

File I

Implementation

1 l3backend-basics implementation

```
1 <*package>
```

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

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

```
2 \ProvidesExplFile
3 <*dvipdfmx>
4   {l3backend-dvipdfmx.def}{2025-06-09}{}
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2025-06-09}{}
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2025-06-09}{}
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2025-06-09}{}
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2025-06-09}{}
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2025-06-09}{}
25   {L3 backend support: XeTeX}
26 </xetex>
```

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

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

```

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

```

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

- Color support is either dvips-like or LuaTeX/pdfTeX-like.
- LuaTeX/pdfTeX and dvipdfmx/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 behavior so a wrapper is provided.

```

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

```

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

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

```

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

```

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

1.1 dvips backend

```

59 <*dvips>

```

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

```

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

```

(End of definition for `_kernel_backend_literal_postscript:n`.)

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

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

(End of definition for `_kernel_backend_postscript:n`.)

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

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

`_kernel_backend_align_begin:` 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.

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

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

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

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

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

```
86 </dvips>
```

1.2 LuaTeX and pdfTeX backends

```
87 <*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:e
```

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

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

(End of definition for `_kernel_backend_literal_pdf:n`.)

```
\_kernel_backend_literal_page:n
\_kernel_backend_literal_page:e
```

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

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

(End of definition for `_kernel_backend_literal_page:n`.)

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

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

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

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

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

(End of definition for `_kernel_backend_matrix:n`.)

```
166 </luatex | pdftex>
```

1.3 dvipdfmx backend

```
167 <*dvipdfmx | xetex>
```

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

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

(End of definition for `_kernel_backend_literal_pdf:n`.)

```
\_kernel_backend_literal_page:n
```

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

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

(End of definition for `_kernel_backend_literal_page:n`.)

```
\_kernel_backend_scope_begin:
```

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

```
173 \cs_new_protected:Npn \_kernel_backend_scope_begin:  
174 { \_kernel_backend_literal:n { x:gsave } }  
175 \cs_new_protected:Npn \_kernel_backend_scope_end:  
176 { \_kernel_backend_literal:n { x:grestore } }
```

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

```
177 </dvipdfmx | xetex>
```

1.4 dvisvgm backend

```
178 <*dvisvgm>
```

```
\_kernel_backend_literal_svg:n  
\_kernel_backend_literal_svg:e
```

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

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

(End of definition for `_kernel_backend_literal_svg:n`.)

```
\g__kernel_backend_scope_int  
\l__kernel_backend_scope_int
```

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

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

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

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

```

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

```

(End of definition for `_kernel_backend_scope_begin:` and others.)

```

219 </dvisvgm>
220 </package>

```

2 l3backend-box implementation

```

221 *package)
222 @@=box)

```

2.1 dvips backend

```

223 *dvips)

```

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

```

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

```

(End of definition for `_box_backend_clip:N`.)

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

```

246 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
247 { \exp_args:Nnf \_box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
248 \cs_new_protected:Npn \_box_backend_rotate_aux:Nn #1#2
249 {
250   \_kernel_backend_scope_begin:
251   \_kernel_backend_align_begin:
252   \_kernel_backend_literal_postscript:e
253   {
254     \fp_compare:nNnTF {#2} = \c_zero_fp
255     { 0 }
256     { \fp_eval:n { round ( -(#2) , 5 ) } } } ~
257     rotate
258   }
259   \_kernel_backend_align_end:
260   \box_use:N #1
261   \_kernel_backend_scope_end:
262 }

```

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

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

```

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

```

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

```
277 </dvips>
```

2.2 LuaTeX and pdfTeX backends

```
278 <*luatex | pdftex>
```

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

```

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

```

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

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

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

```

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

```

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

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

```

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

```

(End of definition for __box_backend_scale:Nnn.)

334 </luatex | pdftex>

2.3 dvipdfmx/X_YTeX backend

335 <*dvipdfmx | xetex>

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

```

336 \cs_new_protected:Npn \__box_backend_clip:N #1

```

```

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

```

(End of definition for __box_backend_clip:N.)

__box_backend_rotate:Nn
 __box_backend_rotate_aux:Nn

Rotating in 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.

```

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

```

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

__box_backend_scale:Nnn

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

```

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

```

(End of definition for `_box_backend_scale:Nnn`.)

```
378 </dviptfm | xetex>
```

2.4 dvisvgm backend

```
379 <*dvisvgm>
```

```
\_box_backend_clip:N  
\_kernel_clip_path_int
```

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

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

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

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

```

(End of definition for __box_backend_scale:Nnn.)

```
462 </dvisvgm>
```

```
463 </package>
```

3 I3backend-color implementation

```
464 <*package>
```

```
465 <@@=color>
```

Color support is split into parts: collecting data from L^AT_εX, 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_qT_EX in particular. Whilst it is in some ways convenient to use the same approach in multiple backends, the fact that dvipdfmx/X_qT_EX is PDF-based means it (largely) sticks closer to direct PDF output.

3.1 The color stack

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

3.1.1 Common code

```
466 <*luatex | pdftex>
```

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

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

(End of definition for \l__color_backend_stack_int.)

```
468 </luatex | pdftex>
```

3.1.2 LuaT_EX and pdfT_EX

```
469 <*luatex | pdftex>
```

__kernel_color_backend_stack_init:Nnn

```
470 \cs_new_protected:Npn \__kernel_color_backend_stack_init:Nnn #1#2#3
```

```
471 {
```

```
472     \int_const:Nn #1
```

```
473     {
```

```
474     <*luatex>
```

```
475     \tex_pdffeedback:D colorstackinit ~
```

```
476 </luatex>
```

```

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

```

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

```

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

```

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

```
504 </luatex | pdftex>
```

3.2 General color

3.2.1 dvips-style

```
505 <*dvips | dvisvgm>
```

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

```

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

```

```

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

```

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

```
523 </dvips | dvisvgm>
```

3.2.2 LuaTeX and pdfTeX

```
524 <*luatex | pdftex>
```

```

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

```

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

```

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

```

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

```
543 </luatex | pdftex>
```

3.2.3 dvipdfmx/X_YTeX

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

```
544 <*dvipdfmx | xetex>
```



```

\__color_backend_select:n Using the single stack is relatively easy as there is only one route.
  \_color_backend_select_cmyk:n 545 \cs_new_protected:Npn \__color_backend_select:n #1
  \_color_backend_select_gray:n 546 { \__kernel_backend_literal:n { pdf : bc ~ [ #1 ] } }
  \_color_backend_select_rgb:n 547 \cs_new_eq:NN \__color_backend_select_cmyk:n \__color_backend_select:n
\__color_backend_reset: 548 \cs_new_eq:NN \__color_backend_select_gray:n \__color_backend_select:n
  549 \cs_new_eq:NN \__color_backend_select_rgb:n \__color_backend_select:n
  550 \cs_new_protected:Npn \__color_backend_reset:
  551 { \__kernel_backend_literal:n { pdf : ec } }

```

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

```

\_color_backend_select_named:n For classical named colors, the only value we should get is Black.
  552 \cs_new_protected:Npn \__color_backend_select_named:n #1
  553 {
  554   \str_if_eq:nnTF {#1} { Black }
  555   { \__color_backend_select_gray:n { 0 } }
  556   { \msg_error:nnn { color } { unknown-named-color } {#1} }
  557 }
  558 \msg_new:nnn { color } { unknown-named-color }
  559 { Named-color-’#1’-is-not-known. }

```

(End of definition for __color_backend_select_named:n.)

```
560 </dviptfm | xetex>
```

3.3 Separations

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

```
561 <*dviptfm | luatex | pdftex | xetex | dvips>
```

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

```

\g__color_backend_colorant_prop
  562 \prop_new:N \g__color_backend_colorant_prop

```

(End of definition for \g__color_backend_colorant_prop.)

```

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

```

```

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

```

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

```
588 </dviPDFmx | luatex | pdftex | xetex | dvips>
```

```
589 <*dvips>
```

```

\__color_backend_select_separation:nn
\__color_backend_select_devicen:nn

```

```

590 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
591 { \__color_backend_select:n { separation ~ #1 ~ #2 } }
592 \cs_new_eq:MN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn

```

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

```
\__color_backend_select_iccbased:nn
```

No support.

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

(End of definition for __color_backend_select_iccbased:nn.)

```

\__color_backend_separation_init:nmnnn
\__color_backend_separation_init:neenn
\__color_backend_separation_init_aux:nmnnnn
\__color_backend_separation_init_/DeviceCMYK:nnm
\__color_backend_separation_init_/DeviceGray:nnm
\__color_backend_separation_init_/DeviceRGB:nnm
\__color_backend_separation_init_Device:Nn
\__color_backend_separation_init:nnm
\__color_backend_separation_init_count:n
\__color_backend_separation_init_count:w
\__color_backend_separation_init:nmnn
\__color_backend_separation_init:w
\__color_backend_separation_init:n
\__color_backend_separation_init:nw
\__color_backend_separation_init_CIELAB:nnm

```

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

```

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

```

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

705 }

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

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

(End of definition for __color_backend_separation_init:nnnn and others.)

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

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

```

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

```

(End of definition for __color_backend_devicen_init:nnn.)

_color_backend_iccbased_init:nnm No support at present.

```

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

```

(End of definition for __color_backend_iccbased_init:nnm.)

```

772 </dvips>

```

```

773 <*dvisvgm>

```

_color_backend_select_separation:nn No support at present.

_color_backend_select_devicen:nn

```

774 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2 { }

```

```

775 \cs_new_eq:NN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn

```

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

_color_backend_separation_init:nnmm No support at present.

_color_backend_separation_init_CIELAB:nnm

```

776 \cs_new_protected:Npn \__color_backend_separation_init:nnmm #1#2#3#4#5 { }

```

```

777 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nnmm #1#2#3 { }

```

(End of definition for _color_backend_separation_init:nnmm and _color_backend_separation_init_CIELAB:nnm.)

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

```

778 \cs_new_protected:Npn \__color_backend_select_iccbased:nn #1#2

```

```

779 {
780   \__kernel_backend_literal_svg:e
781   {

```

```

782     <style>
783       @color-profile ~
784       \str_if_eq:nnTF {#2} { cmyk }
785       { device-cmyk }
786       { --color \int_use:N \g__color_model_int }
787       \c_space_tl
788     {

```

```

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

```

(End of definition for `_color_backend_select_iccbased:nn`.)

```

794 </dvisvgm>
795 <*dvipdfmx | luatex | pdftex | xetex>

```

```

\_color_backend_select_separation:nn
\_color_backend_select_devicen:nn
\_color_backend_select_iccbased:nn

```

```

796 <*dvipdfmx | xetex>
797 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
798 { \_kernel_backend_literal:e { pdf : bc ~ \pdf_object_ref:n {#1} ~ [ #2 ] } }
799 </dvipdfmx | xetex>
800 <*luatex | pdftex>
801 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
802 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
803 </luatex | pdftex>
804 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
805 \cs_new_eq:NN \_color_backend_select_iccbased:nn \_color_backend_select_separation:nn

```

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

```

\_color_backend_init_resource:n

```

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

```

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

```

(End of definition for `_color_backend_init_resource:n`.)

```

\_color_backend_separation_init:nnnn
\_color_backend_separation_init:nn
\_color_backend_separation_init_CIELAB:nnn

```

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

```

823 \cs_new_protected:Npn \_color_backend_separation_init:nnnn #1#2#3#4#5
824 {
825     \pdf_object_unnamed_write:ne { dict }

```

```

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

```

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

```

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

```

(End of definition for _color_backend_separation_init:nnnnn, _color_backend_separation_init:nn, and _color_backend_separation_init_CIELAB:nnn.)

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

```

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

```



```

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

```

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

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

```

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

```

```

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

```

(End of definition for __color_backend_iccbased_init:nnn.)

__color_backend_iccbased_device:nnn

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

```

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

```

(End of definition for __color_backend_iccbased_device:nnn.)

```

947 </dvipdfmx | luatex | pdftex | xetex>

```

3.4 Fill and stroke color

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

```

948 <*dvipdfmx | xetex>

```

```

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

```

(End of definition for `_color_backend_fill:n` and others.)

```

\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
959 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
960   {
961     \_kernel_backend_literal:e
962     { pdf : bc ~ fill ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
963   }
964 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
965   {
966     \_kernel_backend_literal:e
967     { pdf : bc ~ stroke ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
968   }
969 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
970 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

```

\_color_backend_fill_reset:
  \_color_backend_stroke_reset:
971 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
972 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:

```

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

```

973 </dviPDFx | xetex>
974 <*luatex | pdftex>

```

`_color_backend_fill_cmyk:n` `_color_backend_fill_gray:n` `_color_backend_fill_rgb:n` `_color_backend_fill:n` `_color_backend_stroke_cmyk:n` `_color_backend_stroke_gray:n` `_color_backend_stroke_rgb:n` `_color_backend_stroke:n`

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.

```

975 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
976   { \_color_backend_fill:n { #1 ~ k } }
977 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
978   { \_color_backend_fill:n { #1 ~ g } }
979 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
980   { \_color_backend_fill:n { #1 ~ rg } }
981 \cs_new_protected:Npn \_color_backend_fill:n #1
982   {
983     \tl_set:Nn \l__color_backend_fill_tl {#1}
984     \_kernel_color_backend_stack_push:nn \l__color_backend_stack_int
985     { #1 ~ \l__color_backend_stroke_tl }
986   }
987 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
988   { \_color_backend_stroke:n { #1 ~ K } }
989 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
990   { \_color_backend_stroke:n { #1 ~ G } }
991 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
992   { \_color_backend_stroke:n { #1 ~ RG } }
993 \cs_new_protected:Npn \_color_backend_stroke:n #1
994   {
995     \tl_set:Nn \l__color_backend_stroke_tl {#1}
996     \_kernel_color_backend_stack_push:nn \l__color_backend_stack_int
997     { \l__color_backend_fill_tl \c_space_tl #1 }
998   }

```

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

```
\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
1999 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1000 { \_color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
1001 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1002 { \_color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
1003 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1004 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn
```

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

```
\_color_backend_fill_reset:
  \_color_backend_stroke_reset:
1005 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
1006 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:
```

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

```
1007 </luatex | pdftex>
```

```
1008 <*dvips>
```

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

```
\_color_backend_fill_gray:n
\_color_backend_fill_rgb:n
  \_color_backend_fill:n
  \_color_backend_stroke_cmyk:n
  \_color_backend_stroke_gray:n
  \_color_backend_stroke_rgb:n
1009 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1010 { \_color_backend_fill:n { cmyk ~ #1 } }
1011 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1012 { \_color_backend_fill:n { gray ~ #1 } }
1013 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1014 { \_color_backend_fill:n { rgb ~ #1 } }
1015 \cs_new_protected:Npn \_color_backend_fill:n #1
1016 {
1017   \_kernel_backend_literal:n { color~push~ #1 }
1018 }
1019 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1020 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
1021 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1022 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
1023 \cs_new_protected:Npn \_color_backend_stroke_rgb:n #1
1024 { \_kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }
```

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

```
\_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn
  \_color_backend_fill_devicen:nn
  \_color_backend_stroke_devicen:nn
1025 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
1026 { \_color_backend_fill:n { separation ~ #1 ~ #2 } }
1027 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
1028 { \_kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1029 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
1030 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn
```

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

```
\_color_backend_fill_reset:
  \_color_backend_stroke_reset:
1031 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
1032 \cs_new_protected:Npn \_color_backend_stroke_reset: { }
```

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

1033 `</dvips>`

1034 `<*dvisvgm>`

`_color_backend_fill_cmyk:n` Fill color here is the same as general color.

```
1035 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
1036   { \_color_backend_fill:n { cmyk ~ #1 } }
\_color_backend_fill_gray:n
1037 \cs_new_protected:Npn \_color_backend_fill_gray:n #1
1038   { \_color_backend_fill:n { gray ~ #1 } }
\_color_backend_fill_rgb:n
1039 \cs_new_protected:Npn \_color_backend_fill_rgb:n #1
1040   { \_color_backend_fill:n { rgb ~ #1 } }
\_color_backend_fill:n
1041 \cs_new_protected:Npn \_color_backend_fill:n #1
1042   {
1043     \_kernel_backend_literal:n { color~push~ #1 }
1044   }
```

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

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

```
\_color_backend_stroke_gray:n
\_color_backend_stroke_gray_aux:n
1045 \cs_new_protected:Npn \_color_backend_stroke_cmyk:n #1
1046   {
1047     \_color_backend_fill_cmyk:n {#1}
1048     \_kernel_backend_scope:n { stroke = "{?color}" }
1049     \_color_backend_reset:
1050   }
\_color_backend_stroke_rgb:n
\_color_backend_stroke_rgb:w
\_color_backend:nnn
1051 \cs_new_protected:Npn \_color_backend_stroke_gray:n #1
1052   {
1053     \use:e
1054     {
1055       \_color_backend_stroke_gray_aux:n
1056       { \fp_eval:n { 100 * (#1) } }
1057     }
1058   }
1059 \cs_new_protected:Npn \_color_backend_stroke_gray_aux:n #1
1060   { \_color_backend:nnn {#1} {#1} {#1} }
1061 \cs_new_protected:Npn \_color_backend_stroke_new_rgb:n #1
1062   { \_color_backend_rgb:w #1 \s_color_stop }
1063 \cs_new_protected:Npn \_color_backend_stroke_new_rgb:w
1064   #1 ~ #2 ~ #3 \s_color_stop
1065   {
1066     \use:e
1067     {
1068       \_color_backend:nnn
1069       { \fp_eval:n { 100 * (#1) } }
1070       { \fp_eval:n { 100 * (#2) } }
1071       { \fp_eval:n { 100 * (#3) } }
1072     }
1073   }
1074 \cs_new_protected:Npe \_color_backend:nnn #1#2#3
1075   {
1076     \_kernel_backend_scope:n
```

```

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

```

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

At present, these are no-ops.

```

\_color_backend_fill_separation:nn 1089 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2 { }
\_color_backend_stroke_separation:nn 1090 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2 { }
\_color_backend_fill_devicen:nn 1091 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
\_color_backend_stroke_devicen:nn 1092 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

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

```

\_color_backend_fill_reset:
\_color_backend_stroke_reset: 1093 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
1094 \cs_new_protected:Npn \_color_backend_stroke_reset: { }

```

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

No support at present.

