ANN pdf.pdfmark
3404   }
3405   if
3406 }
3407 def
3408 /pdf.bordertracking.continue
3409 {
3410   /pdf.link.dict pdf.globaldict
3411   /pdf.brokenlink.dict get def
3412   /pdf.outerbox pdf.globaldict
3413   /pdf.brokenlink.rect get def
3414   /pdf.baselineskip pdf.globaldict
3415   /pdf.brokenlink.skip get def
3416   pdf.globaldict dup dup
3417   /pdf.brokenlink.dict undef
3418   /pdf.brokenlink.skip undef
3419   /pdf.brokenlink.rect undef
3420   currentpoint
3421   /pdf.originy exch def
3422   /pdf.originx exch def

```



```

3423     /a where
3424     {
3425         /a
3426         {
3427             moveto
3428             SDict
3429             begin
3430                 currentpoint pdf.originy ne exch
3431                 pdf.originx ne or
3432                 {
3433                     pdf.save.linkll
3434                     /pdf.lly
3435                     pdf.lly pdf.outerbox 1 get sub def
3436                     pdf.bordertracking.begin
3437                 }
3438                 if
3439                 end
3440             }
3441             put
3442         }
3443     if
3444     /x where
3445     {
3446         /x
3447         {
3448             0 exch rmoveto
3449             SDict
3450             begin
3451                 currentpoint
3452                 pdf.originy ne exch pdf.originx ne or
3453                 {
3454                     pdf.save.linkll
3455                     /pdf.lly
3456                     pdf.lly pdf.outerbox 1 get sub def
3457                     pdf.bordertracking.begin
3458                 }
3459                 if
3460                 end
3461             }
3462             put
3463         }
3464     if
3465 }
3466 def

```

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

pdf.breaklink Dealing with link breaking itself has multiple stage. The first step is to find the **Rect** entry 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.

```

3467 /pdf.breaklink
3468 {

```

```

3469 pop
3470 counttomark 2 mod 0 eq
3471 {
3472     counttomark /pdf.count exch def
3473     {
3474         pdf.count 0 eq { exit } if
3475         counttomark 2 roll
3476         1 index /Rect eq
3477         {
3478             dup 4 array copy
3479             dup dup
3480             1 get
3481             pdf.outerbox pdf.rect.ht
3482             pdf.linkmargin 2 mul add sub
3483             3 exch put
3484             dup
3485             pdf.outerbox 2 get
3486             pdf.linkmargin add
3487             2 exch put
3488             dup dup
3489             3 get
3490             pdf.outerbox pdf.rect.ht
3491             pdf.linkmargin 2 mul add add
3492             1 exch put
3493             /pdf.currentrect exch def
3494             pdf.breaklink.write
3495             {
3496                 pdf.currentrect
3497                 dup
3498                 pdf.outerbox 0 get
3499                 pdf.linkmargin sub
3500                 0 exch put
3501                 dup
3502                 pdf.outerbox 2 get
3503                 pdf.linkmargin add
3504                 2 exch put
3505                 dup dup
3506                 1 get
3507                 pdf.baselineskip add
3508                 1 exch put
3509                 dup dup
3510                 3 get
3511                 pdf.baselineskip add
3512                 3 exch put
3513                 /pdf.currentrect exch def
3514                 pdf.breaklink.write
3515             }
3516             1 index 3 get
3517             pdf.linkmargin 2 mul add
3518             pdf.outerbox pdf.rect.ht add
3519             2 index 1 get sub
3520             pdf.baselineskip div round cvi 1 sub
3521             exch
3522             repeat

```

```

3523         pdf.currentrect
3524         dup
3525             pdf.outerbox 0 get
3526             pdf.linkmargin sub
3527             0 exch put
3528         dup dup
3529             1 get
3530             pdf.baselineskip add
3531             1 exch put
3532         dup dup
3533             3 get
3534             pdf.baselineskip add
3535             3 exch put
3536         dup 2 index 2 get 2 exch put
3537         /pdf.currentrect exch def
3538         pdf.breaklink.write
3539         SDict /pdf.pdfmark.good false put
3540         exit
3541     }
3542     { pdf.count 2 sub /pdf.count exch def }
3543     ifelse
3544 }
3545 loop
3546 }
3547 if
3548 /ANN
3549 }
3550 def
3551 /pdf.breaklink.write
3552 {
3553     counttomark 1 sub
3554     index /_objdef eq
3555     {
3556         counttomark -2 roll
3557         dup wcheck
3558         {
3559             readonly
3560             counttomark 2 roll
3561         }
3562         { pop pop }
3563     } ifelse
3564 }
3565 if
3566 counttomark 1 add copy
3567 pop pdf.currentrect
3568 /ANN pdfmark
3569 }
3570 def

```

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

```

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

```

The business end of breaking links starts by hooking into pdfmarks. Unlike hypdvips, we avoid altering any links we have not created by using a copy of the core pdfmarks function. Only mark types which are known are altered. At present, this is purely ANN

marks, which are measured relative to the size of the baseline skip. If they are more than one apparent line high, breaking is applied.

```

3571 /pdf.pdfmark
3572 {
3573   SDict /pdf.pdfmark.good true put
3574   dup /ANN eq
3575   {
3576     pdf.pdfmark.store
3577     pdf.pdfmark.dict
3578     begin
3579       Subtype /Link eq
3580       currentdict /Rect known and
3581       SDict /pdf.outerbox known and
3582       SDict /pdf.baselineskip known and
3583       {
3584         Rect 3 get
3585         pdf.linkmargin 2 mul add
3586         pdf.outerbox pdf.rect.ht add
3587         Rect 1 get sub
3588         pdf.baselineskip div round cvi 0 gt
3589         { pdf.breaklink }
3590         if
3591       }
3592       if
3593       end
3594       SDict /pdf.outerbox undef
3595       SDict /pdf.baselineskip undef
3596       currentdict /pdf.pdfmark.dict undef
3597     }
3598     if
3599     pdf.pdfmark.good
3600     { pdfmark }
3601     { cleartomark }
3602     ifelse
3603   }
3604   def
3605 /pdf.pdfmark.store
3606 {
3607   /pdf.pdfmark.dict 65534 dict def
3608   counttomark 1 add copy
3609   pop
3610   {
3611     dup mark eq
3612     {
3613       pop
3614       exit
3615     }
3616     {
3617       pdf.pdfmark.dict
3618       begin def end
3619     }
3620     ifelse
3621   }
3622   loop

```