```

\_color_backend_devicen_init:nnn 1095 \cs_new_protected:Npn \_color_backend_devicen_init:nnn #1#2#3 { }
\_color_backend_iccbased_init:nnn 1096 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3 { }

```

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

```

1097 </divisvgn>
1098 </package>

```

3.5 Font handling integration

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

```

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

```

```

1111     token.create'tex_endlocalcontrol:D',
1112     token.create'tex_hpack:D',
1113     token.new(0, 1),
1114     token.create'color_export:nnN',
1115     token.new(0, 1),
1116     '',
1117     token.new(0, 2),
1118     token.new(0, 1),
1119     'backend',
1120     token.new(0, 2),
1121     token.create'l_tmpa_tl',
1122     token.create'exp_after:wN',
1123     token.create'__color_select:nn',
1124     token.create'l_tmpa_tl',
1125     token.new(0, 2),
1126 }
1127 local group_end = token.create'group_end:'
1128 local value = (1 - l.P'}')^0
1129 luatexbase.add_to_callback('luaotfload.parse_color', function (value)
1130 % Also allow HTML colors to preserve compatibility
1131     local html = htmlcolor:match(value)
1132     if html then return html end
1133
1134 % If no l3color named color with this name is known, check for defined xcolor colors
1135     local l3color_prop = token.get_macro(string.format('l__color_named_%s_prop', value))
1136     if l3color_prop == nil or l3color_prop == '' then
1137         local legacy_color_macro = token.create(string.format('\\color@%s', value))
1138         if legacy_color_macro.cmdname ~= 'undefined_cs' then
1139             token.put_next(legacy_color_macro)
1140             return token.scan_argument()
1141         end
1142     end
1143
1144     tex.runtoks(function()
1145         token.get_next()
1146         color_export[6] = value
1147         tex.sprint(-2, color_export)
1148     end)
1149     local list = token.scan_list()
1150     if not list.head or list.head.next
1151         or list.head.subtype ~= node.subtype'pdf_colorstack' then
1152         error'Unexpected backend behavior'
1153     end
1154     local cmd = list.head.data
1155     node.free(list)
1156     return cmd
1157 end, 'l3color')
1158 end
1159 </lua>
1160 <*luatex>
1161 <*package>
1162 \lua_load_module:n {l3backend-luatex}
1163 </package>

```

1164 \langle /luatex \rangle

4 I3backend-draw implementation

1165 \langle *package \rangle
1166 \langle @@=draw \rangle

4.1 dvips backend

1167 \langle *dvips \rangle

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

$_draw_backend_literal:e$ 1168 $\backslash cs_new_eq:Nn _draw_backend_literal:n _kernel_backend_literal_postscript:n$
1169 $\backslash cs_generate_variant:Nn _draw_backend_literal:n \{ e \}$

(End of definition for $_draw_backend_literal:n$.)

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

```
1170  $\backslash cs\_new\_protected:Npn \_draw\_backend\_begin:$   
1171 {  
1172    $\_draw\_backend\_literal:n \{ [begin] \}$   
1173    $\_draw\_backend\_literal:n \{ /draw.x~currentpoint~/draw.y~exch~def~def \}$   
1174    $\_draw\_backend\_literal:n \{ @beginspecial \}$   
1175 }  
1176  $\backslash cs\_new\_protected:Npn \_draw\_backend\_end:$   
1177 {  
1178    $\_draw\_backend\_literal:n \{ @endspecial \}$   
1179    $\_draw\_backend\_literal:n \{ [end] \}$   
1180 }
```

(End of definition for $_draw_backend_begin:$ and $_draw_backend_end:.$)

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

```
1181  $\backslash cs\_new\_protected:Npn \_draw\_backend\_scope\_begin:$   
1182 {  $\_draw\_backend\_literal:n \{ save \} \}$   
1183  $\backslash cs\_new\_protected:Npn \_draw\_backend\_scope\_end:$   
1184 {  $\_draw\_backend\_literal:n \{ restore \} \}$ 
```

(End of definition for $_draw_backend_scope_begin:$ and $_draw_backend_scope_end:.$)

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

```
1185  $\backslash cs\_new\_protected:Npn \_draw\_backend\_moveto:nn \#1\#2$   
1186 {  
1187    $\_draw\_backend\_literal:e$ 
```



```

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

```

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

```

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

```

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

```

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

```

```

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

```

```

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

```

(End of definition for __draw_backend_closepath: and others.)

__draw_backend_dash_pattern:nn
 __draw_backend_dash:n
 __draw_backend_linewidth:n
 __draw_backend_miterlimit:n
 __draw_backend_cap_but:tt
 __draw_backend_cap_round:tt
 __draw_backend_cap_rectangle:tt
 __draw_backend_join_miter:tt
 __draw_backend_join_round:tt
 __draw_backend_join_bevel:tt

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

```

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

```

```

1332 { \_draw_backend_literal:n { 0 ~ setlinejoin } }
1333 \cs_new_protected:Npn \_draw_backend_join_round:
1334 { \_draw_backend_literal:n { 1 ~ setlinejoin } }
1335 \cs_new_protected:Npn \_draw_backend_join_bevel:
1336 { \_draw_backend_literal:n { 2 ~ setlinejoin } }

```

(End of definition for `_draw_backend_dash_pattern:nn` and others.)

```

\_draw_backend_transform:nmmn
\_draw_backend_shift:nn

```

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

```

1337 \cs_new_protected:Npn \_draw_backend_transform:nmmn #1#2#3#4
1338 {
1339   \_draw_backend_literal:n
1340   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1341 }
1342 \cs_new_protected:Npn \_draw_backend_shift:nn #1#2
1343 {
1344   \_draw_backend_literal:n
1345   { [ 1 ~ 0 ~ 0 ~ 1 ~ #1 ~ #2 ] ~ concat }
1346 }

```

(End of definition for `_draw_backend_transform:nmmn` and `_draw_backend_shift:nn`.)

```

\_draw_backend_box_use:Nmmmm

```

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

```

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

```

```

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

```

(End of definition for _draw_backend_box_use:Nnnnn.)

```

1370 </dvips>

```

4.2 LuaTeX, pdfTeX, dvipdfmx and XeTeX

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

```

1371 <{*dvipdfmx | luatex | pdftex | xetex}

```

4.2.1 Drawing

_draw_backend_literal:n Pass data through using a dedicated interface.

```

\_draw_backend_literal:e 1372 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_pdf:n
1373 \cs_new_eq:NN \_draw_backend_literal:e \_kernel_backend_literal_pdf:e

```

(End of definition for _draw_backend_literal:n.)

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

```

\_draw_backend_end: 1374 \cs_new_protected:Npn \_draw_backend_begin:
1375 { \_draw_backend_scope_begin: }
1376 \cs_new_protected:Npn \_draw_backend_end:
1377 { \_draw_backend_scope_end: }

```

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

_draw_backend_scope_begin: Use the backend-level scope mechanisms.

```

\_draw_backend_scope_end: 1378 \cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:
1379 \cs_new_eq:NN \_draw_backend_scope_end: \_kernel_backend_scope_end:

```

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

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

```

\_draw_backend_lineto:nn 1380 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
\_draw_backend_curveto:nnnnn 1381 {
\_draw_backend_rectangle:nnnn 1382     \_draw_backend_literal:e
1383     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1384 }
1385 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1386 {
1387     \_draw_backend_literal:e
1388     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1389 }
1390 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1391 {
1392     \_draw_backend_literal:e
1393     {
1394         \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~

```

```

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

```

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

```

\__draw_backend_evenodd_rule: The even-odd rule here can be implemented as a simply switch.
\__draw_backend_nonzero_rule:
\g__draw_draw_eor_bool
1409 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1410 { \bool_gset_true:N \g__draw_draw_eor_bool }
1411 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1412 { \bool_gset_false:N \g__draw_draw_eor_bool }
1413 \bool_new:N \g__draw_draw_eor_bool

```

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

```

\__draw_backend_closepath: Converting paths to output is again a case of mapping directly to PDF operations.
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__draw_backend_discardpath:
1414 \cs_new_protected:Npn \__draw_backend_closepath:
1415 { \__draw_backend_literal:n { h } }
1416 \cs_new_protected:Npn \__draw_backend_stroke:
1417 { \__draw_backend_literal:n { S } }
1418 \cs_new_protected:Npn \__draw_backend_closestroke:
1419 { \__draw_backend_literal:n { s } }
1420 \cs_new_protected:Npn \__draw_backend_fill:
1421 {
1422     \__draw_backend_literal:e
1423     { f \bool_if:NT \g__draw_draw_eor_bool * }
1424 }
1425 \cs_new_protected:Npn \__draw_backend_fillstroke:
1426 {
1427     \__draw_backend_literal:e
1428     { B \bool_if:NT \g__draw_draw_eor_bool * }
1429 }
1430 \cs_new_protected:Npn \__draw_backend_clip:
1431 {
1432     \__draw_backend_literal:e
1433     { W \bool_if:NT \g__draw_draw_eor_bool * }
1434 }
1435 \cs_new_protected:Npn \__draw_backend_discardpath:
1436 { \__draw_backend_literal:n { n } }

```

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

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

```

    \_draw_backend_dash_pattern:nn
    \_draw_backend_dash:n      1437 \cs_new_protected:Npn \_draw_backend_dash_pattern:nn #1#2
    \_draw_backend_linewidth:n 1438 {
    \_draw_backend_miterlimit:n 1439   \_draw_backend_literal:e
    \_draw_backend_cap_but:    1440   {
    \_draw_backend_cap_round:  1441   [
    \_draw_backend_cap_rectangle: 1442   \exp_args:Nf \use:n
    \_draw_backend_join_miter:  1443   { \clist_map_function:nN {#1} \_draw_backend_dash:n }
    \_draw_backend_join_round:  1444   ] ~
    \_draw_backend_join_bevel:  1445   \dim_to_decimal_in_bp:n {#2} ~ d
    \_draw_backend_join_bevel:  1446   }
    \_draw_backend_join_bevel:  1447   }
    \cs_new:Npn \_draw_backend_dash:n #1
    \_draw_backend_dash:n      1448 { ~ \dim_to_decimal_in_bp:n {#1} }
    \_draw_backend_dash:n      1449 { ~ \dim_to_decimal_in_bp:n {#1} }
    \cs_new_protected:Npn \_draw_backend_linewidth:n #1
    \_draw_backend_dash:n      1450 \cs_new_protected:Npn \_draw_backend_linewidth:n #1
    \_draw_backend_dash:n      1451 {
    \_draw_backend_dash:n      1452   \_draw_backend_literal:e
    \_draw_backend_dash:n      1453   { \dim_to_decimal_in_bp:n {#1} ~ w }
    \_draw_backend_dash:n      1454   }
    \cs_new_protected:Npn \_draw_backend_miterlimit:n #1
    \_draw_backend_dash:n      1455 { \_draw_backend_literal:e { #1 ~ M } }
    \cs_new_protected:Npn \_draw_backend_cap_but:
    \_draw_backend_dash:n      1456 { \_draw_backend_literal:n { 0 ~ J } }
    \cs_new_protected:Npn \_draw_backend_cap_round:
    \_draw_backend_dash:n      1457 { \_draw_backend_literal:n { 1 ~ J } }
    \cs_new_protected:Npn \_draw_backend_cap_rectangle:
    \_draw_backend_dash:n      1458 { \_draw_backend_literal:n { 2 ~ J } }
    \cs_new_protected:Npn \_draw_backend_join_miter:
    \_draw_backend_dash:n      1459 { \_draw_backend_literal:n { 0 ~ j } }
    \cs_new_protected:Npn \_draw_backend_join_round:
    \_draw_backend_dash:n      1460 { \_draw_backend_literal:n { 1 ~ j } }
    \cs_new_protected:Npn \_draw_backend_join_bevel:
    \_draw_backend_dash:n      1461 { \_draw_backend_literal:n { 2 ~ j } }
    \cs_new_protected:Npn \_draw_backend_join_bevel:
    \_draw_backend_dash:n      1462 { \_draw_backend_literal:n { 2 ~ j } }
    \cs_new_protected:Npn \_draw_backend_join_bevel:
    \_draw_backend_dash:n      1463 { \_draw_backend_literal:n { 2 ~ j } }
    \cs_new_protected:Npn \_draw_backend_join_bevel:
    \_draw_backend_dash:n      1464 { \_draw_backend_literal:n { 2 ~ j } }
    \cs_new_protected:Npn \_draw_backend_join_bevel:
    \_draw_backend_dash:n      1465 { \_draw_backend_literal:n { 2 ~ j } }
    \cs_new_protected:Npn \_draw_backend_join_bevel:
    \_draw_backend_dash:n      1466 { \_draw_backend_literal:n { 2 ~ j } }
    \cs_new_protected:Npn \_draw_backend_join_bevel:
    \_draw_backend_dash:n      1467 { \_draw_backend_literal:n { 2 ~ j } }
    \cs_new_protected:Npn \_draw_backend_join_bevel:
    \_draw_backend_dash:n      1468 { \_draw_backend_literal:n { 2 ~ j } }

```

(End of definition for `_draw_backend_dash_pattern:nn` and others.)

```

    \_draw_backend_transform:nnnn
    \_draw_backend_transform_aux:nnnn
    \_draw_backend_shift:nn

```

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! As working out the rotation is relatively expensive, we optimize for the case where there is only a scaling.

```

1469 \cs_new_protected:Npn \_draw_backend_transform:nnnn #1#2#3#4
1470 {
1471 <*luatex | pdftex>
1472   \_kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1473 </luatex | pdftex>
1474 <*dvipdfmx | xetex>
1475   \str_if_eq:nnTF { #2 ~ #3 } { 0 ~ 0 }
1476   {

```

```

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

```

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

```

1512 \cs_new_protected:Npn \_draw_backend_shift:nn #1#2
1513 {
1514   \_draw_backend_literal:n
1515   { 1 ~ 0 ~ 0 ~ 1 ~ #1 ~ #2 ~ cm }
1516 }

```

(End of definition for `_draw_backend_transform:nnnn`, `_draw_backend_transform_aux:nnnn`, and `_draw_backend_shift:nn`.)

```

\_draw_backend_transform_decompose:nnnnN
draw_backend_transform_decompose_auxi:nnnnN
draw_backend_transform_decompose_auxii:nnnnN
draw_backend_transform_decompose_auxiii:nnnnN

```

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

```

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

```

```

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

```

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

`__draw_backend_box_use:Nnnnn`

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

```

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

```

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

```

1577 </dvipdfmx | luatex | pdftex | xetex>

```

4.3 dvisvgm backend

```

1578 <*dvisvgm>

```

The same as the more general literal call.

`__draw_backend_literal:n`
`__draw_backend_literal:e`

```

1579 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1580 \cs_generate_variant:Nn \__draw_backend_literal:n { e }

```

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

`__draw_backend_scope_begin:`
`__draw_backend_scope_end:`

Use the backend-level scope mechanisms.

```

1581 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1582 \cs_new_eq:NN \__draw_backend_scope_end: \__kernel_backend_scope_end:

```

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

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

```

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

```

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

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

```

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

```

```

1627     }
1628   }
1629   \tl_new:N \g__draw_backend_path_tl

```

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

```

\__draw_backend_evenodd_rule: The fill rules here have to be handled as scopes.
\__draw_backend_nonzero_rule: 1630 \cs_new_protected:Npn \__draw_backend_evenodd_rule:
                               1631   { \__kernel_backend_scope:n { fill-rule="evenodd" } }
                               1632 \cs_new_protected:Npn \__draw_backend_nonzero_rule:
                               1633   { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

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

```

\__draw_backend_path:n Setting fill and stroke effects and doing clipping all has to be done using scopes. This
\__draw_backend_closepath: means setting up the various requirements in a shared auxiliary which deals with the
\__draw_backend_stroke: bits and pieces. Clipping paths are reused for path drawing; not essential but avoids
\__draw_backend_closestroke: constructing them twice. Discarding a path needs a separate function as it's not quite
\__draw_backend_fill: the same.
\__draw_backend_fillstroke: 1634 \cs_new_protected:Npn \__draw_backend_closepath:
\__draw_backend_clip: 1635   { \__draw_backend_add_to_path:n { Z } }
\__draw_backend_discardpath: 1636 \cs_new_protected:Npn \__draw_backend_path:n #1
\g__draw_draw_clip_bool 1637   {
\g__draw_draw_path_int 1638     \bool_if:NTF \g__draw_draw_clip_bool
1639     {
1640       \int_gincr:N \g__kernel_clip_path_int
1641       \__draw_backend_literal:e
1642       {
1643         < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1644         { ?nl }
1645         <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1646         < /clipPath > { ? nl }
1647         <
1648         use~xlink:href =
1649         "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1650         #1
1651         />
1652       }
1653       \__kernel_backend_scope:e
1654       {
1655         clip-path =
1656         "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1657       }
1658     }
1659   {
1660     \__draw_backend_literal:e
1661     { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1662   }
1663   \tl_gclear:N \g__draw_backend_path_tl
1664   \bool_gset_false:N \g__draw_draw_clip_bool
1665 }
1666 \int_new:N \g__draw_backend_path_int
1667 \cs_new_protected:Npn \__draw_backend_stroke:
1668   { \__draw_backend_path:n { style="fill:none" } }

```

```

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

```

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

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

```

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

```

```

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

```

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

`__draw_backend_transform:nmmn` The four arguments here are floats (the affine matrix), the last two are a displacement vector.
`__draw_backend_shift:nn`

```

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

```

(End of definition for `__draw_backend_transform:nmmn` and `__draw_backend_shift:nn`.)