```
3623 }  
3624 def  
  
(End definition for pdf.pdfmark and others. These functions are documented on page ??.)  
3625 </dvips & header>
```

Index

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

<code>_</code>	154	<code>_box_backend_rotate:Nn</code>	235, 283, 340, <u>419</u>
A			
<code>\AtBeginDvi</code>	57	<code>_box_backend_rotate_aux:Nn</code> ...	235, 283, <u>340</u>
B			
bool commands:			
<code>\bool_gset_false:N</code>	1278, 1297, 1320, 1342, 1358, 1459, 1698, 1734, 2302, 2348	<code>_box_backend_scale:Nnn</code>	252, <u>311</u> , <u>355</u> , <u>432</u>
<code>\bool_gset_true:N</code>	1276, 1345, 1457, 1713, 2295, 2301	<code>\l_box_backend_sin_fp</code>	<u>283</u>
<code>\bool_if:NTF</code>	67, 702, 1288, 1292, 1308, 1311, 1315, 1326, 1333, 1337, 1349, 1353, 1470, 1475, 1480, 1672, 1717, 1830, 1865, 1975, 2017, 2290, 2305, 2310, 2315	C	
<code>\bool_if:nTF</code>	2524, 2790, 3044	char commands:	
<code>\bool_lazy_and:nnTF</code>	918, 992, 3134, 3161	<code>\char_set_catcode_space:n</code>	154
<code>\bool_lazy_or:nnTF</code>	1857, 2010	clist commands:	
<code>\bool_new:N</code>	1279, 1346, 1460, 1714, 2275, 2276	<code>\clist_map_function:nN</code>	1366, 1490, 1741
<code>\bool_set_false:N</code>	1840, 1942, 2035, 2099	color internal commands:	
box commands:		<code>_color_backend:nnn</code>	<u>1164</u>
<code>\box_dp:N</code>	224, 226, 274, 276, 331, 333, 380, 382, 384, 386, 2327, 2360, 2361, 2386	<code>_color_backend_cmyk:w</code>	<u>1165</u>
<code>\box_ht:N</code>	226, 276, 333, 384, 386, 1877, 2072, 2332, 2371, 2372, 2388	<code>\g_color_backend_colorant_prop</code> ..	668, 687, 690, 710, 935
<code>\box_if_empty:NTF</code>	2421	<code>_color_backend_devicen_-colorants:n</code>	669, 871, 990
<code>\box_move_down:nn</code>	2249, 2327	<code>_color_backend_devicen_-colorants:w</code>	669
<code>\box_move_up:nn</code>	2251, 2332	<code>_color_backend_devicen_-init:nnn</code>	858, <u>960</u>
<code>\box_new:N</code>	2134, 2239, 2240	<code>_color_backend_devicen_init:w</code> ..	<u>960</u>
<code>\box_set_dp:Nn</code>	1797	<code>_color_backend_fill:n</code>	1071, 1098, <u>1128</u> , <u>1146</u> , <u>1153</u>
<code>\box_set_ht:Nn</code>	1796	<code>_color_backend_fill_cmyk:n</code> ...	1071, 1105, <u>1128</u> , <u>1153</u>
<code>\box_set_wd:Nn</code>	288, 1795	<code>_color_backend_fill_devicen:nn</code> ..	1097, 1119, <u>1145</u> , <u>1215</u>
<code>\box_use:N</code>	231, 249, 263, 279, 306, 320, 336, 352, 364, 415, 429, 448, 1410, 1605, 1798, 2280	<code>_color_backend_fill_gray:n</code> ...	1071, 1105, <u>1128</u> , <u>1153</u>
<code>\box_wd:N</code>	225, 233, 275, 281, 332, 338, 381, 383, 1876, 2071	<code>_color_backend_fill_rgb:n</code> ...	1071, 1105, <u>1128</u> , <u>1153</u>
box internal commands:		<code>_color_backend_fill_separation:nn</code> ..	1097, 1105, <u>1145</u> , <u>1215</u>
<code>_box_backend_clip:N</code>	213, 268, <u>325</u> , <u>369</u>	<code>\l_color_backend_fill_tl</code>	639, 649, 1079, 1094
<code>\l_box_backend_cos_fp</code>	<u>283</u>	<code>_color_backend_iccbase_-device:nnn</code>	<u>1037</u>
		<code>_color_backend_iccbase_-init:nnn</code>	877, <u>1011</u>
		<code>\c_color_backend_main_stack_int</code> ..	<u>516</u>
		<code>_color_backend_pickup:N</code> ..	<u>456</u> , <u>479</u>
		<code>_color_backend_pickup:w</code> ..	<u>14</u> , <u>456</u> , <u>479</u>

_color_backend_reset:	622 , 641 , 657 , 1082 , 1095 , 1105 , 1137 , 1162	_color_backend_stroke_cmyk:n	1071 , 1128 , 1164
_color_backend_rgb:w	1188	_color_backend_stroke_cmyk:w	1164
_color_backend_select:n	622 , 697	_color_backend_stroke_devicen:nn	1097 , 1123 , 1145 , 1215
_color_backend_select:nn	641 , 903	_color_backend_stroke_gray:n	1071 , 1128 , 1164
_color_backend_select_cmyk:n	622 , 641 , 657	_color_backend_stroke_gray-aux:n	1164
_color_backend_select_devicen:nn	696 , 880 , 902 , 1063	_color_backend_stroke_rgb:n	1071 , 1128 , 1164
_color_backend_select_gray:n	622 , 641 , 657	_color_backend_stroke_rgb:w	1164
_color_backend_select_iccbased:nn	699 , 884 , 902	_color_backend_stroke_separation:nn	1097 , 1105 , 1145 , 1215
_color_backend_select_rgb:n	622 , 641 , 657	\l_color_backend_stroke_tl	639 , 650 , 1081 , 1092
_color_backend_select_separation:nn	696 , 880 , 902 , 1063	\g_color_model_int	707 , 716 , 864 , 892 , 926 , 1000 , 1032
_color_backend_separation-init:n	700	\c_color_model_range_CIELAB_tl	819 , 854 , 949 , 956
_color_backend_separation-init:nn	906	color.sc	622 , 3244
_color_backend_separation-init:nnn	700	cs commands:	
_color_backend_separation-init:nnnn	700	\cs_generate_variant:Nn	49 , 63 , 66 , 99 , 138 , 143 , 170 , 201 , 207 , 572 , 609 , 721 , 1225 , 1420 , 1614 , 1989 , 2046 , 2062 , 2138 , 2175 , 2234 , 2726 , 2754 , 2864 , 2886 , 2921 , 3131 , 3199
_color_backend_separation-init:nnnnn	700 , 882 , 906	\cs_gset:Npx	2536 , 2540 , 3049 , 3054
_color_backend_separation-init:nw	700	\cs_gset_eq:NN	661 , 662 , 1066 , 1112 , 1113 , 1119 , 1121 , 1123
_color_backend_separation-init:w	700	\cs_gset_protected:Npn	551 , 659 , 663 , 1065 , 1107 , 1114 , 1116 , 1118 , 3165 , 3204 , 3213
_color_backend_separation-init_/DeviceCMYK:nnn	700	\cs_if_exist:NTF	27 , 50 , 457 , 480 , 539 , 1027 , 1050 , 2417 , 2815 , 2841
_color_backend_separation-init_/DeviceGray:nnn	700	\cs_if_exist_p:N	919 , 993 , 3135 , 3162
_color_backend_separation-init_/DeviceRGB:nnn	700	\cs_if_exist_use:NTF	38 , 734
_color_backend_separation-init_aux:nnnnnn	700	\cs_new:Npn	684 , 743 , 745 , 747 , 749 , 756 , 762 , 764 , 770 , 787 , 794 , 796 , 1005 , 1371 , 1495 , 1745 , 2075 , 2084 , 2128 , 2153 , 2235 , 2237 , 2270 , 2442 , 2542 , 2543 , 2696 , 2727 , 2728 , 2846 , 2879 , 2922 , 2924 , 2940 , 3057 , 3058 , 3068 , 3073 , 3074 , 3079 , 3080
_color_backend_separation-init_CIELAB:nnn	700 , 882 , 906	\cs_new:Npx	669 , 2563 , 2598 , 2755 , 2766 , 2833 , 2970
_color_backend_separation-init_CIELAB:nnnnnn	883	\cs_new_eq:NN	46 , 57 , 59 , 698 , 881 , 904 , 905 , 1101 , 1102 , 1149 , 1150 , 1217 , 1218 , 1224 , 1419 , 1425 , 1426 , 1613 , 1615 , 1616 , 1622 , 1807 , 1836 , 1887 , 1888 , 1930 , 1938 , 1960 , 2031 , 2088 , 2095 , 2127 , 2280
_color_backend_separation-init_count:n	700	\cs_new_protected:Npn	47 , 54 , 61 , 64 , 72 , 78 , 83 , 85 , 89 , 100 ,
_color_backend_separation-init_count:w	700		
_color_backend_separation-init_Device:Nn	700		
\g_color_backend_stack_int	516		
\l_color_backend_stack_int	513 , 541 , 547 , 651 , 654 , 1080 , 1093		
_color_backend_stroke:n	1071 , 1100 , 1105		