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

```

1758 \cs_new_protected:Npn \__draw_backend_box_use:Nmmmm #1#2#3#4#5
1759 {
1760     \__kernel_backend_scope_begin:
1761     \__draw_backend_transform:nmmn {#2} {#3} {#4} {#5}

```

```

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

```

(End of definition for __draw_backend_box_use:Nnnnn.)

```
1776 </dvisvgm>
```

```
1777 </package>
```

5 l3backend-graphics implementation

```

1778 <*package>
1779 <@@=graphics>

```

5.1 dvips backend

```
1780 <*dvips>
```

```
\l_graphics_search_ext_seq
```

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

(End of definition for \l_graphics_search_ext_seq.)

_graphics_backend_getbb_eps:n Simply use the generic function.

```

\_graphics_backend_getbb_eps:n 1782 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
\_graphics_backend_getbb_ps:n 1783 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n

```

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

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

```

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

```

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

`_graphics_backend_get_pagecount:n`

```
1796 \cs_new_eq:NN \_graphics_backend_get_pagecount:n \_graphics_get_pagecount:n
(End of definition for \_graphics_backend_get_pagecount:n.)
1797 </dvips>
```

5.2 LuaTeX and pdfTeX backends

```
1798 <*luatex | pdftex>
```

`\l_graphics_search_ext_seq`

```
1799 \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1800 { .pdf , .eps , .ps , .png , .jpg , .jpeg }
(End of definition for \l_graphics_search_ext_seq.)
```

`\l__graphics_attr_tl`

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

```
1801 \tl_new:N \l__graphics_attr_tl
```

(End of definition for `\l__graphics_attr_tl`.)

`_graphics_backend_getbb_jpg:n`

`_graphics_backend_getbb_jpeg:n`

`_graphics_backend_getbb_pdf:n`

`_graphics_backend_getbb_png:n`

`_graphics_backend_getbb_auxi:n`

`_graphics_backend_getbb_auxii:n`

`_graphics_backend_getbb_auxiii:n`

`_graphics_backend_dequote:w`

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

```
1802 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
1803 {
1804   \int_zero:N \l__graphics_page_int
1805   \tl_clear:N \l__graphics_pagebox_tl
1806   \tl_set:Ne \l__graphics_attr_tl
1807     {
1808       \tl_if_empty:NF \l__graphics_decodearray_str
1809         { :D \l__graphics_decodearray_str }
1810       \bool_if:NT \l__graphics_interpolate_bool
1811         { :I }
1812       \str_if_empty:NF \l__graphics_pdf_str
1813         { :X \l__graphics_pdf_str }
1814     }
1815   \_graphics_backend_getbb_auxi:n {#1}
1816 }
1817 \cs_new_eq:NN \_graphics_backend_getbb_jpeg:n \_graphics_backend_getbb_jpg:n
1818 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
1819 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
1820 {
1821   \tl_clear:N \l__graphics_decodearray_str
1822   \bool_set_false:N \l__graphics_interpolate_bool
1823   \tl_set:Ne \l__graphics_attr_tl
1824     {
1825     : \l__graphics_pagebox_tl
```



```

1826     \int_compare:nNnT \l__graphics_page_int > 1
1827     { :P \int_use:N \l__graphics_page_int }
1828     \str_if_empty:NF \l__graphics_pdf_str
1829     { :X \l__graphics_pdf_str }
1830   }
1831   \__graphics_backend_getbb_auxi:n {#1}
1832 }
1833 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1834 {
1835   \__graphics_bb_restore:eF { #1 \l__graphics_attr_tl }
1836   { \__graphics_backend_getbb_auxii:n {#1} }
1837 }

```

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

```

1838 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1839 {
1840   \exp_args:Ne \__graphics_backend_getbb_auxiii:n
1841   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1842   \int_const:cn { c__graphics_#1 \l__graphics_attr_tl_int }
1843   { \tex_the:D \tex_pdflastximage:D }
1844   \__graphics_bb_save:e { #1 \l__graphics_attr_tl }
1845 }
1846 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1847 {
1848   \tex_immediate:D \tex_pdfximage:D
1849   \bool_lazy_any:nT
1850   {
1851     { \l__graphics_interpolate_bool }
1852     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1853     { ! \str_if_empty_p:N \l__graphics_pdf_str }
1854   }
1855   {
1856     attr ~
1857     {
1858       \tl_if_empty:NF \l__graphics_decodearray_str
1859       { /Decode~[ \l__graphics_decodearray_str ] }
1860       \bool_if:NT \l__graphics_interpolate_bool
1861       { /Interpolate~true }
1862       \l__graphics_pdf_str
1863     }
1864   }
1865   \int_compare:nNnT \l__graphics_page_int > 0
1866   { page ~ \int_use:N \l__graphics_page_int }
1867   \tl_if_empty:NF \l__graphics_pagebox_tl
1868   { \l__graphics_pagebox_tl }
1869   {#1}
1870   \hbox_set:Nn \l__graphics_tmp_box
1871   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1872   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_tmp_box }
1873   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_tmp_box }
1874 }

```

```
1875 \cs_new:Npn \__graphics_backend_dequote:w #1 " #2 " #3 \s__graphics_stop {#2}
```

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

__graphics_backend_include_jpg:n 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.

```
\__graphics_backend_include_jpeg:n
\_graphics_backend_include_pdf:n
\_graphics_backend_include_png:n
```

```
1876 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1877 {
1878   \tex_pdfrefximage:D
1879   \int_use:c { c__graphics_ #1 \l__graphics_attr_tl_int }
1880 }
1881 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1882 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1883 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
```

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

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

```
\__graphics_backend_getbb_ps:n
\_graphics_backend_getbb_eps:nn
\_graphics_backend_include_eps:n
\_graphics_backend_include_ps:n
```

```
1884 \sys_if_shell:T
1885 {
1886   \str_new:N \l__graphics_backend_dir_str
1887   \str_new:N \l__graphics_backend_name_str
1888   \str_new:N \l__graphics_backend_ext_str
1889   \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1890   {
1891     \file_parse_full_name:nNNN {#1}
1892     \l__graphics_backend_dir_str
1893     \l__graphics_backend_name_str
1894     \l__graphics_backend_ext_str
1895     \exp_args:Ne \__graphics_backend_getbb_eps:nn
1896     {
1897       \exp_args:Ne \__kernel_file_name_quote:n
1898       {
1899         \l__graphics_backend_name_str
1900         - \str_tail:N \l__graphics_backend_ext_str
1901         -converted-to.pdf
1902       }
1903     }
1904     {#1}
1905   }
1906   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_backend_getbb_eps:n
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_final_name_str {#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:Ne \__graphics_backend_include_pdf:n
1922   {
1923     \exp_args:Ne \__kernel_file_name_quote:n
1924     {
1925       \l__graphics_backend_name_str
1926       - \str_tail:N \l__graphics_backend_ext_str
1927       -converted-to.pdf
1928     }
1929   }
1930 }
1931 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1932 }

```

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

__graphics_backend_get_pagecount:n Simply load and store.

```

1933 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1934 {
1935   \tex_pdfximage:D {#1}
1936   \int_const:cn { c__graphics_#1_pages_int }
1937   { \int_use:N \tex_pdflastximagepages:D }
1938 }

```

(End of definition for __graphics_backend_get_pagecount:n.)

```
1939 </luatex | pdftex>
```

5.3 dvipdfmx backend

```
1940 <*dvipdfmx | xetex>
```

\l__graphics_search_ext_seq

```

1941 \seq_set_from_clist:Nn \l__graphics_search_ext_seq
1942 { .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }

```

(End of definition for \l__graphics_search_ext_seq.)

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

```

\__graphics_backend_getbb_ps:n 1943 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
\__graphics_backend_getbb_jpg:n 1944 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
\__graphics_backend_getbb_jpeg:n 1945 <*dvipdfmx>
\__graphics_backend_getbb_pdf:n 1946 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
\__graphics_backend_getbb_png:n 1947 {
\__graphics_backend_getbb_bmp:n 1948   \int_zero:N \l__graphics_page_int
1949   \tl_clear:N \l__graphics_pagebox_tl
1950   \__graphics_extract_bb:n {#1}
1951 }
1952 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1953 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1954 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
1955 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1

```

```

1956 {
1957   \tl_clear:N \l__graphics_decodearray_str
1958   \bool_set_false:N \l__graphics_interpolate_bool
1959   \__graphics_extract_bb:n {#1}
1960 }
1961 </dviptdvmx>

```

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

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

```

1962 \int_new:N \g__graphics_track_int

```

(End of definition for `\g__graphics_track_int`.)

`__graphics_backend_include_eps:n` The special syntax depends on the file type. There is a difference in how PDF graphics are best handled between `dviptdvmx` and `XYLATEX`: 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_ps:n
\__graphics_backend_include_png:n
\__graphics_backend_include_jpeg:n
\__graphics_backend_include_jpseg:n
\__graphics_backend_include_pdf:n
\__graphics_backend_include_png:n
\__graphics_backend_include_bmp:n
\__graphics_backend_include_auxi:nn
\__graphics_backend_include_auxii:nnn
\__graphics_backend_include_auxiii:enn
\__graphics_backend_include_auxiiii:nnnn
1963 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1964 {
1965   \__kernel_backend_literal:e
1966   {
1967     PSfile = #1 \c_space_tl
1968     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1969     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1970     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1971     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1972   }
1973 }
1974 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1975 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1976 { \__graphics_backend_include_auxi:nn {#1} { image } }
1977 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1978 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
1979 \cs_new_eq:NN \__graphics_backend_include_bmp:n \__graphics_backend_include_jpg:n
1980 <*dviptdvmx>
1981 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1982 { \__graphics_backend_include_auxii:enn {#1} { epdf } }
1983 </dviptdvmx>

```

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.

```

1984 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1985 {
1986   \__graphics_backend_include_auxii:enn
1987   {
1988     \tl_if_empty:NF \l__graphics_pagebox_tl
1989     { : \l__graphics_pagebox_tl }
1990     \int_compare:nNnT \l__graphics_page_int > 1
1991     { :P \int_use:N \l__graphics_page_int }
1992     \tl_if_empty:NF \l__graphics_decodearray_str
1993     { :D \l__graphics_decodearray_str }
1994     \bool_if:NT \l__graphics_interpolate_bool

```

```

1995         { :I }
1996     }
1997     {#1} {#2}
1998 }
1999 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
2000 {
2001     \int_if_exist:cTF { c__graphics_ #2#1 _int }
2002     {
2003         \__kernel_backend_literal:e
2004         { pdf:useobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
2005     }
2006     { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
2007 }
2008 \cs_generate_variant:Nn \__graphics_backend_include_auxii:nnn { e }

```

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

```

2009 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
2010 {
2011     \int_gincr:N \g__graphics_track_int
2012     \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
2013     \__kernel_backend_literal:e
2014     {
2015         pdf:#3~
2016         @graphic \int_use:c { c__graphics_ #1#2 _int } ~
2017         \int_compare:nNnT \l__graphics_page_int > 1
2018         { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2019         \tl_if_empty:NF \l__graphics_pagebox_tl
2020         {
2021             pagebox ~ \l__graphics_pagebox_tl \c_space_tl
2022             bbox ~
2023             \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2024             \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2025             \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2026             \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
2027         }
2028         (#1)
2029         \bool_lazy_or:nnT
2030         { \l__graphics_interpolate_bool }
2031         { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
2032         {
2033             <<
2034             \tl_if_empty:NF \l__graphics_decodearray_str
2035             { /Decode~[ \l__graphics_decodearray_str ] }
2036             \bool_if:NT \l__graphics_interpolate_bool
2037             { /Interpolate~true }
2038             >>
2039         }
2040     }
2041 }

```

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

`__graphics_backend_get_pagecount:n`

```

2042 <*dvipdfmx>
2043 \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n
2044 </dvipdfmx>

(End of definition for \__graphics_backend_get_pagecount:n.)

2045 </dvipdfmx | xetex>

```

5.4 X_YTeX backend

```

2046 <*xetex>

```

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_jpeg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
\__graphics_backend_getbb_bmp:n
\__graphics_backend_getbb_auxi:nN
\__graphics_backend_getbb_auxii:nNn
\__graphics_backend_getbb_auxii:VnN
\__graphics_backend_getbb_auxiii:nNnn
\__graphics_backend_getbb_auxiv:nNnn
\__graphics_backend_getbb_auxiv:VnNnn
\__graphics_backend_getbb_auxv:nNnn
\__graphics_backend_getbb_auxv:VnNnn
\__graphics_backend_getbb_pagebox:w

```

```

2047 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2048 {
2049   \int_zero:N \l__graphics_page_int
2050   \tl_clear:N \l__graphics_pagebox_tl
2051   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2052 }
2053 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2054 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
2055 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2056 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2057 {
2058   \tl_clear:N \l__graphics_decodearray_str
2059   \bool_set_false:N \l__graphics_interpolate_bool
2060   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2061 }
2062 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2063 {
2064   \int_compare:nNnTF \l__graphics_page_int > 1
2065     { \__graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2066     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2067 }
2068 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nN #1#2#3
2069 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2070 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nN { V }
2071 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2072 {
2073   \tl_if_empty:NTF \l__graphics_pagebox_tl
2074     { \__graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2075     { \__graphics_backend_getbb_auxv:nNnn }
2076     {#1} #2 {#3} {#4}
2077 }
2078 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nNnn #1#2#3#4#5
2079 {
2080   \use:e
2081   {
2082     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2083     {
2084       #5
2085       \tl_if_blank:nF {#1}

```

```

2086         { \c_space_tl \_graphics_backend_getbb_pagebox:w #1 }
2087     }
2088 }
2089 }
2090 \cs_generate_variant:Nn \_graphics_backend_getbb_auxiv:nnNnn { V }
2091 \cs_new_protected:Npn \_graphics_backend_getbb_auxv:nNnn #1#2#3#4
2092 {
2093     \_graphics_bb_restore:nF {#1#3}
2094     { \_graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2095 }
2096 \cs_new_protected:Npn \_graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2097 {
2098     \hbox_set:Nn \l__graphics_tmp_box { #2 #1 ~ #4 }
2099     \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_tmp_box }
2100     \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_tmp_box }
2101     \_graphics_bb_save:n {#1#3}
2102 }
2103 \cs_new:Npn \_graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End of definition for `_graphics_backend_getbb_jpg:n` and others.)

`_graphics_backend_include_pdf:n` For PDF graphics, properly supporting the pagebox concept in X_YT_EX is best done using the `\tex_XeTeXpdfxfile: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`.

```

2104 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
2105 {
2106     \tex_XeTeXpdfxfile:D #1 ~
2107     \int_compare:nNnT \l__graphics_page_int > 0
2108     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2109     \exp_after:wN \_graphics_backend_getbb_pagebox:w \l__graphics_pagebox_tl
2110 }

```

(End of definition for `_graphics_backend_include_pdf:n`.)

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

```

2111 \cs_new_protected:Npn \_graphics_backend_get_pagecount:n #1
2112 {
2113     \int_const:cn { c__graphics_ #1 _pages_int }
2114     {
2115         \int_max:nn
2116         { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2117         { 1 }
2118     }
2119 }

```

(End of definition for `_graphics_backend_get_pagecount:n`.)

```

2120 </xetex>

```

5.5 dvisvgm backend

2121 `*dvisvgm)`

`\l_graphics_search_ext_seq`

```
2122 \seq_set_from_clist:Nn \l_graphics_search_ext_seq
2123 { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }
```

(End of definition for `\l_graphics_search_ext_seq`.)

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

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

```
2124 \cs_new_protected:Npn \_graphics_backend_getbb_svg:n #1
2125 {
2126   \_graphics_bb_restore:nF {#1}
2127   {
2128     \ior_open:Nn \l__graphics_tmp_ior {#1}
2129     \ior_if_eof:NTF \l__graphics_tmp_ior
2130       { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2131       {
2132         \dim_zero:N \l__graphics_llx_dim
2133         \dim_zero:N \l__graphics_lly_dim
2134         \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2135         \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2136         \ior_map_inline:Nn \l__graphics_tmp_ior
2137           {
2138             \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2139               {
2140                 \_graphics_backend_getbb_svg_auxi:nNn
2141                   { width } \l__graphics_urx_dim {##1}
2142               }
2143             \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2144               {
2145                 \_graphics_backend_getbb_svg_auxi:nNn
2146                   { height } \l__graphics_ury_dim {##1}
2147               }
2148             \bool_lazy_and:nnF
2149               { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2150               { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2151               { \ior_map_break: }
2152           }
2153         \_graphics_bb_save:n {#1}
2154       }
2155     \ior_close:N \l__graphics_tmp_ior
2156   }
2157 }
2158 \cs_new_protected:Npn \_graphics_backend_getbb_svg_auxi:nNn #1#2#3
2159 {
2160   \use:e
2161   {
2162     \cs_set_protected:Npn \_graphics_backend_getbb_svg_auxii:w
2163       ##1 \tl_to_str:n {#1} = ##2 \tl_to_str:n {#1} = ##3
2164     \s__graphics_stop
```



```

2165     }
2166     {
2167     \tl_if_blank:nF {##2}
2168     {
2169     \peek_remove_spaces:n
2170     {
2171     \peek_meaning:NTF ' % '
2172     { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2173     {
2174     \peek_meaning:NTF " % "
2175     { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2176     { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2177     }
2178     }
2179     ##2 \s__graphics_stop
2180     }
2181     }
2182     \use:e
2183     {
2184     \__graphics_backend_getbb_svg_auxii:w #3
2185     \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2186     \s__graphics_stop
2187     }
2188     }
2189     \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2190     \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop
2191     { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2192     \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2193     { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2194     \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2195     { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2196     \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2197     {
2198     \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2199     \l__graphics_tmp_dim #2 bp \scan_stop:
2200     \dim_set_eq:NN #1 \l__graphics_tmp_dim
2201     }
2202     \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvii:w #1 \scan_stop: { }

```

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

__graphics_backend_getbb_eps:n
 __graphics_backend_getbb_ps:n

Simply use the generic function.

```

2203 \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2204 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n

```

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

__graphics_backend_getbb_png:n
 __graphics_backend_getbb_jpg:n
 __graphics_backend_getbb_jpeg:n

These can be included by extracting the bounding box data.

```

2205 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2206 {
2207     \int_zero:N \l__graphics_page_int
2208     \tl_clear:N \l__graphics_pagebox_tl
2209     \__graphics_extract_bb:n {#1}
2210 }
2211 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n

```

```
2212 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
```

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

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

```
2213 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2214 {
2215   \tl_clear:N \l__graphics_decodearray_str
2216   \bool_set_false:N \l__graphics_interpolate_bool
2217   \__graphics_extract_bb:n {#1}
2218 }
```

(End of definition for __graphics_backend_getbb_pdf:n.)

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

```
2219 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2220 { \__graphics_backend_include:nn { PSfile } {#1} }
2221 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
2222 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2223 { \__graphics_backend_include:nn { pdffile } {#1} }
2224 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
2225 {
2226   \__kernel_backend_literal:e
2227   {
2228     #1 = #2 \c_space_tl
2229     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2230     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2231     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2232     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2233   }
2234 }
```

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

__graphics_backend_include_svg:n __graphics_backend_include_png:n __graphics_backend_include_jpg:n __graphics_backend_include_jpeg:n __graphics_backend_include_dequote:w The backend here has built-in support for basic graphic inclusion (see dvisvgm.def for a more complex approach, needed if clipping, etc., is covered at the graphic backend level). We have to deal with the fact that the image reference point is at the *top*, so there is a need for a vertical shift to put it in the right place. The other issue is that #1 must be quote-corrected. The dvisvgm:img operation quotes the file name, but if it is already quoted (contains spaces) then we have an issue: we simply strip off any quotes as a result.

```
2235 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2236 {
2237   \box_move_up:nn { \l__graphics_ury_dim }
2238   {
2239     \hbox:n
2240     {
2241       \__kernel_backend_literal:e
2242       {
2243         dvisvgm:img~
2244         \dim_to_decimal:n { \l__graphics_urx_dim } ~
2245         \dim_to_decimal:n { \l__graphics_ury_dim } ~
2246         \__graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
```

```

2247         }
2248     }
2249 }
2250 }
2251 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_svg:n
2252 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_svg:n
2253 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_svg:n
2254 \cs_new:Npn \__graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2255     {#2}

```

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

__graphics_backend_get_pagecount:n

```

2256 \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n

```

(End of definition for __graphics_backend_get_pagecount:n.)

```

2257 </dvisvgm>

```

```

2258 </package>

```

6 I3backend-pdf implementation

```

2259 <*package>

```

```

2260 <@@=pdf>

```

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

6.1 dvips backend

```

2261 <*dvips>

```

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

__pdf_backend_pdfmark:e

```

2262 \cs_new_protected:Npn \__pdf_backend_pdfmark:n #1
2263     { \__kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2264 \cs_generate_variant:Nn \__pdf_backend_pdfmark:n { e }

```

(End of definition for __pdf_backend_pdfmark:n.)

6.1.1 Catalogue entries

__pdf_backend_catalog_gput:nn

__pdf_backend_info_gput:nn

```

2265 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2266     { \__pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2267 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2268     { \__pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }

```

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

6.1.2 Objects

```

\__pdf_backend_object_new:
\__pdf_backend_object_ref:n 2269 \cs_new_protected:Npn \__pdf_backend_object_new:
\__pdf_backend_object_id:n 2270 { \int_gincr:N \g__pdf_backend_object_int }
2271 \cs_new:Npn \__pdf_backend_object_ref:n #1 { { pdf.obj #1 } }
2272 \cs_new_eq:NN \__pdf_backend_object_id:n \__pdf_backend_object_ref:n

```

(End of definition for __pdf_backend_object_new:, __pdf_backend_object_ref:n, and __pdf_backend_object_id:n.)

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

```

\__pdf_backend_object_write:nnn 2273 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
\__pdf_backend_object_write:nne 2274 {
\__pdf_backend_object_write_aux:nnn 2275 \__pdf_backend_object_write_aux:nnn
\__pdf_backend_object_write_array:nn 2276 { \__pdf_backend_object_ref:n {#1} }
\__pdf_backend_object_write_dict:nn 2277 {#2} {#3}
\__pdf_backend_object_write_fstream:nn 2278 }
\__pdf_backend_object_write_stream:nnn 2279 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
\__pdf_backend_object_write_stream:nnn 2280 \cs_new_protected:Npn \__pdf_backend_object_write_aux:nnn #1#2#3
2281 {
2282 \__pdf_backend_pdfmark:e
2283 {
2284 /_objdef ~ #1
2285 /type
2286 \str_case:nn {#2}
2287 {
2288 { array } { /array }
2289 { dict } { /dict }
2290 { fstream } { /stream }
2291 { stream } { /stream }
2292 }
2293 /OBJ
2294 }
2295 \use:c { __pdf_backend_object_write_ #2 :nn } {#1} {#3}
2296 }
2297 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2298 {
2299 \__pdf_backend_pdfmark:e
2300 { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2301 }
2302 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2303 {
2304 \__pdf_backend_pdfmark:e
2305 { #1 << \exp_not:n {#2} >> /PUT }
2306 }
2307 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2308 {
2309 \exp_args:Ne
2310 \__pdf_backend_object_write_fstream:nnn {#1} #2
2311 }
2312 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2313 {

```

```

2314   \__kernel_backend_postscript:n
2315   {
2316     SDict ~ begin ~
2317     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2318     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2319     end
2320   }
2321 }
2322 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2323 {
2324   \exp_args:Ne
2325   \__pdf_backend_object_write_stream:nnn {#1} #2
2326 }
2327 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2328 {
2329   \__kernel_backend_postscript:n
2330   {
2331     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2332     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2333   }
2334 }

```

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

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

```

\__pdf_backend_object_now:ne
2335 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2336 {
2337   \int_gincr:N \g__pdf_backend_object_int
2338   \__pdf_backend_object_write_aux:nnn
2339   { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2340   {#1} {#2}
2341 }
2342 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last: Much like the annotation version.

```

2343 \cs_new:Npn \__pdf_backend_object_last:
2344 { { pdf.obj \int_use:N \g__pdf_backend_object_int } }

```

(End of definition for __pdf_backend_object_last:.)

__pdf_backend_pageobject_ref:n Page references are easy in dvips.

```

2345 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2346 { { Page #1 } }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.1.3 Destinations

`__pdf_backend_destination:nn`
`__pdf_backend_destination:nmnn`
`__pdf_backend_destination_aux:nmnn`

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

```

2347 \cs_new_protected:Npn __pdf_backend_destination:nn #1#2
2348 {
2349   __kernel_backend_postscript:n { pdf.dest.anchor }
2350   __pdf_backend_pdfmark:e
2351   {
2352     /View
2353     [
2354       \str_case:nnF {#2}
2355       {
2356         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2357         { fit } { /Fit }
2358         { fitb } { /FitB }
2359         { fitbh } { /FitBH ~ pdf.dest.y }
2360         { fitbv } { /FitBV ~ pdf.dest.x }
2361         { fith } { /FitH ~ pdf.dest.y }
2362         { fitv } { /FitV ~ pdf.dest.x }
2363         { fitr } { /Fit }
2364       }
2365       {
2366         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2367       }
2368     ]
2369     /Dest ( \exp_not:n {#1} ) cvn
2370     /DEST
2371   }
2372 }
2373 \cs_new_protected:Npn __pdf_backend_destination:nmnn #1#2#3#4
2374 {
2375   \exp_args:Ne __pdf_backend_destination_aux:nmnn
2376   { \dim_eval:n {#2} } {#1} {#3} {#4}
2377 }
2378 \cs_new_protected:Npn __pdf_backend_destination_aux:nmnn #1#2#3#4
2379 {
2380   \vbox_to_zero:n
2381   {
2382     __kernel_kern:n {#4}
2383     \hbox:n { __kernel_backend_postscript:n { pdf.save.ll } }
2384     \tex_vss:D
2385   }
2386   __kernel_kern:n {#1}
2387   \vbox_to_zero:n
2388   {
2389     __kernel_kern:n { -#3 }
2390     \hbox:n { __kernel_backend_postscript:n { pdf.save.ur } }
2391     \tex_vss:D
2392   }
2393   __kernel_kern:n { -#1 }
2394   __pdf_backend_pdfmark:n

```

```

2395     {
2396     /View
2397     [
2398     /FitR ~
2399     pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2400     pdf.urx ~ pdf.ury ~ pdf.dest2device
2401     ]
2402     /Dest ( #2 ) cvn
2403     /DEST
2404     }
2405 }

```

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

6.1.4 Structure

`__pdf_backend_compresslevel:n` Doable for the usual ps2pdf method.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
2406 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2407 {
2408   \int_compare:nNnT {#1} = 0
2409   {
2410     \__kernel_backend_literal_postscript:n
2411     {
2412       /setdistillerparams ~ where
2413       { pop << /CompressPages ~ false >> setdistillerparams }
2414       if
2415     }
2416   }
2417 }
2418 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2419 {
2420   \bool_if:nF {#1}
2421   {
2422     \__kernel_backend_literal_postscript:n
2423     {
2424       /setdistillerparams ~ where
2425       { pop << /CompressStreams ~ false >> setdistillerparams }
2426       if
2427     }
2428   }
2429 }

```

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

```

\__pdf_backend_version_major_gset:n
\__pdf_backend_version_minor_gset:n
2430 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2431 {
2432   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2433 }
2434 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2435 {
2436   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2437 }

```

(End of 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: 2438 \cs_new:Npn \_pdf_backend_version_major: { -1 }  
2439 \cs_new:Npn \_pdf_backend_version_minor: { -1 }
```

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

6.1.5 Marked content

```
\_pdf_backend_bdc:nn Simple wrappers.  
\_pdf_backend_emc: 2440 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2  
2441 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }  
2442 \cs_new_protected:Npn \_pdf_backend_emc:  
2443 { \_pdf_backend_pdfmark:n { /EMC } }  
  
(End of definition for \_pdf_backend_bdc:nn and \_pdf_backend_emc:.)  
  
2444 </dvips>
```

6.2 LuaTeX and pdfTeX backend

```
2445 <*luatex | pdftex>
```

6.2.1 Destinations

```
\_pdf_backend_destination:nn A simple task: pass the data to the primitive. The \scan_stop: deals with the danger  
\_pdf_backend_destination:nnnn 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.
```

```
2446 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2  
2447 {  
2448 <*luatex>  
2449 \tex_pdfextension:D dest ~  
2450 </luatex>  
2451 <*pdftex>  
2452 \tex_pdfdest:D  
2453 </pdftex>  
2454 name {#1}  
2455 \str_case:nnF {#2}  
2456 {  
2457 { xyz } { xyz }  
2458 { fit } { fit }  
2459 { fitb } { fitb }  
2460 { fitbh } { fitbh }  
2461 { fitbv } { fitbv }  
2462 { fith } { fith }  
2463 { fitv } { fitv }  
2464 { fitr } { fitr }  
2465 }  
2466 { xyz ~ zoom \fp_eval:n { #2 * 10 } }  
2467 \scan_stop:  
2468 }  
2469 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4  
2470 {
```



```

2471 <*luatex>
2472   \tex_pdfextension:D dest ~
2473 </luatex>
2474 <*pdftex>
2475   \tex_pdfdest:D
2476 </pdftex>
2477   name {#1}
2478   fitr ~
2479   width \dim_eval:n {#2} ~
2480   height \dim_eval:n {#3} ~
2481   depth \dim_eval:n {#4} \scan_stop:
2482 }

```

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

6.2.2 Catalogue entries

```

\__pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn
2483 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2484 {
2485 <*luatex>
2486   \tex_pdfextension:D catalog
2487 </luatex>
2488 <*pdftex>
2489   \tex_pdfcatalog:D
2490 </pdftex>
2491   { / #1 ~ #2 }
2492 }
2493 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2494 {
2495 <*luatex>
2496   \tex_pdfextension:D info
2497 </luatex>
2498 <*pdftex>
2499   \tex_pdfinfo:D
2500 </pdftex>
2501   { / #1 ~ #2 }
2502 }

```

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

6.2.3 Objects

`\g__pdf_backend_object_prop` For tracking objects to allow finalization.

```

2503 \prop_new:N \g__pdf_backend_object_prop

```

(End of definition for `\g__pdf_backend_object_prop`.)

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

```

\__pdf_backend_object_ref:n
\__pdf_backend_object_id:n
2504 \cs_new_protected:Npn \__pdf_backend_object_new:
2505 {
2506 <*luatex>
2507   \tex_pdfextension:D obj ~
2508 </luatex>

```

```

2509 <*pdftex>
2510     \tex_pdfobj:D
2511 </pdftex>
2512     reserveobjnum ~
2513     \int_gset:Nn \g__pdf_backend_object_int
2514 <*luatex>
2515     { \tex_pdffeedback:D lastobj }
2516 </luatex>
2517 <*pdftex>
2518     { \tex_pdflastobj:D }
2519 </pdftex>
2520 }
2521 \cs_new:Npn \__pdf_backend_object_ref:n #1 { #1 ~ 0 ~ R }
2522 \cs_new:Npn \__pdf_backend_object_id:n #1 {#1}

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

```

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

```

\__pdf_backend_object_write:nne
\__pdf_backend_object_write:nn
\__pdf_exp_not_i:nn
\__pdf_exp_not_ii:nn
2523 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2524 {
2525 <*luatex>
2526     \tex_immediate:D \tex_pdfextension:D obj ~
2527 </luatex>
2528 <*pdftex>
2529     \tex_immediate:D \tex_pdfobj:D
2530 </pdftex>
2531     useobjnum ~ #1
2532     \__pdf_backend_object_write:nn {#2} {#3}
2533 }
2534 \cs_new:Npn \__pdf_backend_object_write:nn #1#2
2535 {
2536     \str_case:nn {#1}
2537     {
2538         { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2539         { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2540         { fstream }
2541         {
2542             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2543             file ~ { \__pdf_exp_not_ii:nn #2 }
2544         }
2545         { stream }
2546         {
2547             stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2548             { \__pdf_exp_not_ii:nn #2 }
2549         }
2550     }
2551 }
2552 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2553 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2554 \cs_new:Npn \__pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

```

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

__pdf_backend_object_now:nn Much like writing, but direct creation.

__pdf_backend_object_now:ne

```

2555 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2556 {
2557 <*luatex>
2558   \tex_immediate:D \tex_pdfextension:D obj ~
2559 </luatex>
2560 <*pdftex>
2561   \tex_immediate:D \tex_pdfobj:D
2562 </pdftex>
2563   \__pdf_backend_object_write:nn {#1} {#2}
2564 }
2565 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last: Much like annotation.

```

2566 \cs_new:Npe \__pdf_backend_object_last:
2567 {
2568   \exp_not:N \int_value:w
2569 <*luatex>
2570   \exp_not:N \tex_pdffeedback:D lastobj ~
2571 </luatex>
2572 <*pdftex>
2573   \exp_not:N \tex_pdflastobj:D
2574 </pdftex>
2575   \c_space_tl 0 ~ R
2576 }

```

(End of definition for __pdf_backend_object_last:.)

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

```

2577 \cs_new:Npe \__pdf_backend_pageobject_ref:n #1
2578 {
2579   \exp_not:N \int_value:w
2580 <*luatex>
2581   \exp_not:N \tex_pdffeedback:D pageref
2582 </luatex>
2583 <*pdftex>
2584   \exp_not:N \tex_pdfpageref:D
2585 </pdftex>
2586   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2587 }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.2.4 Structure

__pdf_backend_compresslevel:n Simply pass data to the engine.