```

\__draw_backend_cm_decompose_-
  auxiii:nnnnN ..... 1551
\__draw_backend_curveto:nnnnnn ..
  ..... 1240, 1427, 1623
\__draw_backend_dash:n .....
  ..... 1360, 1484, 1736
\__draw_backend_dash_aux:nn .. 1736
\__draw_backend_dash_pattern:nn .
  ..... 1360, 1484, 1736
\__draw_backend_discardpath: ...
  ..... 1280, 1461, 1668
\__draw_backend_end: 1226, 1421, 1617
\__draw_backend_evenodd_rule: ...
  ..... 1275, 1456, 1664
\__draw_backend_fill: 1280, 1461, 1668
\__draw_backend_fillstroke: ....
  ..... 1280, 1461, 1668
\__draw_backend_join_bevel: ....
  ..... 1360, 1484, 1736
\__draw_backend_join_miter: ....
  ..... 1360, 1484, 1736
\__draw_backend_join_round: ....
  ..... 1360, 1484, 1736
\__draw_backend_lineto:nn .....
  ..... 1240, 1427, 1623
\__draw_backend_linewidth:n ....
  ..... 1360, 1484, 1736
\__draw_backend_literal:n .....
  ..... 1224, 1229, 1233, 1237,
1239, 1242, 1250, 1258, 1267, 1281,
1284, 1285, 1286, 1287, 1290, 1296,
1306, 1313, 1319, 1324, 1329, 1330,
1331, 1332, 1335, 1341, 1351, 1357,
1362, 1375, 1379, 1381, 1383, 1385,
1387, 1389, 1391, 1394, 1399, 1400,
1401, 1402, 1403, 1404, 1408, 1409,
1411, 1412, 1413, 1414, 1415, 1419,
1429, 1434, 1439, 1449, 1462, 1464,
1466, 1469, 1474, 1479, 1483, 1486,
1499, 1503, 1505, 1507, 1509, 1511,
1513, 1515, 1613, 1675, 1694, 1720
\__draw_backend_miterlimit:n ...
  ..... 1360, 1484, 1736
\__draw_backend_moveto:nn .....
  ..... 1240, 1427, 1623
\__draw_backend_nonzero_rule: ...
  ..... 1275, 1456, 1664
\__draw_backend_path:n ..... 1668
\g__draw_backend_path_int 1683, 1700
\g__draw_backend_path_tl .....
  ..... 1623, 1679, 1695, 1697, 1724
\__draw_backend_rectangle:nnnn ..
  ..... 1240, 1427, 1623
\__draw_backend_scope_begin: ...
  ..... 1236, 1422, 1425, 1615
\__draw_backend_scope_end: .....
  ..... 1236, 1424, 1425, 1615
\__draw_backend_stroke: .....
  ..... 1280, 1461, 1668
\g__draw_draw_clip_bool .. 1280, 1668
\g__draw_draw_eor_bool .....
  ... 1275, 1292, 1308, 1315, 1326,
1337, 1353, 1456, 1470, 1475, 1480
\g__draw_draw_path_int ..... 1668
\g__draw_path_tl ..... 1733