```

\__pdf_backend_compresslevel:n
\__pdf_backend_compress_objects:n
\__pdf_backend_objcompresslevel:n
2588 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2589 {
2590   \tex_global:D
2591 <*luatex>
2592   \tex_pdfvariable:D compresslevel
2593 </luatex>
2594 <*pdftex>
2595   \tex_pdfcompresslevel:D

```

```

2596 </pdftex>
2597     \int_value:w \int_eval:n {#1} \scan_stop:
2598 }
2599 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2600 {
2601     \bool_if:nTF {#1}
2602         { \__pdf_backend_objcompresslevel:n { 2 } }
2603         { \__pdf_backend_objcompresslevel:n { 0 } }
2604 }
2605 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2606 {
2607     \tex_global:D
2608 <*luatex>
2609     \tex_pdfvariable:D objcompresslevel
2610 </luatex>
2611 <*pdftex>
2612     \tex_pdfobjcompresslevel:D
2613 </pdftex>
2614     #1 \scan_stop:
2615 }

```

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

__pdf_backend_version_major_gset:n
 __pdf_backend_version_minor_gset:n

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

```

2616 \cs_new_protected:Npe \__pdf_backend_version_major_gset:n #1
2617 {
2618 <*luatex>
2619     \int_compare:nNnT \tex_luatexversion:D > { 106 }
2620     {
2621         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2622         \exp_not:N \int_eval:n {#1} \scan_stop:
2623     }
2624 </luatex>
2625 <*pdftex>
2626     \cs_if_exist:NT \tex_pdfmajorversion:D
2627     {
2628         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2629         \exp_not:N \int_eval:n {#1} \scan_stop:
2630     }
2631 </pdftex>
2632 }
2633 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2634 {
2635     \tex_global:D
2636 <*luatex>
2637     \tex_pdfvariable:D minorversion
2638 </luatex>
2639 <*pdftex>
2640     \tex_pdfminorversion:D
2641 </pdftex>
2642     \int_eval:n {#1} \scan_stop:
2643 }

```

(End of 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: 2644 \cs_new:Npe \__pdf_backend_version_major:
2645 {
2646 <*luatex>
2647   \int_compare:nNnTF \tex_luatexversion:D > { 106 }
2648     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2649     { 1 }
2650 </luatex>
2651 <*pdftex>
2652   \cs_if_exist:NTF \tex_pdfmajorversion:D
2653     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2654     { 1 }
2655 </pdftex>
2656 }
2657 \cs_new:Npn \__pdf_backend_version_minor:
2658 {
2659   \tex_the:D
2660 <*luatex>
2661   \tex_pdfvariable:D minorversion
2662 </luatex>
2663 <*pdftex>
2664   \tex_pdfminorversion:D
2665 </pdftex>
2666 }
```

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

6.2.5 Marked content

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

`__pdf_backend_emc:`

```
2667 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2668 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2669 \cs_new_protected:Npn \__pdf_backend_emc:
2670 { \__kernel_backend_literal_page:n { EMC } }
```

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

```
2671 </luatex | pdftex>
```

6.3 dvipdfmx backend

```
2672 <*dvipdfmx | xetex>
```

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

`__pdf_backend:e`

```
2673 \cs_new_protected:Npe \__pdf_backend:n #1
2674 { \__kernel_backend_literal:n { pdf: #1 } }
2675 \cs_generate_variant:Nn \__pdf_backend:n { e }
```

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

6.3.1 Catalogue entries

```

    \_pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn 2676 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2677 { \__pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2678 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2679 { \__pdf_backend:n { docinfo << /#1 ~ #2 >> } }

```

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

6.3.2 Objects

`\g__pdf_backend_object_prop` For tracking objects to allow finalization.

```
2680 \prop_new:N \g__pdf_backend_object_prop
```

(End of definition for `\g__pdf_backend_object_prop`.)

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

```

\__pdf_backend_object_ref:n 2681 \cs_new_protected:Npn \_pdf_backend_object_new:
\__pdf_backend_object_id:n 2682 { \int_gincr:N \g__pdf_backend_object_int }
2683 \cs_new:Npn \_pdf_backend_object_ref:n #1 { @pdf.obj #1 }
2684 \cs_new_eq:NN \_pdf_backend_object_id:n \_pdf_backend_object_ref:n

```

(End of definition for `_pdf_backend_object_new:`, `_pdf_backend_object_ref:n`, and `_pdf_backend_object_id:n`.)

`_pdf_backend_object_write:nnn` This is where we choose the actual type.

```

    \_pdf_backend_object_write:nnn
    \_pdf_backend_object_write:nne
\__pdf_backend_object_write_array:nn 2685 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
2686 {
\__pdf_backend_object_write_dict:nn 2687   \use:c { \_pdf_backend_object_write_ #2 :nn }
\__pdf_backend_object_write_fstream:nn 2688   { \_pdf_backend_object_ref:n {#1} } {#3}
\__pdf_backend_object_write_stream:nn 2689 }
\__pdf_backend_object_write_stream:nnnn 2690 \cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nne }
\__pdf_backend_object_write_stream:nnnn 2691 \cs_new_protected:Npn \_pdf_backend_object_write_array:nn #1#2
2692 {
2693   \__pdf_backend:e
2694   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2695 }
2696 \cs_new_protected:Npn \_pdf_backend_object_write_dict:nn #1#2
2697 {
2698   \__pdf_backend:e
2699   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2700 }
2701 \cs_new_protected:Npn \_pdf_backend_object_write_fstream:nn #1#2
2702 { \_pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2703 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nn #1#2
2704 { \_pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2705 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nnnn #1#2#3#4
2706 {
2707   \__pdf_backend:e
2708   {
2709     #1 stream ~ #2 ~
2710     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2711   }
2712 }

```

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

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

```

2713 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2714 {
2715   \int_gincr:N \g_pdf_backend_object_int
2716   \exp_args:Nne \use:c { \_pdf_backend_object_write_ #1 :nn }
2717   { @pdf.obj \int_use:N \g_pdf_backend_object_int }
2718   {#2}
2719 }
2720 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { ne }

```

(End of definition for `_pdf_backend_object_now:nn`.)

`_pdf_backend_object_last:`

```

2721 \cs_new:Npn \_pdf_backend_object_last:
2722 { @pdf.obj \int_use:N \g_pdf_backend_object_int }

```

(End of definition for `_pdf_backend_object_last:.`)

`_pdf_backend_pageobject_ref:n` Page references are easy in dvipdfmx/ \X_{TeX} .

```

2723 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2724 { @page #1 }

```

(End of definition for `_pdf_backend_pageobject_ref:n`.)

6.3.3 Destinations

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

```

2725 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2726 {
2727   \_pdf_backend:e
2728   {
2729     dest ~ ( \exp_not:n {#1} )
2730     [
2731       @thispage
2732       \str_case:nnF {#2}
2733       {
2734         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2735         { fit } { /Fit }
2736         { fitb } { /FitB }
2737         { fitbh } { /FitBH }
2738         { fitbv } { /FitBV ~ @xpos }
2739         { fith } { /FitH ~ @ypos }
2740         { fitv } { /FitV ~ @xpos }
2741         { fitr } { /Fit }
2742       }
2743       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2744     ]
2745   }
2746 }

```

```

2747 \cs_new_protected:Npn \__pdf_backend_destination:nmmm #1#2#3#4
2748 {
2749   \exp_args:Ne \__pdf_backend_destination_aux:nmmm
2750   { \dim_eval:n {#2} } {#1} {#3} {#4}
2751 }
2752 \cs_new_protected:Npn \__pdf_backend_destination_aux:nmmm #1#2#3#4
2753 {
2754   \vbox_to_zero:n
2755   {
2756     \__kernel_kern:n {#4}
2757     \hbox:n
2758     {
2759       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2760       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2761     }
2762     \tex_vss:D
2763   }
2764   \__kernel_kern:n {#1}
2765   \vbox_to_zero:n
2766   {
2767     \__kernel_kern:n { -#3 }
2768     \hbox:n
2769     {
2770       \__pdf_backend:n
2771       {
2772         dest ~ (#2)
2773         [
2774           @thispage
2775           /FitR ~
2776           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2777           @xpos ~ @ypos
2778         ]
2779       }
2780     }
2781     \tex_vss:D
2782   }
2783   \__kernel_kern:n { -#1 }
2784 }

```

(End of definition for __pdf_backend_destination:nn, __pdf_backend_destination:nmmm, and __pdf_backend_destination_aux:nmm.)

6.3.4 Structure

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

```

2785 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2786 { \__kernel_backend_literal:e { dvipdfmx:config~z~ \int_eval:n {#1} } }
2787 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2788 {
2789   \bool_if:nF {#1}
2790   { \__kernel_backend_literal:n { dvipdfmx:config-C-0x40 } }
2791 }

```

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


```

\__pdf_backend_version_major_gset:n We start with the assumption that the default is active.
\__pdf_backend_version_minor_gset:n
2792 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2793 {
2794   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2795   \__kernel_backend_literal:e { pdf:majorversion~ \__pdf_backend_version_major: }
2796 }
2797 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2798 {
2799   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2800   \__kernel_backend_literal:e { pdf:minorversion~ \__pdf_backend_version_minor: }
2801 }

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

```

```

\__pdf_backend_version_major: We start with the assumption that the default is active.
\__pdf_backend_version_minor:
2802 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2803 \cs_new:Npn \__pdf_backend_version_minor: { 7 }

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

```

6.3.5 Marked content

```

\__pdf_backend_bdc:nn Simple wrappers. May need refinement: see https://chat.stackexchange.com/transcript/message/49970158#49970158.
\__pdf_backend_emc:
2804 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2805 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2806 \cs_new_protected:Npn \__pdf_backend_emc:
2807 { \__kernel_backend_literal_page:n { EMC } }

(End of definition for \__pdf_backend_bdc:nn and \__pdf_backend_emc:.)
2808 </dviPDFmx|xetex>

```

6.4 dvisvgm backend

```
2809 <*dvisvgm>
```

6.4.1 Destinations

```

\__pdf_backend_destination:nn
\__pdf_backend_destination:nmmn
2810 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2 { }
2811 \cs_new_protected:Npn \__pdf_backend_destination:nmmn #1#2#3#4 { }

(End of definition for \__pdf_backend_destination:nn and \__pdf_backend_destination:nmmn.)

```

6.4.2 Catalogue entries

```

\__pdf_backend_catalog_gput:nn No-op.
\__pdf_backend_info_gput:nn
2812 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2 { }
2813 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2 { }

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

```

6.4.3 Objects

```
\_pdf_backend_object_new: All no-ops here.
\_pdf_backend_object_ref:n 2814 \cs_new_protected:Npn \_pdf_backend_object_new: { }
\_pdf_backend_object_id:n 2815 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
  \_pdf_backend_object_write:nnn 2816 \cs_new:Npn \_pdf_backend_object_id:n #1 { }
  \_pdf_backend_object_write:ne 2817 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3 { }
\_pdf_backend_object_now:nn 2818 \cs_new_protected:Npn \_pdf_backend_object_write:nne #1#2#3 { }
\_pdf_backend_object_now:ne 2819 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
\_pdf_backend_object_last: 2820 \cs_new_protected:Npn \_pdf_backend_object_now:ne #1#2 { }
  \_pdf_backend_pageobject_ref:n 2821 \cs_new:Npn \_pdf_backend_object_last: { }
  2822 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1 { }
```

(End of definition for _pdf_backend_object_new: and others.)

6.4.4 Structure

```
\_pdf_backend_compresslevel:n These are all no-ops.
\_pdf_backend_compress_objects:n 2823 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1 { }
  2824 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1 { }
```

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

```
\_pdf_backend_version_major_gset:n Data not available!
\_pdf_backend_version_minor_gset:n 2825 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
  2826 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1 { }
```

(End of 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: 2827 \cs_new:Npn \_pdf_backend_version_major: { -1 }
  2828 \cs_new:Npn \_pdf_backend_version_minor: { -1 }

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

```
\_pdf_backend_bdc:nn More no-ops.
  \_pdf_backend_emc: 2829 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2 { }
  2830 \cs_new_protected:Npn \_pdf_backend_emc: { }
```

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

```
2831 </dvisvgm>
```

6.5 PDF Page size (media box)

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

```
2832 <*dvipdfmx | dvips>
```

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

```
2833 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
2834 {
2835   \_kernel_backend_first_shipout:n
2836   {
2837     \_kernel_backend_literal:e
2838     {
2839   <*dvi $\text{pdfmx}$ >
2840       pdf:pagesize ~
2841         width ~ \dim_eval:n {#1} ~
2842         height ~ \dim_eval:n {#2}
2843   </dvi $\text{pdfmx}$ >
2844   <*dvips>
2845       papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
2846   </dvips>
2847     }
2848   }
2849 }
```

(End of definition for `_pdf_backend_pagesize_gset:nn`.)

```
2850 </dvi $\text{pdfmx}$  | dvips>
2851 <*luatex | pdftex | xetex>
```

`_pdf_backend_pagesize_gset:nn` Pass to the primitives.

```
2852 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
2853 {
2854   \dim_gset:Nn \tex_pagewidth:D {#1}
2855   \dim_gset:Nn \tex_pageheight:D {#2}
2856 }
```

(End of definition for `_pdf_backend_pagesize_gset:nn`.)

```
2857 </luatex | pdftex | xetex>
2858 <*dvisvgm>
```

`_pdf_backend_pagesize_gset:nn` A no-op.

```
2859 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2 { }
```

(End of definition for `_pdf_backend_pagesize_gset:nn`.)

```
2860 </dvisvgm>
2861 </package>
```

7 I3backend-pdfannot implementation

```
2862 <*package>
2863 <@@=pdfannot>
```

7.1 dvips backend

```
2864 <*dvips>
```

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

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

`\l_pdfannot_backend_content_box` The content of an annotation.
 2865 `\box_new:N \l_pdfannot_backend_content_box`
 (End of definition for `\l_pdfannot_backend_content_box`.)

`\l_pdfannot_backend_model_box` For creating model sizing for links.
 2866 `\box_new:N \l_pdfannot_backend_model_box`
 (End of definition for `\l_pdfannot_backend_model_box`.)

`\g_pdfannot_backend_int` Needed to track annotations.
 2867 `\int_new:N \g_pdfannot_backend_int`
 (End of definition for `\g_pdfannot_backend_int`.)

`_pdfannot_backend_generic:nmnn` Annotations are objects but they are not in the object data lists. Here, to get the
`_pdfannot_backend_generic_aux:nmnn` coordinates 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_EX₂_ε picture of zero size). Once the data is collected, use it to set up the annotation border.

```

2868 \cs_new_protected:Npn \_pdfannot_backend_generic:nmnn #1#2#3#4
2869 {
2870   \exp_args:Nf \_pdfannot_backend_generic_aux:nmnn
2871     { \dim_eval:n {#1} } {#2} {#3} {#4}
2872 }
2873 \cs_new_protected:Npn \_pdfannot_backend_generic_aux:nmnn #1#2#3#4
2874 {
2875   \box_move_down:nn {#3}
2876   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2877   \box_move_up:nn {#2}
2878   {
2879     \hbox:n
2880     {
2881       \_kernel_kern:n {#1}
2882       \_kernel_backend_postscript:n { pdf.save.ur }
2883       \_kernel_kern:n { -#1 }
2884     }
2885   }
2886   \int_gincr:N \g_pdfannot_backend_int
2887   \_kernel_backend_postscript:e
2888   {
2889     mark
2890     /_objdef { pdf.annot \int_use:N \g_pdfannot_backend_int }
2891     pdf.rect
2892     #4 ~
2893     /ANN ~
2894     pdfmark
2895   }
2896 }

```

(End of definition for `_pdfannot_backend_generic:nmnn` and `_pdfannot_backend_generic_aux:nmnn`.)

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

```
2897 \cs_new:Npn \_pdfannot_backend_last:  
2898 { { pdf.annot \int_use:N \g_pdfannot_backend_int } }
```

(End of definition for `_pdfannot_backend_last:.`)

`\g_pdfannot_backend_link_int` To track annotations which are links.

```
2899 \int_new:N \g_pdfannot_backend_link_int
```

(End of definition for `\g_pdfannot_backend_link_int.`)

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

```
2900 \tl_new:N \g_pdfannot_backend_link_dict_tl
```

(End of definition for `\g_pdfannot_backend_link_dict_tl.`)

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

```
2901 \int_new:N \g_pdfannot_backend_link_sf_int
```

(End of definition for `\g_pdfannot_backend_link_sf_int.`)

`\g_pdfannot_backend_link_math_bool` Needed to save/restore math mode.

```
2902 \bool_new:N \g_pdfannot_backend_link_math_bool
```

(End of definition for `\g_pdfannot_backend_link_math_bool.`)

`\g_pdfannot_backend_link_bool` Track link formation: we cannot nest at all.

```
2903 \bool_new:N \g_pdfannot_backend_link_bool
```

(End of definition for `\g_pdfannot_backend_link_bool.`)

`\l_pdfannot_backend_breaklink_pdfmark_tl` Swappable content for link breaking.

```
2904 \tl_new:N \l_pdfannot_backend_breaklink_pdfmark_tl  
2905 \tl_set:Nn \l_pdfannot_backend_breaklink_pdfmark_tl { pdfmark }
```

(End of definition for `\l_pdfannot_backend_breaklink_pdfmark_tl.`)

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

```
2906 \cs_new_protected:Npn \_pdfannot_backend_breaklink_postscript:n #1 { }
```

(End of definition for `_pdfannot_backend_breaklink_postscript:n.`)

`_pdfannot_backend_breaklink_usebox:N` Swappable box unpacking or use.

```
2907 \cs_new_eq:NN \_pdfannot_backend_breaklink_usebox:N \box_use:N
```

(End of definition for `_pdfannot_backend_breaklink_usebox:N.`)

```

\_pdfannot_backend_link_begin_goto:nnw
\_pdfannot_backend_link_begin_user:nnw
\_pdfannot_backend_link:nw
  \_pdfannot_backend_link_aux:nw
  \_pdfannot_backend_link_end:
  \_pdfannot_backend_link_end_aux:
  \_pdfannot_backend_link_minima:
\_pdfannot_backend_link_outerbox:n
  \_pdfannot_backend_link_sf_save:
\_pdfannot_backend_link_sf_restore:

```

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

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

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

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

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

```

2908 \cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nnw #1#2
2909 {
2910   \_pdfannot_backend_link_begin:nw
2911   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2912 }
2913 \cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nnw #1#2
2914 { \_pdfannot_backend_link_begin:nw {#1#2} }
2915 \cs_new_protected:Npn \_pdfannot_backend_link_begin:nw #1
2916 {
2917   \bool_if:NF \g__pdfannot_backend_link_bool
2918   { \_pdfannot_backend_link_begin_aux:nw {#1} }
2919 }

```

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.

```

2920 \cs_new_protected:Npn \_pdfannot_backend_link_begin_aux:nw #1
2921 {
2922   \bool_gset_true:N \g__pdfannot_backend_link_bool
2923   \_kernel_backend_postscript:n
2924   { /pdf.link.dict ( #1 ) def }
2925   \tl_gset:Nn \g__pdfannot_backend_link_dict_tl {#1}
2926   \_pdfannot_backend_link_sf_save:
2927   \mode_if_math:TF
2928   { \bool_gset_true:N \g__pdfannot_backend_link_math_bool }
2929   { \bool_gset_false:N \g__pdfannot_backend_link_math_bool }
2930   \hbox_set:Nw \l__pdfannot_backend_content_box
2931   \_pdfannot_backend_link_sf_restore:
2932   \bool_if:NT \g__pdfannot_backend_link_math_bool
2933   { \c_math_toggle_token }
2934 }
2935 \cs_new_protected:Npn \_pdfannot_backend_link_end:
2936 {
2937   \bool_if:NT \g__pdfannot_backend_link_bool
2938   { \_pdfannot_backend_link_end_aux: }
2939 }
2940 \cs_new_protected:Npn \_pdfannot_backend_link_end_aux:

```

```

2941 {
2942   \bool_if:NT \g__pdfannot_backend_link_math_bool
2943     { \c_math_toggle_token }
2944   \__pdfannot_backend_link_sf_save:
2945   \hbox_set_end:
2946   \__pdfannot_backend_link_minima:
2947   \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
2948   \exp_args:Ne \__pdfannot_backend_link_outerbox:n
2949     {
2950       \int_if_odd:nTF { \value { page } }
2951         { \oddsidemargin }
2952         { \evensidemargin }
2953     }
2954   \box_move_down:nn { \box_dp:N \l__pdfannot_backend_content_box }
2955     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2956   \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.begin }
2957   \__pdfannot_backend_breaklink_usebox:N \l__pdfannot_backend_content_box
2958   \__pdfannot_backend_breaklink_postscript:n { pdf.bordertracking.end }
2959   \box_move_up:nn { \box_ht:N \l__pdfannot_backend_content_box }
2960     {
2961       \hbox:n
2962         { \__kernel_backend_postscript:n { pdf.save.linkur } }
2963     }
2964   \int_gincr:N \g__pdfannot_backend_int
2965   \int_gset_eq:NN \g__pdfannot_backend_link_int \g__pdfannot_backend_int
2966   \__kernel_backend_postscript:e
2967     {
2968       mark
2969       /_objdef { pdf.annot \int_use:N \g__pdfannot_backend_link_int }
2970       \g__pdfannot_backend_link_dict_tl \c_space_tl
2971       pdf.rect
2972       /ANN ~ \l__pdfannot_backend_breaklink_pdfmark_tl
2973     }
2974   \__pdfannot_backend_link_sf_restore:
2975   \bool_gset_false:N \g__pdfannot_backend_link_bool
2976 }
2977 \cs_new_protected:Npn \__pdfannot_backend_link_minima:
2978 {
2979   \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
2980   \__kernel_backend_postscript:e
2981     {
2982       /pdf.linkdp.pad ~
2983       \dim_to_decimal:n
2984         {
2985           \dim_max:nn
2986             {
2987               \box_dp:N \l__pdfannot_backend_model_box
2988               - \box_dp:N \l__pdfannot_backend_content_box
2989             }
2990           { Opt }
2991         } ~
2992       pdf.pt.dvi ~ def
2993       /pdf.linkht.pad ~
2994       \dim_to_decimal:n

```

```

2995         {
2996           \dim_max:nn
2997           {
2998             \box_ht:N \l__pdfannot_backend_model_box
2999             - \box_ht:N \l__pdfannot_backend_content_box
3000           }
3001           { Opt }
3002         } ~
3003         pdf.pt.dvi ~ def
3004     }
3005 }
3006 \cs_new_protected:Npn \__pdfannot_backend_link_outerbox:n #1
3007 {
3008   \__kernel_backend_postscript:e
3009   {
3010     /pdf.outerbox
3011     [
3012       \dim_to_decimal:n {#1} ~
3013       \dim_to_decimal:n { -\box_dp:N \l__pdfannot_backend_model_box } ~
3014       \dim_to_decimal:n { #1 + \textwidth } ~
3015       \dim_to_decimal:n { \box_ht:N \l__pdfannot_backend_model_box }
3016     ]
3017     [ exch { pdf.pt.dvi } forall ] def
3018     /pdf.baselineskip ~
3019     \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
3020     { pdf.pt.dvi ~ def }
3021     { pop ~ pop }
3022     ifelse
3023   }
3024 }
3025 \cs_new_protected:Npn \__pdfannot_backend_link_sf_save:
3026 {
3027   \int_gset:Nn \g__pdfannot_backend_link_sf_int
3028   {
3029     \mode_if_horizontal:TF
3030     { \tex_spacefactor:D }
3031     { 0 }
3032   }
3033 }
3034 \cs_new_protected:Npn \__pdfannot_backend_link_sf_restore:
3035 {
3036   \mode_if_horizontal:T
3037   {
3038     \int_compare:nNnT \g__pdfannot_backend_link_sf_int > { 0 }
3039     { \int_set:Nn \tex_spacefactor:D \g__pdfannot_backend_link_sf_int }
3040   }
3041 }

```

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

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

```

3042 \use_none:nnn
3043 \cs_if_exist:NT \hook_gput_code:nnn
3044 {

```



```

3045 \hook_gput_code:nnn { build/column/after } { backend }
3046 {
3047   \box_if_empty:NF \l_shipout_box
3048   {
3049     \vbox_set:Nn \l_shipout_box
3050     {
3051       \__kernel_backend_postscript:n
3052       {
3053         pdf.globaldict /pdf.brokenlink.rect ~ known
3054         { pdf.bordertracking.continue }
3055         if
3056       }
3057       \vbox_unpack_drop:N \l_shipout_box
3058       \__kernel_backend_postscript:n
3059       { pdf.bordertracking.endpage }
3060     }
3061   }
3062 }
3063 \tl_set:Nn \l__pdfannot_backend_breaklink_pdfmark_tl { pdf.pdfmark }
3064 \cs_set_eq:NN \__pdfannot_backend_breaklink_postscript:n
3065 \__kernel_backend_postscript:n
3066 \cs_set_eq:NN \__pdfannot_backend_breaklink_usebox:N \hbox_unpack:N
3067 }

```

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

```

3068 \cs_new:Npn \__pdfannot_backend_link_last:
3069 { { pdf.annot \int_use:N \g__pdfannot_backend_link_int } }

```

(End of definition for `__pdfannot_backend_link_last:.`)

`_pdfannot_backend_link_margin:n` Convert to big points and pass to PostScript.

```

3070 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1
3071 {
3072   \__kernel_backend_postscript:e
3073   {
3074     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
3075   }
3076 }

```

(End of definition for `_pdfannot_backend_link_margin:n.`)

`_pdfannot_backend_link_on:`

```

\__pdfannot_backend_link_off:
3077 \cs_new_protected:Npn \__pdfannot_backend_link_on: { }
3078 \cs_new_protected:Npn \__pdfannot_backend_link_off: { }

```

(End of definition for `_pdfannot_backend_link_on: and _pdfannot_backend_link_off:.`)

```

3079 </dvips>

```

7.2 LuaTeX and pdfTeX backend

3080 `<*luatex | pdftex>`

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

```

3081 \cs_new_protected:Npn \_pdfannot_backend_generic:nnnn #1#2#3#4
3082 {
3083 <*luatex>
3084   \tex_pdfextension:D annot ~
3085 </luatex>
3086 <*pdftex>
3087   \tex_pdfannot:D
3088 </pdftex>
3089   width ~ \dim_eval:n {#1} ~
3090   height ~ \dim_eval:n {#2} ~
3091   depth ~ \dim_eval:n {#3} ~
3092   {#4}
3093 }

```

(End of definition for `_pdfannot_backend_generic:nnnn`.)

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

```

3094 \cs_new:Npe \_pdfannot_backend_last:
3095 {
3096   \exp_not:N \int_value:w
3097 <*luatex>
3098   \exp_not:N \tex_pdffeedback:D lastannot ~
3099 </luatex>
3100 <*pdftex>
3101   \exp_not:N \tex_pdflastannot:D
3102 </pdftex>
3103   \c_space_tl 0 ~ R
3104 }

```

(End of definition for `_pdfannot_backend_last:`.)

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

```

\_pdfannot_backend_link_begin_user:nnw
\_pdfannot_backend_link_begin:nnnw
\_pdfannot_backend_link_end:
3105 \cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nnw #1#2
3106 { \_pdfannot_backend_link_begin:nnnw {#1} { goto~name } {#2} }
3107 \cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nnw #1#2
3108 { \_pdfannot_backend_link_begin:nnnw {#1} { user } {#2} }
3109 \cs_new_protected:Npn \_pdfannot_backend_link_begin:nnnw #1#2#3
3110 {
3111 <*luatex>
3112   \tex_pdfextension:D startlink ~
3113 </luatex>
3114 <*pdftex>
3115   \tex_pdfstartlink:D
3116 </pdftex>
3117   attr {#1}
3118   #2 {#3}
3119 }
3120 \cs_new_protected:Npn \_pdfannot_backend_link_end:

```

```

3121 {
3122 <*luatex>
3123   \tex_pdfextension:D endlink \scan_stop:
3124 </luatex>
3125 <*pdftex>
3126   \tex_pdfendlink:D
3127 </pdftex>
3128 }

```

(End of definition for _pdfannot_backend_link_begin_goto:nw and others.)

_pdfannot_backend_link_last: Formatted for direct use.

```

3129 \cs_new:Npe \_pdfannot_backend_link_last:
3130 {
3131   \exp_not:N \int_value:w
3132 <*luatex>
3133   \exp_not:N \tex_pdffeedback:D lastlink ~
3134 </luatex>
3135 <*pdftex>
3136   \exp_not:N \tex_pdflastlink:D
3137 </pdftex>
3138   \c_space_tl 0 ~ R
3139 }

```

(End of definition for _pdfannot_backend_link_last:.)

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

```

3140 \cs_new_protected:Npn \_pdfannot_backend_link_margin:n #1
3141 {
3142 <*luatex>
3143   \tex_pdfvariable:D linkmargin
3144 </luatex>
3145 <*pdftex>
3146   \tex_pdflinkmargin:D
3147 </pdftex>
3148   \dim_eval:n {#1} \scan_stop:
3149 }

```

(End of definition for _pdfannot_backend_link_margin:n.)

_pdfannot_backend_link_on: Separate definitions for the two engines.

```

\_pdfannot_backend_link_off:
3150 \cs_new_protected:Npn \_pdfannot_backend_link_on:
3151 <*luatex>
3152   { \tex_pdfextension:D linkstate 0 ~ }
3153 </luatex>
3154 <*pdftex>
3155   { \tex_pdfrunninglinkon:D }
3156 </pdftex>
3157 \cs_new_protected:Npn \_pdfannot_backend_link_off:
3158 <*luatex>
3159   { \tex_pdfextension:D linkstate 1 ~ }
3160 </luatex>
3161 <*pdftex>
3162   { \tex_pdfrunninglinkoff:D }
3163 </pdftex>

```

(End of definition for `_pdfannot_backend_link_on:` and `_pdfannot_backend_link_off:`.)

3164 `</!latex | pdftex>`

7.3 dvipdfmx backend

3165 `<*dvipdfmx | xetex>`

`_pdfannot_backend:n` A generic function for the backend PDF specials

```
3166 \cs_new_protected:Npe \_pdfannot_backend:n #1
3167 { \_kernel_backend_literal:n { pdf: #1 } }
3168 \cs_generate_variant:Nn \_pdfannot_backend:n { e }
```

(End of definition for `_pdfannot_backend:n`.)

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

```
3169 \int_new:N \g_pdfannot_backend_int
```

(End of definition for `\g_pdfannot_backend_int`.)

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

```
3170 \cs_new_protected:Npn \_pdfannot_backend_generic:nnnn #1#2#3#4
3171 {
3172   \int_gincr:N \g_pdfannot_backend_int
3173   \_pdfannot_backend:e
3174   {
3175     ann ~ @pdfannot \int_use:N \g_pdfannot_backend_int \c_space_tl
3176     width ~ \dim_eval:n {#1} ~
3177     height ~ \dim_eval:n {#2} ~
3178     depth ~ \dim_eval:n {#3} ~
3179     << /Type /Annot #4 >>
3180   }
3181 }
```

(End of definition for `_pdfannot_backend_generic:nnnn`.)

`_pdfannot_backend_last:`

```
3182 \cs_new:Npn \_pdfannot_backend_last:
3183 { @pdfannot \int_use:N \g_pdfannot_backend_int }
```

(End of definition for `_pdfannot_backend_last:`.)

`\g_pdfannot_backend_link_int` To track annotations which are links.

```
3184 \int_new:N \g_pdfannot_backend_link_int
```

(End of definition for `\g_pdfannot_backend_link_int`.)

`_pdfannot_backend_link_begin_goto:nnw` All created using the same internals.

```
\_pdfannot_backend_link_begin_user:nnw 3185 \cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nnw #1#2
\_pdfannot_backend_link_begin:n 3186 {
\_pdfannot_backend_link_end: 3187   \_pdfannot_backend_link_begin:n
3188   { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> }
3189 }
3190 \cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nnw #1#2
3191 { \_pdfannot_backend_link_begin:n {#1#2} }
3192 \cs_new_protected:Npe \_pdfannot_backend_link_begin:n #1
```

```

3193 {
3194   \int_gincr:N \exp_not:N \g__pdfannot_backend_int
3195   \int_gset_eq:NN \exp_not:N \g__pdfannot_backend_link_int
3196   \exp_not:N \g__pdfannot_backend_int
3197   \__pdfannot_backend:e
3198   {
3199     bann ~
3200     @pdfannot
3201     \exp_not:N \int_use:N \exp_not:N \g__pdfannot_backend_link_int
3202     \c_space_tl
3203     <<
3204     /Type /Annot
3205     #1
3206     >>
3207   }
3208 }
3209 \cs_new_protected:Npn \__pdfannot_backend_link_end:
3210 { \__pdfannot_backend:n { eann } }

```

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

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

```

3211 \cs_new:Npn \__pdfannot_backend_link_last:
3212 { @pdfannot \int_use:N \g__pdfannot_backend_link_int }

```

(End of definition for __pdfannot_backend_link_last:.)

_pdfannot_backend_link_margin:n Pass to dvipdfmx.

```

3213 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1
3214 { \__kernel_backend_literal:e { dvipdfmx:config~ \dim_eval:n {#1} } }

```

(End of definition for __pdfannot_backend_link_margin:n.)

_pdfannot_backend_link_on:

```

\__pdfannot_backend_link_off: 3215 \cs_new_protected:Npn \__pdfannot_backend_link_on: { \__pdfannot_backend:n { link } }
3216 \cs_new_protected:Npn \__pdfannot_backend_link_off: { \__pdfannot_backend:n { nolink } }

```

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

```

3217 </dvipdfmx | xetex>

```

7.4 dvisvgm backend

```

3218 <*dvisvgm>

```

_pdfannot_backend_generic:nnnn

```

3219 \cs_new_protected:Npn \__pdfannot_backend_generic:nnnn #1#2#3#4 { }

```

(End of definition for __pdfannot_backend_generic:nnnn.)