E
\errmsgmessage ..... 38
\evensidemargin ..... 2325
exp commands:
\exp_after:wN ..... 159, 465, 2082
\exp_args:Ne ..... 758, 2479, 3004
\exp_args:Nf ..... 1365, 1489, 2244
\exp_args:NNf ..... 236, 284, 341
\exp_args:Nnx ..... 2231, 2917
\exp_args:NV ..... 461
\exp_args:Nx . 704, 916, 1900, 1921,
2188, 2203, 2321, 2883, 3090, 3147
\exp_last_unbraced:Nx ..... 470, 484
\exp_not:N ..... 522, 523, 531,
533, 671, 677, 678, 679, 706, 707,
710, 711, 716, 2565, 2567, 2570,
2600, 2602, 2605, 2757, 2759, 2762,
2768, 2770, 2773, 2810, 2811, 2817,
2818, 2837, 2842, 2951, 2959, 2975
\exp_not:n .....
  .... 48, 97, 108, 136, 1019, 2179,
2184, 2473, 2712, 2713, 2727, 2728,
2740, 2741, 2895, 2900, 2911, 2984
\ExplBackendFileDate ..... 1

F
file commands:
\file_compare_timestamp:nNnTF . 1909
\file_parse_full_name:nnnn 1896, 1919
\fmtversion ..... 52
fp commands:
\fp_compare:nNnTF .....
  . 243, 290, 296, 348, 1532, 1545, 1590
\fp_eval:n . 236, 245, 258, 259, 284,
301, 316, 318, 341, 350, 361, 362,
426, 441, 442, 1172, 1173, 1174,
1182, 1195, 1196, 1197, 1534, 1539,
1540, 1547, 1557, 1558, 1559, 1560,
1569, 1570, 1571, 1572, 1581, 1582,
1583, 1584, 2470, 2639, 2998, 3091,
3101, 3108, 3148, 3172, 3179, 3240

```

\fp_new:N	309, 310	__graphics_backend_getbb_-auxvi:nNnn	2066, 2068
\fp_set:Nn	289, 292	__graphics_backend_getbb_eps:n	1807, 1889, 1930, 2088
\fp_use:N	295, 299, 304	__graphics_backend_getbb_eps:nm	1889
\fp_zero:N	291	__graphics_backend_getbb_eps:nn	1900, 1907
\c_zero_fp	243, 290, 296, 348, 1532, 1545	__graphics_backend_getbb_jpg:n	1822, 1930, 2025, 2089
G			
graphics commands:			
\graphics_bb_restore:nTF	1851, 2065	__graphics_backend_getbb_-pagebox:w	2025, 2082
\graphics_bb_save:n	1880, 2073	__graphics_backend_getbb_pdf:n	1822, 1915, 1930, 2025, 2096
\l_graphics_decodearray_tl	1828, 1829, 1839, 1859, 1863, 1864, 1941, 1973, 1974, 2012, 2015, 2016, 2034, 2098	__graphics_backend_getbb_png:n	1822, 1930, 2025, 2089
\graphics_extract_bb:n	1936, 1943, 2093, 2100	__graphics_backend_include:nn	2102
\l_graphics_interpolate_bool	1830, 1840, 1858, 1865, 1942, 1975, 2011, 2017, 2035, 2099	__graphics_backend_include-auxi:nn	1947
\l_graphics_llx_dim	1813, 1952, 2004, 2111	__graphics_backend_include-auxii:nnn	1947
\l_graphics_lly_dim	1814, 1953, 2005, 2112	__graphics_backend_include-auxiii:nnn	1947
\l_graphics_name_tl	1914	__graphics_backend_include-bitmap_quote:w	2076, 2117
\l_graphics_page_int	1824, 1844, 1845, 1869, 1870, 1934, 1971, 1972, 1998, 1999, 2027, 2040, 2041, 2080, 2081, 2091	__graphics_backend_include-eps:n	1808, 1889, 1947, 2102
\l_graphics_pagebox_tl	54, 1825, 1843, 1871, 1872, 1935, 1969, 1970, 2000, 2002, 2028, 2049, 2050, 2082, 2092	__graphics_backend_include-jpg:n	1882, 1947, 2117
\graphics_read_bb:n	1807, 1930, 2088	__graphics_backend_include-pdf:n	1882, 1921, 1947, 2076, 2102
\l_graphics_urx_dim	1815, 1876, 1954, 2006, 2071, 2113	__graphics_backend_include-pdf_quote:w	2079, 2084
\l_graphics_ury_dim	1816, 1877, 1955, 2007, 2072, 2114, 2122, 2123	__graphics_backend_include-png:n	1882, 1947, 2117
graphics internal commands:			
\l__graphics_backend_dir_str	1889	\l_graphics_backend_name_str	1889