_pdfannot_backend_last:

```

3220 \cs_new:Npn \__pdfannot_backend_last: { }

```

(End of definition for __pdfannot_backend_last:.)

```

\_pdfannot_backend_link_begin_goto:nnw
\_pdfannot_backend_link_begin_user:nnw 3221 \cs_new_protected:Npn \__pdfannot_backend_link_begin_goto:nnw #1#2 { }
\_pdfannot_backend_link_begin:nnnw 3222 \cs_new_protected:Npn \__pdfannot_backend_link_begin_user:nnw #1#2 { }
\_pdfannot_backend_link_end: 3223 \cs_new_protected:Npn \__pdfannot_backend_link_begin:nnnw #1#2#3 { }
3224 \cs_new_protected:Npn \__pdfannot_backend_link_end: { }

```

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

```

\_pdfannot_backend_link_last:
3225 \cs_new:Npe \__pdfannot_backend_link_last: { }

```

(End of definition for __pdfannot_backend_link_last:.)

```

\_pdfannot_backend_link_margin:n
3226 \cs_new_protected:Npn \__pdfannot_backend_link_margin:n #1 { }

```

(End of definition for __pdfannot_backend_link_margin:n.)

```

\_pdfannot_backend_link_on: For handling places like headers.
\_pdfannot_backend_link_off: 3227 \cs_new_protected:Npn \__pdfannot_backend_link_on: { }
3228 \cs_new_protected:Npn \__pdfannot_backend_link_off: { }

```

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

```

3229 </dvisvgm>

```

7.5 Transitional code

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

```

3230 <@@=)
3231 \cs_new_eq:NN \_pdf_backend_annotation:nnnn \_pdfannot_backend_generic:nnnn
3232 \cs_new_eq:NN \_pdf_backend_annotation_last: \_pdfannot_backend_last:
3233 \clist_map_inline:nn
3234 {
3235   begin_goto:nnw ,
3236   begin_user:nnw ,
3237   begin:nnnw ,
3238   end: ,
3239   last: ,
3240   margin:n
3241 }
3242 { \cs_new_eq:cc { \_pdf_backend_link_ #1 } { \_pdfannot_backend_link_ #1 } }
3243 </package>

```

8 I3backend-opacity implementation

```
3244 (*package)
3245 (@@=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.

```
3246 (*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.

```

3247 \cs_new_protected:Npn \__opacity_backend_select:n #1
3248 {
3249   \__opacity_backend:nnn {#1} { fill } { ca }
3250   \__opacity_backend:nnn {#1} { stroke } { CA }
3251   \group_insert_after:N \__opacity_backend_reset_fill:
3252   \group_insert_after:N \__opacity_backend_reset_stroke:
3253 }
3254 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3255 {
3256   \__opacity_backend:nnn
3257   { #1 }
3258   { fill }
3259   { ca }
3260   \group_insert_after:N \__opacity_backend_reset_fill:
3261 }
3262 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3263 {
3264   \__opacity_backend:nnn
3265   { #1 }
3266   { stroke }
3267   { CA }
3268   \group_insert_after:N \__opacity_backend_reset_stroke:
3269 }
3270 \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3271 {
3272   \__kernel_backend_postscript:n
3273   {
3274     product ~ (Ghostscript) ~ search
3275     {
3276       pop ~ pop ~ pop ~
3277       #1 ~ .set #2 constantalpha
3278     }
3279     {
3280       pop ~
3281       mark ~
3282       /#3 ~ #1
3283       /SetTransparency ~

```

```

3284         pdfmark
3285     }
3286     ifelse
3287 }
3288 }
3289 \cs_new_protected:Npn \__opacity_backend_reset_fill:
3290 {
3291     \__opacity_backend:nnn
3292     { 1 }
3293     { fill }
3294     { ca }
3295 }
3296 \cs_new_protected:Npn \__opacity_backend_reset_stroke:
3297 {
3298     \__opacity_backend:nnn
3299     { 1 }
3300     { stroke }
3301     { CA }
3302 }

```

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

```

3303 </dvips>
3304 <*dvipdfmx | luatex | pdftex | xetex>

```

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

```

3305 \bool_lazy_and:nnT
3306 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3307 { \pdfmanagement_if_active_p: }
3308 {
3309 <*luatex | pdftex>
3310     \__kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3311     { page ~ direct } { /opacity 1 ~ gs }
3312 </luatex | pdftex>
3313     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3314     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3315 }

```

(End of definition for \c__opacity_backend_stack_int.)

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

```

3316 \tl_new:N \l__opacity_backend_fill_tl
3317 \tl_new:N \l__opacity_backend_stroke_tl
3318 \tl_set:Nn \l__opacity_backend_fill_tl { 1 }
3319 \tl_set:Nn \l__opacity_backend_stroke_tl { 1 }

```

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

__opacity_backend_select:n Much the same as color.

```

\__opacity_backend_reset:
3320 \cs_new_protected:Npn \__opacity_backend_select:n #1
3321 {
3322     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3323     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3324     \pdfmanagement_add:nnn { Page / Resources / ExtGState }

```



```

3325     { opacity #1 }
3326     { << /ca ~ #1 /CA ~ #1 >> }
3327 <*dviptfm | xetex>
3328     \__kernel_backend_literal_pdf:n
3329 </dviptfm | xetex>
3330 <*luatex | pdftex>
3331     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3332 </luatex | pdftex>
3333     { /opacity #1 ~ gs }
3334     \group_insert_after:N \__opacity_backend_reset:
3335 }
3336 \cs_new_protected:Npn \__opacity_backend_reset:
3337 {
3338 <*dviptfm | xetex>
3339     \__kernel_backend_literal_pdf:n
3340     { /opacity1 ~ gs }
3341 </dviptfm | xetex>
3342 <*luatex | pdftex>
3343     \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int
3344 </luatex | pdftex>
3345 }

```

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

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

```

\__opacity_backend_fill_stroke:nn
3346 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3347 {
3348     \exp_args:Nno \__opacity_backend_fill_stroke:nn
3349     { #1 }
3350     { \l__opacity_backend_stroke_tl }
3351 }
3352 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3353 {
3354     \exp_args:No \__opacity_backend_fill_stroke:nn
3355     { \l__opacity_backend_fill_tl }
3356     { #1 }
3357 }
3358 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3359 {
3360     \str_if_eq:nnTF {#1} {#2}
3361     { \__opacity_backend_select:n {#1} }
3362     {
3363         \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3364         \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3365         \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3366         { opacity.fill #1 }
3367         { << /ca ~ #1 >> }
3368         \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3369         { opacity.stroke #2 }
3370         { << /CA ~ #2 >> }
3371 <*dviptfm | xetex>
3372     \__kernel_backend_literal_pdf:n
3373 </dviptfm | xetex>

```

```

3374 <*luatex | pdftex>
3375     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3376 </luatex | pdftex>
3377     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3378     \group_insert_after:N \__opacity_backend_reset:
3379   }
3380 }

```

(End of definition for `__opacity_backend_fill:n`, `__opacity_backend_stroke:n`, and `__opacity_backend_fill_stroke:nn`.)

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

```

3381 \bool_lazy_and:nnF
3382   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3383   { \pdfmanagement_if_active_p: }
3384   {
3385     \cs_gset_protected:Npn \__opacity_backend_select:n #1 { }
3386     \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2 { }
3387   }

```

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

```

3388 </dviptfm | luatex | pdftex | xetex>
3389 <*dvisvgm>

```

`__opacity_backend_select:n` `__opacity_backend_fill:n` `__opacity_backend_stroke:n` `__opacity_backend:nn` 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.

```

3390 \cs_new_protected:Npn \__opacity_backend_select:n #1
3391   { \__opacity_backend:nn {#1} { } }
3392 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3393   { \__opacity_backend:nn {#1} { fill- } }
3394 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3395   { \__opacity_backend:nn {#1} { stroke- } }
3396 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3397   { \__kernel_backend_scope:e { #2 opacity = " #1 " } }

```

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

```

3398 </dvisvgm>
3399 </package>

```

8.1 Font handling integration

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

```

3400 <*lua>
      First we need to check if pdfmanagement is active from Lua.
3401 local pdfmanagement_active do
3402   local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3403   local cmd = pdfmanagement_if_active_p.cmdname
3404   if cmd == 'undefined_cs' then
3405     pdfmanagement_active = false
3406   else
3407     token.put_next(pdfmanagement_if_active_p)

```

```

3408     pdfmanagement_active = token.scan_int() ~= 0
3409   end
3410 end
3411
3412 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3413   luaotfload.set_transparent_colorstack(function() return token.create'c__opacity_backend_st
3414
3415   local transparent_register = {
3416     token.create'pdfmanagement_add:nm',
3417     token.new(0, 1),
3418     'Page/Resources/ExtGState',
3419     token.new(0, 2),
3420     token.new(0, 1),
3421     '',
3422     token.new(0, 2),
3423     token.new(0, 1),
3424     '<</ca ',
3425     '',
3426     '/CA ',
3427     '',
3428     '>>',
3429     token.new(0, 2),
3430   }
3431   luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)
3432     value = (octet * -1):match(value)
3433     if not value then
3434       tex.error'Invalid transparency value'
3435       return
3436     end
3437     value = value:sub(1, -2)
3438     local result = 'opacity' .. value
3439     tex.runtoks(function()
3440       transparent_register[6], transparent_register[10], transparent_register[12] = result,
3441       tex.sprint(-2, transparent_register)
3442     end)
3443     return '/' .. result .. ' gs'
3444   end, 'l3opacity')
3445 end
3446 </lua>

```

9 l3backend-header implementation

```

3447 <*dvips & header>

```

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

```

3448 /color.sc { } def

```

(End of definition for color.sc.)

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

```

3449 TeXDict begin
3450 /TeXcolorseparation { setcolor } def
3451 end

```

(End of definition for TeXcolorseparation and separation.)

pdf.globaldict A small global dictionary for backend use.

```
3452 true setglobal
3453 /pdf.globaldict 4 dict def
3454 false setglobal
```

(End of definition for pdf.globaldict.)

pdf.cvs 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.

```
pdf.dvi.pt
pdf.pt.dvi
pdf.rect.ht
3455 /pdf.cvs { 65534 string cvs } def
3456 /pdf.dvi.pt { 72.27 mul Resolution div } def
3457 /pdf.pt.dvi { 72.27 div Resolution mul } def
3458 /pdf.rect.ht { dup 1 get neg exch 3 get add } def
```

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

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

```
pdf.linkdp.pad
pdf.linkht.pad
3459 /pdf.linkmargin { 1 pdf.pt.dvi } def
3460 /pdf.linkdp.pad { 0 } def
3461 /pdf.linkht.pad { 0 } def
```

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

pdf.rect 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.

```
pdf.save.ll
pdf.save.ur
pdf.save.linkll
pdf.save.linkur
pdf.llx
pdf.lly
pdf.urx
pdf.ury
3462 /pdf.rect
3463 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
3464 /pdf.save.ll
3465 {
3466     currentpoint
3467     /pdf.lly exch def
3468     /pdf.llx exch def
3469 }
3470 def
3471 /pdf.save.ur
3472 {
3473     currentpoint
3474     /pdf.ury exch def
3475     /pdf.urx exch def
3476 }
3477 def
3478 /pdf.save.linkll
3479 {
3480     currentpoint
3481     pdf.linkmargin add
3482     pdf.linkdp.pad add
3483     /pdf.lly exch def
3484     pdf.linkmargin sub
3485     /pdf.llx exch def
3486 }
```

```

3487     def
3488 /pdf.save.linkur
3489   {
3490     currentpoint
3491     pdf.linkmargin sub
3492     pdf.linkht.pad sub
3493     /pdf.ury exch def
3494     pdf.linkmargin add
3495     /pdf.urx exch def
3496   }
3497   def

```

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

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

```

pdf.dev.x 3498 /pdf.dest.anchor
pdf.dev.y 3499   {
pdf.tmpa 3500     currentpoint exch
pdf.tmpb 3501     pdf.dvi.pt 72 add
pdf.tmpc 3502     /pdf.dest.x exch def
pdf.tmpd 3503     pdf.dvi.pt
3504     vsize 72 sub exch sub
3505     /pdf.dest.y exch def
3506   }
3507   def
3508 /pdf.dest.point
3509   { pdf.dest.x pdf.dest.y } def
3510 /pdf.dest2device
3511   {
3512     /pdf.dest.y exch def
3513     /pdf.dest.x exch def
3514     matrix currentmatrix
3515     matrix defaultmatrix
3516     matrix invertmatrix
3517     matrix concatmatrix
3518     cvx exec
3519     /pdf.dev.y exch def
3520     /pdf.dev.x exch def
3521     /pdf.tmpd exch def
3522     /pdf.tmpc exch def
3523     /pdf.tmpb exch def
3524     /pdf.tmpa exch def
3525     pdf.dest.x pdf.tmpa mul
3526     pdf.dest.y pdf.tmpc mul add
3527     pdf.dev.x add
3528     pdf.dest.x pdf.tmpb mul
3529     pdf.dest.y pdf.tmpd mul add
3530     pdf.dev.y add
3531   }
3532   def

```

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

```
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 3533 /pdf.bordertracking false def
pdf.brokenlink.skip 3534 /pdf.bordertracking.begin
pdf.brokenlink.dict 3535 {
pdf.bordertracking.endpage 3536 SDict /pdf.bordertracking true put
pdf.bordertracking.continue 3537 SDict /pdf.leftboundary undef
pdf.originx 3538 SDict /pdf.rightboundary undef
pdf.originy 3539 /a where
3540 {
3541 /a
3542 {
3543 currentpoint pop
3544 SDict /pdf.rightboundary known dup
3545 {
3546 SDict /pdf.rightboundary get 2 index lt
3547 { not }
3548 if
3549 }
3550 if
3551 { pop }
3552 { SDict exch /pdf.rightboundary exch put }
3553 ifelse
3554 moveto
3555 currentpoint pop
3556 SDict /pdf.leftboundary known dup
3557 {
3558 SDict /pdf.leftboundary get 2 index gt
3559 { not }
3560 if
3561 }
3562 if
3563 { pop }
3564 { SDict exch /pdf.leftboundary exch put }
3565 ifelse
3566 }
3567 put
3568 }
3569 if
3570 }
3571 def
3572 /pdf.bordertracking.end
3573 {
3574 /a where { /a { moveto } put } if
3575 /x where { /x { 0 exch rmoveto } put } if
3576 SDict /pdf.leftboundary known
3577 { pdf.outerbox 0 pdf.leftboundary put }
3578 if
3579 SDict /pdf.rightboundary known
3580 { pdf.outerbox 2 pdf.rightboundary put }
```

```

3581     if
3582     SDict /pdf.bordertracking false put
3583   }
3584   def
3585   /pdf.bordertracking.endpage
3586   {
3587   pdf.bordertracking
3588     {
3589     pdf.bordertracking.end
3590     true setglobal
3591     pdf.globaldict
3592     /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3593     pdf.globaldict
3594     /pdf.brokenlink.skip pdf.baselineskip put
3595     pdf.globaldict
3596     /pdf.brokenlink.dict
3597     pdf.link.dict pdf.cvs put
3598     false setglobal
3599     mark pdf.link.dict cvx exec /Rect
3600     [
3601     pdf.llx
3602     pdf.lly
3603     pdf.outerbox 2 get pdf.linkmargin add
3604     currentpoint exch pop
3605     pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3606     ]
3607     /ANN pdf.pdfmark
3608   }
3609   if
3610   }
3611   def
3612   /pdf.bordertracking.continue
3613   {
3614     /pdf.link.dict pdf.globaldict
3615     /pdf.brokenlink.dict get def
3616     /pdf.outerbox pdf.globaldict
3617     /pdf.brokenlink.rect get def
3618     /pdf.baselineskip pdf.globaldict
3619     /pdf.brokenlink.skip get def
3620     pdf.globaldict dup dup
3621     /pdf.brokenlink.dict undef
3622     /pdf.brokenlink.skip undef
3623     /pdf.brokenlink.rect undef
3624     currentpoint
3625     /pdf.originy exch def
3626     /pdf.originx exch def
3627     /a where
3628     {
3629     /a
3630     {
3631     moveto
3632     SDict
3633     begin
3634     currentpoint pdf.originy ne exch

```

```

3635         pdf.originx ne or
3636         {
3637             pdf.save.linkll
3638             /pdf.lly
3639             pdf.lly pdf.outerbox 1 get sub def
3640             pdf.bordertracking.begin
3641         }
3642         if
3643         end
3644     }
3645     put
3646 }
3647 if
3648 /x where
3649 {
3650     /x
3651     {
3652         0 exch rmoveto
3653         SDict
3654         begin
3655         currentpoint
3656         pdf.originy ne exch pdf.originx ne or
3657         {
3658             pdf.save.linkll
3659             /pdf.lly
3660             pdf.lly pdf.outerbox 1 get sub def
3661             pdf.bordertracking.begin
3662         }
3663         if
3664         end
3665     }
3666     put
3667 }
3668 if
3669 }
3670 def

```

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

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

```

3671 /pdf.breaklink
3672 {
3673     pop
3674     counttomark 2 mod 0 eq
3675     {
3676         counttomark /pdf.count exch def
3677         {
3678             pdf.count 0 eq { exit } if
3679             counttomark 2 roll
3680             1 index /Rect eq