\l__graphics_backend_ext_str	1889	\l_graphics_graphics_attr_tl	1821, 1826, 1833, 1841, 1851, 1878, 1880, 1885
__graphics_backend_getbb_auxi:n	1822	\l__graphics_internal_box	1874, 1876, 1877, 2070, 2071, 2072
__graphics_backend_getbb-auxi:nN	2025	\g_graphics_track_int	1946, 1992, 1993
__graphics_backend_getbb-auxii:n	1822	group commands:	
__graphics_backend_getbb-auxii:nnN	2025	\group_begin:	151, 179, 198
__graphics_backend_getbb-auxiii:nNnn	2025	\group_end:	164, 187
__graphics_backend_getbb-auxiv:nnNnn	2025	\group_insert_after:N	1082, 1095, 1110, 1137, 1162, 3159, 3196
__graphics_backend_getbb-auxiv:nnNnn	2025	H	
hbox commands:			
\hbox:n	2250, 2253, 2328, 2334, 2487, 2494, 3012, 3023		

`\hbox_overlap_right:n` 231,
 263, 279, 320, 336, 364, 448, 1410, 1605
`\hbox_set:Nn` .. 1874, 2070, 2320, 2352
`\hbox_set:Nw` 2303
`\hbox_set_end:` 2318
`\hbox_unpack:N` 2439
 hook commands:
`\hook_gput_code:nnn` 55

I

int commands:
`\int_compare:nNnTF` 516,
 561, 657, 1063, 1105, 1844, 1869,
 1971, 1998, 2040, 2080, 2411, 2512,
 2808, 2836, 2949, 2956, 2972, 3202
`\int_const:Nn` 157, 163, 523,
 549, 587, 1878, 1993, 2148, 2686, 2874
`\int_eval:n`
 . 568, 578, 607, 618, 754, 763, 776,
 778, 782, 795, 2536, 2540, 2786,
 2811, 2818, 2831, 3041, 3049, 3054
`\int_gincr:N` 205, 371,
 522, 1674, 1719, 1992, 2147, 2216,
 2260, 2337, 2873, 2916, 2929, 2951
`\int_gset:Nn` 180, 199, 2400
`\int_gset_eq:NN` 188, 2261, 2338, 2930
`\int_if_exist:NTF` 1982
`\int_if_odd:nTF` 2323
`\int_new:N` 171, 172,
 418, 513, 519, 1700, 1946, 2143,
 2241, 2272, 2274, 2869, 2926, 2942
`\int_set:Nn` 541
`\int_set_eq:NN` ... 176, 195, 547, 2412
`\int_step_function:nnnN` 780
`\int_use:N` . 373, 404, 531, 542, 557,
 707, 716, 864, 892, 926, 1000, 1032,
 1677, 1683, 1690, 1722, 1730, 1845,
 1870, 1885, 1972, 1985, 1997, 1999,
 2081, 2154, 2219, 2232, 2236, 2264,
 2271, 2342, 2443, 2697, 2707, 2880,
 2918, 2923, 2933, 2941, 2959, 2975
`\int_value:w`
 2565, 2600, 2757, 2768, 2786
`\int_zero:N` ... 1824, 1934, 2027, 2091

K

kernel internal commands:
`__kernel_backend_align_begin:` ..
 72, 216, 240, 255
`__kernel_backend_align_end:` ...
 72, 230, 248, 262
`__kernel_backend_first_shipout:n`
 50, 69, 526, 704

`\g__kernel_backend_header_bool` ..
 67, 702
`__kernel_backend_literal:n`
 46, 62, 65, 70,
 74, 81, 84, 86, 142, 145, 147, 149,
 169, 345, 358, 528, 553, 554, 565,
 575, 630, 636, 660, 664, 724, 860,
 1109, 1115, 1117, 1136, 1161, 1228,
 1234, 1529, 1536, 1542, 1602, 1607,
 1810, 1949, 1984, 1994, 2108, 2119,
 2863, 2979, 3041, 3045, 3050, 3055
`__kernel_backend_literal_page:n`
 100, 144, 2857, 2859, 3060, 3062
`__kernel_backend_literal_pdf:n` .
 .. 89, 141, 271, 328, 1419, 3211, 3226
`__kernel_backend_literal_-
 postscript:n`
 61, 75, 76, 80, 217, 218, 220,
 221, 229, 241, 256, 1224, 2514, 2526
`__kernel_backend_literal_svg:n` .
 168, 175, 186, 194, 204,
 372, 374, 391, 886, 1613, 1788, 1799
`__kernel_backend_matrix:n`
 128, 293, 314, 1519
`__kernel_backend_postscript:n` ..
 64,
 632, 1140, 1142, 1144, 1148, 2137,
 2193, 2208, 2250, 2256, 2296, 2328,
 2335, 2339, 2353, 2381, 2425, 2432,
 2438, 2446, 2453, 2487, 2494, 3114
`__kernel_backend_scope:n` 173, 401,
 406, 1202, 1620, 1665, 1667, 1687,
 1727, 1749, 1761, 1763, 1765, 1767,
 1769, 1771, 1773, 1775, 1778, 3240
`__kernel_backend_scope_begin:` ..
 83, 110, 146, 173,
 215, 239, 254, 270, 287, 313, 327,
 344, 357, 1425, 1597, 1615, 1619, 1786
`__kernel_backend_scope_begin:n` .
 173, 393, 421, 434
`__kernel_backend_scope_end:` 83,
 110, 146, 173, 232, 250, 264, 280,
 307, 321, 337, 353, 365, 416, 430,
 449, 551, 1426, 1609, 1616, 1622, 1800
`\g__kernel_backend_scope_int` ...
 171, 178, 180, 185, 189, 197, 199, 205
`\l__kernel_backend_scope_int` ...
 171, 177, 190, 196
`\g__kernel_clip_path_int`
 369, 1674, 1677, 1690, 1719, 1722, 1730
`__kernel_color_backend_stack_-
 init:Nnn` 516, 585, 3138
`__kernel_color_backend_stack_-
 pop:n` 561, 599, 654, 3168

_kernel_color_backend_stack_- push:nn 561 , 599 , 651 , 1080 , 1093 , 3157 , 3194	\pdf_object_write:nn . 943 , 1016 , 1042
_kernel_dependency_version_- check:Nn 1	pdf internal commands:
_kernel_dependency_version_- check:nn 27 , 29	_pdf_backend:n 2862 , 2866 , 2868 , 2894 , 2899 , 2908 , 2931 , 2953 , 2969 , 2982 , 3014 , 3015 , 3025
_kernel_kern:n 2255 , 2257 , 2486 , 2490 , 2493 , 2497 , 3011 , 3019 , 3022 , 3038	_pdf_backend_annotation:nnnn 2242 , 2550 , 2927
\c_kernel_sys_dvipdfmx_version_- int 151 , 516 , 561 , 657 , 1063 , 1105 , 2949 , 2956 , 2972 , 3202	_pdf_backend_annotation_int 2241 , 2261 , 2271 , 2926 , 2930 , 2941
	_pdf_backend_annotation_last: 2270 , 2563 , 2940
	_pdf_backend_bdc:nn 2544 , 2856 , 3059 , 3081
	_pdf_backend_catalog_gput:nn 2139 , 2656 , 2865 , 3065
	_pdf_backend_compress_objects:n 2510 , 2777 , 3040 , 3075
	_pdf_backend_compresslevel:n 2510 , 2777 , 3040 , 3075
	\l_pdf_backend_content_box 2239 , 2303 , 2327 , 2330 , 2332 , 2361 , 2372
	_pdf_backend_destination:nn 2451 , 2619 , 2980
	_pdf_backend_destination:nnnn 2451 , 2619 , 2980
	_pdf_backend_destination_- aux:nnnn 2451 , 2980
	_pdf_backend_emc: 2544 , 2856 , 3059 , 3081
	_pdf_backend_info_gput:nn 2139 , 2656 , 2865 , 3065
	_pdf_backend_link:nw 2281
	_pdf_backend_link_aux:nw ... 2281
	_pdf_backend_link_begin:n .. 2943
	_pdf_backend_link_begin:nnnw 2574
	_pdf_backend_link_begin:nw 2283 , 2287 , 2288
	_pdf_backend_link_begin_aux:nw 2291 , 2293
	_pdf_backend_link_begin_- goto:nnw 2281 , 2574 , 2943
	_pdf_backend_link_begin_- user:nnw 2281 , 2574 , 2943
	\g_pdf_backend_link_bool 2276 , 2290 , 2295 , 2310 , 2348
	\g_pdf_backend_link_dict_tl 2273 , 2298 , 2343
	_pdf_backend_link_end: 2281 , 2574 , 2943
	_pdf_backend_link_end_aux: . 2281
M	
\MessageBreak 40	
mode commands:	
\mode_if_horizontal:TF ... 2402 , 2409	
\mode_if_math:TF 2300	
O	
\oddsidemargin 2324	
opacity internal commands:	
_opacity_backend:nn 3233	
_opacity_backend:nnn 3088	
_opacity_backend_fill:n 3088 , 3169 , 3233	
_opacity_backend_fill_stroke:nn 3171 , 3177 , 3181 , 3199 , 3213	
\l_opacity_backend_fill_tl 3143 , 3152 , 3178 , 3186 , 3206 , 3218	
_opacity_backend_fillstroke:nn 3169	
_opacity_backend_reset: 3145 , 3196	
_opacity_backend_select:n 3088 , 3145 , 3202 , 3233	
_opacity_backend_select_aux:n 3088 , 3145 , 3184 , 3204 , 3216	
\c_opacity_backend_stack_int 3134 , 3157 , 3168 , 3194	
_opacity_backend_stroke:n 3088 , 3169 , 3233	
\l_opacity_backend_stroke_tl 3143 , 3153 , 3173 , 3187 , 3207 , 3219	
P	
pdf commands:	
\pdf_object_if_exist:nTF 940 , 1013 , 1039	
\pdf_object_new:nn ... 942 , 1015 , 1041	
\pdf_object_ref:n 955 , 1026 , 1049	
\pdf_object_ref_last: 927 , 934 , 936 , 989 , 1001 , 1033 , 1057	
\pdf_object_unnamed_write:nn 908 , 933 , 962 , 984 , 1025 , 1048	

\g_pdf_backend_link_int	__pdf_backend_pdfmark:n
..... 2272, 2338,	2136, 2140, 2142, 2157, 2178, 2183,
2342, 2443, 2942, 2951, 2959, 2975	2217, 2262, 2454, 2498, 2545, 2547
__pdf_backend_link_last:	__pdf_backend_version_major: ...
..... 2442, 2598, 2970 2536,
__pdf_backend_link_margin:n ...	2542, 2833, 3049, 3050, 3057, 3079
..... 2444, 2609, 2978	__pdf_backend_version_major_-
\g_pdf_backend_link_math_bool ..	gset:n 2534, 2805, 3047, 3077
..... 2275, 2301, 2302, 2305, 2315	__pdf_backend_version_minor: ...
__pdf_backend_link_minima: .. 2281 2540,
__pdf_backend_link_outerbox:n 2281	2542, 2833, 3054, 3055, 3057, 3079
\g_pdf_backend_link_sf_int	__pdf_backend_version_minor_-
..... 2274, 2400, 2411, 2412	gset:n 2534, 2805, 3047, 3077
__pdf_backend_link_sf_restore: 2281	\l_pdf_breaklink_pdfmark_tl ...
__pdf_backend_link_sf_save: . 2281 2277, 2345, 2437
\l_pdf_backend_model_box . 2240,	__pdf_breaklink_postscript:n ...
2320, 2352, 2360, 2371, 2386, 2388 2279, 2329, 2331, 2438
__pdf_backend_objcompresslevel:n	__pdf_breaklink_usebox:N
..... 2777 2280, 2330, 2439
\g_pdf_backend_object_int	__pdf_exp_not_i:nn . 2698, 2744, 2749
..... 2143, 2147, 2150,	__pdf_exp_not_ii:nn 2698, 2745, 2750
2216, 2219, 2232, 2236, 2260, 2261,	\l_pdf_internal_box
2264, 2337, 2338, 2869, 2873, 2876,	pdf.baselineskip
2916, 2918, 2923, 2929, 2930, 2933	pdf.baselineskip
__pdf_backend_object_last:	pdf.bordertracking
..... 2235, 2755, 2922, 3067	pdf.bordertracking.begin
__pdf_backend_object_new:nn ...	pdf.bordertracking.continue
..... 2145, 2677, 2871, 3067	pdf.bordertracking.end
__pdf_backend_object_now:nn ...	pdf.bordertracking.endpage
..... 2214, 2729, 2914, 3067	pdf.bordertracking
\g_pdf_backend_object_prop	pdf.breaklink
..... 2143, 2151, 2162, 2172,	pdf.breaklink.write
2676, 2694, 2710, 2869, 2877, 2884	pdf.brokenlink.dict
__pdf_backend_object_ref:n 2145,	pdf.brokenlink.rect
2159, 2173, 2677, 2871, 2890, 3067	pdf.brokenlink.skip
__pdf_backend_object_write:nn ..	pdf.count
..... 2155, 2698, 2881, 3067	pdf.currentrect
__pdf_backend_object_write:nnn 2881	pdf.cvs
__pdf_backend_object_write_-	pdf.dest.anchor
array:nn 2155, 2881	pdf.dest.point
__pdf_backend_object_write_-	pdf.dest.x
dict:nn 2155, 2881	pdf.dest.y
__pdf_backend_object_write_-	pdf.dest2device
fstream:nn 2155, 2881	pdf.dev.x
__pdf_backend_object_write_-	pdf.dev.y
fstream:nnn 2189, 2191	pdf.dvi.pt
__pdf_backend_object_write_-	pdf.globaldict
stream:nn 2155, 2881	pdf.leftboundary
__pdf_backend_object_write_-	pdf.link.dict
stream:nnn 2155	pdf.linkdp.pad
__pdf_backend_object_write_-	pdf.linkht.pad
stream:nnnn 2881	pdf.linkmargin
__pdf_backend_pageobject_ref:n .	pdf.llx
..... 2237, 2766, 2924, 3067	pdf.lly 2281, 3258
	pdf.lly 2281, 3258
	pdf.originx
	pdf.originy