```



```

3681     {
3682     dup 4 array copy
3683     dup dup
3684     1 get
3685     pdf.outerbox pdf.rect.ht
3686     pdf.linkmargin 2 mul add sub
3687     3 exch put
3688     dup
3689     pdf.outerbox 2 get
3690     pdf.linkmargin add
3691     2 exch put
3692     dup dup
3693     3 get
3694     pdf.outerbox pdf.rect.ht
3695     pdf.linkmargin 2 mul add add
3696     1 exch put
3697     /pdf.currentrect exch def
3698     pdf.breaklink.write
3699     {
3700     pdf.currentrect
3701     dup
3702     pdf.outerbox 0 get
3703     pdf.linkmargin sub
3704     0 exch put
3705     dup
3706     pdf.outerbox 2 get
3707     pdf.linkmargin add
3708     2 exch put
3709     dup dup
3710     1 get
3711     pdf.baselineskip add
3712     1 exch put
3713     dup dup
3714     3 get
3715     pdf.baselineskip add
3716     3 exch put
3717     /pdf.currentrect exch def
3718     pdf.breaklink.write
3719     }
3720     1 index 3 get
3721     pdf.linkmargin 2 mul add
3722     pdf.outerbox pdf.rect.ht add
3723     2 index 1 get sub
3724     pdf.baselineskip div round cvi 1 sub
3725     exch
3726     repeat
3727     pdf.currentrect
3728     dup
3729     pdf.outerbox 0 get
3730     pdf.linkmargin sub
3731     0 exch put
3732     dup dup
3733     1 get
3734     pdf.baselineskip add

```

```

3735         1 exch put
3736     dup dup
3737         3 get
3738         pdf.baselineskip add
3739         3 exch put
3740     dup 2 index 2 get 2 exch put
3741     /pdf.currentrect exch def
3742     pdf.breaklink.write
3743     SDict /pdf.pdfmark.good false put
3744     exit
3745     }
3746     { pdf.count 2 sub /pdf.count exch def }
3747     ifelse
3748     }
3749     loop
3750     }
3751     if
3752     /ANN
3753     }
3754     def
3755 /pdf.breaklink.write
3756     {
3757     counttomark 1 sub
3758     index /_objdef eq
3759     {
3760     counttomark -2 roll
3761     dup wcheck
3762     {
3763     readonly
3764     counttomark 2 roll
3765     }
3766     { pop pop }
3767     ifelse
3768     }
3769     if
3770     counttomark 1 add copy
3771     pop pdf.currentrect
3772     /ANN pdfmark
3773     }
3774     def

```

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

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

```

3775 /pdf.pdfmark
3776     {
3777     SDict /pdf.pdfmark.good true put
3778     dup /ANN eq
3779     {
3780     pdf.pdfmark.store

```

```

3781     pdf.pdfmark.dict
3782     begin
3783         Subtype /Link eq
3784         currentdict /Rect known and
3785         SDict /pdf.outerbox known and
3786         SDict /pdf.baselineskip known and
3787         {
3788             Rect 3 get
3789             pdf.linkmargin 2 mul add
3790             pdf.outerbox pdf.rect.ht add
3791             Rect 1 get sub
3792             pdf.baselineskip div round cvi 0 gt
3793             { pdf.breaklink }
3794             if
3795         }
3796     if
3797     end
3798     SDict /pdf.outerbox undef
3799     SDict /pdf.baselineskip undef
3800     currentdict /pdf.pdfmark.dict undef
3801 }
3802 if
3803 pdf.pdfmark.good
3804 { pdfmark }
3805 { cleartomark }
3806 ifelse
3807 }
3808 def
3809 /pdf.pdfmark.store
3810 {
3811     /pdf.pdfmark.dict 65534 dict def
3812     counttomark 1 add copy
3813     pop
3814     {
3815         dup mark eq
3816         {
3817             pop
3818             exit
3819         }
3820         {
3821             pdf.pdfmark.dict
3822             begin def end
3823         }
3824     ifelse
3825 }
3826 loop
3827 }
3828 def

```

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

```
3829 </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.

Symbols	
<code>\</code>	1137
A	
<code>\AtBeginDvi</code>	56
B	
bool commands:	
<code>\bool_gset_false:N</code>	1223, 1242, 1265, 1287, 1303, 1412, 1664, 1700, 2929, 2975
<code>\bool_gset_true:N</code>	1221, 1290, 1410, 1679, 2922, 2928
<code>\bool_if:NTF</code>	66, 596, 1233, 1237, 1253, 1256, 1260, 1271, 1278, 1282, 1294, 1298, 1423, 1428, 1433, 1638, 1683, 1810, 1860, 1994, 2036, 2917, 2932, 2937, 2942
<code>\bool_if:nTF</code>	2420, 2601, 2789
<code>\bool_lazy_and:nnTF</code>	809, 2148, 3305, 3381
<code>\bool_lazy_any:nTF</code>	1849
<code>\bool_lazy_or:nnTF</code>	2029
<code>\bool_new:N</code>	1224, 1291, 1413, 1680, 2902, 2903
<code>\bool_set_false:N</code>	1822, 1958, 2059, 2216
box commands:	
<code>\box_dp:N</code>	235, 237, 285, 287, 342, 344, 391, 393, 395, 397, 2954, 2987, 2988, 3013
<code>\box_ht:N</code>	237, 287, 344, 395, 397, 1873, 2100, 2959, 2998, 2999, 3015
<code>\box_if_empty:NTF</code>	3047
<code>\box_move_down:nn</code>	2875, 2954
<code>\box_move_up:nn</code>	2237, 2877, 2959
<code>\box_new:N</code>	2865, 2866
<code>\box_set_dp:Nn</code>	1771
<code>\box_set_ht:Nn</code>	1770
<code>\box_set_wd:Nn</code>	299, 1769
<code>\box_use:N</code>	242, 260, 274, 290, 317, 331, 347, 363, 375, 426, 440, 459, 1363, 1571, 1772, 2907
<code>\box_wd:N</code>	236, 244, 286, 292, 343, 349, 392, 394, 1872, 2099
box internal commands:	
<code>__box_backend_clip:N</code>	224, 224, 279, 279, 336, 336, 380, 380
<code>\l__box_backend_cos_fp</code>	294
<code>__box_backend_rotate:Nn</code>	246, 246, 294, 294, 351, 351, 430, 430
<code>__box_backend_rotate_aux:Nn</code>	246, 247, 248, 294, 295, 296, 351, 352, 353
<code>__box_backend_scale:Nnn</code>	263, 263, 322, 322, 366, 366, 443, 443
<code>\l__box_backend_sin_fp</code>	294
C	
clist commands:	
<code>\clist_map_function:nN</code>	1311, 1443, 1707
<code>\clist_map_inline:nn</code>	3233
color internal commands:	
<code>__color_backend:nnn</code>	1045, 1060, 1068, 1074
<code>\g__color_backend_colorant_prop</code>	562, 581, 584, 604, 845
<code>__color_backend_devicen_colorants:n</code>	563, 563, 765, 903
<code>__color_backend_devicen_colorants:w</code>	563, 571, 578, 586
<code>__color_backend_devicen_init:nnn</code>	752, 752, 870, 870, 1095, 1095
<code>__color_backend_devicen_init:w</code>	870, 879, 908, 912
<code>__color_backend_fill:n</code>	949, 949, 951, 952, 953, 975, 976, 978, 980, 981, 1000, 1009, 1010, 1012, 1014, 1015, 1026, 1035, 1036, 1038, 1040, 1041
<code>__color_backend_fill_cmyk:n</code>	949, 951, 975, 975, 1009, 1009, 1035, 1035, 1047
<code>__color_backend_fill_devicen:nn</code>	959, 969, 999, 1003, 1025, 1029, 1089, 1091
<code>__color_backend_fill_gray:n</code>	949, 952, 975, 977, 1009, 1011, 1035, 1037
<code>__color_backend_fill_reset:</code>	971, 971, 1005, 1005, 1031, 1031, 1093, 1093
<code>__color_backend_fill_rgb:n</code>	949, 953, 975, 979, 1009, 1013, 1035, 1039
<code>__color_backend_fill_separation:nn</code>	959, 959, 969, 999, 999, 1003, 1025, 1025, 1029, 1089, 1089, 1091

`\l_color_backend_fill_tl` 525, 537, 983, 997
`_color_backend_iccbased_-`
 `device:mnn` 932, 932
`_color_backend_iccbased_-`
 `init:nnn` 771, 771, 914, 914, 1095, 1096
`_color_backend_init_resource:n`
 806, 806, 835, 906, 930, 945
`_color_backend_reset:`
 506, 521, 529, 541, 545, 550,
 971, 972, 1005, 1006, 1031, 1049, 1093
`_color_backend_rgb:w` 1062
`_color_backend_select:n`
 506, 507, 509, 511,
 513, 514, 545, 545, 547, 548, 549, 591
`_color_backend_select:mn`
 529, 530, 532, 534, 535, 802
`_color_backend_select_cmyk:n` ..
 506, 506, 529, 529, 545, 547
`_color_backend_select_devicen:mnn`
 590, 592, 774, 775, 796, 804
`_color_backend_select_gray:n` ..
 506, 508, 529, 531, 545, 548, 555
`_color_backend_select_iccbased:mn`
 593, 593, 778, 778, 796, 805
`_color_backend_select_named:n` ..
 506, 510, 552, 552
`_color_backend_select_rgb:n` ...
 506, 512, 529, 533, 545, 549
`_color_backend_select_separation:mnn`
 590, 590, 592,
 774, 774, 775, 796, 797, 801, 804, 805
`_color_backend_separation_-`
 `init:n` 594, 675, 688
`_color_backend_separation_-`
 `init:nn` 823, 833, 837
`_color_backend_separation_-`
 `init:nnn` 594, 629, 650
`_color_backend_separation_-`
 `init:nnnn` 594, 652, 664
`_color_backend_separation_-`
 `init:nnnnn` 594,
 594, 615, 708, 776, 776, 823, 823, 863
`_color_backend_separation_-`
 `init:nw` 594, 679, 690, 704
`_color_backend_separation_-`
 `init:w` 594, 666, 681, 686
`_color_backend_separation_-`
 `init_/DeviceCMYK:nnn` 594
`_color_backend_separation_-`
 `init_/DeviceGray:nnn` 594
`_color_backend_separation_-`
 `init_/DeviceRGB:nnn` 594
`_color_backend_separation_-`
 `init_aux:nnnnn` 594, 600, 616
`_color_backend_separation_-`
 `init_CIELAB:mnn`
 594, 706, 776, 823, 848
`_color_backend_separation_-`
 `init_CIELAB:nnnnnn` 777
`_color_backend_separation_-`
 `init_count:n` 594, 653, 656
`_color_backend_separation_-`
 `init_count:w` ... 594, 657, 658, 662
`_color_backend_separation_-`
 `init_Device:Nn`
 594, 638, 640, 642, 643
`\l_color_backend_stack_int`
 467, 539, 542, 984, 996
`_color_backend_stroke:n`
 949, 954, 956,
 957, 958, 975, 988, 990, 992, 993, 1002
`_color_backend_stroke_cmyk:n` ..
 949,
 956, 975, 987, 1009, 1019, 1045, 1045
`_color_backend_stroke_devicen:mnn`
 959,
 970, 999, 1004, 1025, 1030, 1089, 1092
`_color_backend_stroke_gray:n` ..
 949,
 957, 975, 989, 1009, 1021, 1045, 1051
`_color_backend_stroke_gray_-`
 `aux:n` 1045, 1055, 1059
`_color_backend_stroke_reset:` ..
 971,
 972, 1005, 1006, 1031, 1032, 1093, 1094
`_color_backend_stroke_rgb:n` ...
 949,
 958, 975, 991, 1009, 1023, 1045, 1061
`_color_backend_stroke_rgb:w` ...
 1045, 1063
`_color_backend_stroke_separation:mnn`
 .. 959, 964, 970, 999, 1001, 1004,
 1025, 1027, 1030, 1089, 1090, 1092
`\l_color_backend_stroke_tl`
 525, 538, 985, 995
`\g_color_model_int` 601, 610, 758,
 786, 835, 841, 842, 896, 897, 906, 930
`\c_color_model_range_CIELAB_tl` ..
 713, 748, 859, 866
`color.sc` 3448
cs commands:
 `\cs_generate_variant:Nn` .. 62, 65,
 170, 181, 212, 218, 615, 1169, 1580,
 2008, 2070, 2090, 2264, 2279, 2342,
 2552, 2565, 2675, 2690, 2720, 3168
 `\cs_gset:Npe` .. 2432, 2436, 2794, 2799

<code>\cs_gset_protected:Npn</code> . . .	3385, 3386	1414, 1416, 1418, 1420, 1425, 1430,
<code>\cs_if_exist:NTF</code>		1435, 1437, 1450, 1455, 1457, 1459,
.	27, 49, 2626, 2652, 3043	1461, 1463, 1465, 1467, 1469, 1488,
<code>\cs_if_exist_p:N</code>	810, 3306, 3382	1512, 1518, 1530, 1542, 1554, 1561,
<code>\cs_if_exist_use:NTF</code>	38, 628	1583, 1589, 1594, 1599, 1610, 1620,
<code>\cs_new:Npe</code>		1630, 1632, 1634, 1636, 1667, 1669,
.	563, 2566, 2577, 2644, 3094, 3129, 3225	1674, 1676, 1678, 1681, 1702, 1713,
<code>\cs_new:Npn</code>	578, 637, 639,	1726, 1728, 1730, 1732, 1734, 1736,
.	641, 643, 650, 656, 658, 664, 681,	1738, 1740, 1742, 1750, 1758, 1784,
.	688, 690, 908, 1316, 1448, 1711,	1802, 1819, 1833, 1838, 1846, 1876,
.	1875, 2103, 2254, 2271, 2343, 2345,	1889, 1907, 1917, 1933, 1946, 1955,
.	2438, 2439, 2521, 2522, 2534, 2553,	1963, 1975, 1981, 1984, 1999, 2009,
.	2554, 2657, 2683, 2721, 2723, 2802,	2047, 2056, 2062, 2068, 2071, 2078,
.	2803, 2815, 2816, 2821, 2822, 2827,	2091, 2096, 2104, 2111, 2124, 2158,
.	2828, 2897, 3068, 3182, 3211, 3220	2189, 2190, 2192, 2194, 2196, 2202,
<code>\cs_new_eq:NN</code>	46, 56, 58, 547,	2205, 2213, 2219, 2222, 2224, 2235,
.	548, 549, 592, 775, 804, 805, 951,	2262, 2265, 2267, 2269, 2273, 2280,
.	952, 953, 956, 957, 958, 969, 970,	2297, 2302, 2307, 2312, 2322, 2327,
.	971, 972, 1003, 1004, 1005, 1006,	2335, 2347, 2373, 2378, 2406, 2418,
.	1029, 1030, 1031, 1091, 1092, 1093,	2430, 2434, 2440, 2442, 2446, 2469,
.	1168, 1372, 1373, 1378, 1379, 1579,	2483, 2493, 2504, 2523, 2555, 2588,
.	1581, 1582, 1588, 1782, 1783, 1795,	2599, 2605, 2633, 2667, 2669, 2676,
.	1796, 1817, 1818, 1881, 1882, 1883,	2678, 2681, 2685, 2691, 2696, 2701,
.	1906, 1931, 1943, 1944, 1952, 1953,	2703, 2705, 2713, 2725, 2747, 2752,
.	1954, 1974, 1977, 1978, 1979, 2043,	2785, 2787, 2792, 2797, 2804, 2806,
.	2053, 2054, 2055, 2203, 2204, 2211,	2810, 2811, 2812, 2813, 2814, 2817,
.	2212, 2221, 2251, 2252, 2253, 2256,	2818, 2819, 2820, 2823, 2824, 2825,
.	2272, 2684, 2907, 3231, 3232, 3242	2826, 2829, 2830, 2833, 2852, 2859,
<code>\cs_new_protected:Npe</code>		2868, 2873, 2906, 2908, 2913, 2915,
.	594, 1074, 2616, 2673, 3166, 3192	2920, 2935, 2940, 2977, 3006, 3025,
<code>\cs_new_protected:Npn</code>		3034, 3070, 3077, 3078, 3081, 3105,
.	47, 53, 60, 63, 71, 77,	3107, 3109, 3120, 3140, 3150, 3157,
.	82, 84, 88, 98, 108, 118, 128, 137,	3170, 3185, 3190, 3209, 3213, 3215,
.	146, 156, 168, 171, 173, 175, 179,	3216, 3219, 3221, 3222, 3223, 3224,
.	184, 193, 203, 213, 224, 246, 248,	3226, 3227, 3228, 3247, 3254, 3262,
.	263, 279, 294, 296, 322, 336, 351,	3270, 3289, 3296, 3320, 3336, 3346,
.	353, 366, 380, 430, 443, 470, 484,	3352, 3358, 3390, 3392, 3394, 3396
.	494, 506, 508, 510, 512, 514, 521,	<code>\cs_set_eq:NN</code>
.	529, 531, 533, 535, 541, 545, 550,	3064, 3066
.	552, 590, 593, 616, 706, 752, 771,	<code>\cs_set_protected:Npn</code>
.	774, 776, 777, 778, 797, 801, 806,	2162
.	823, 837, 848, 870, 914, 932, 949,	
.	954, 959, 964, 975, 977, 979, 981,	
.	987, 989, 991, 993, 999, 1001, 1009,	
.	1011, 1013, 1015, 1019, 1021, 1023,	
.	1025, 1027, 1032, 1035, 1037, 1039,	
.	1041, 1045, 1051, 1059, 1061, 1063,	
.	1089, 1090, 1094, 1095, 1096, 1170,	
.	1176, 1181, 1183, 1185, 1193, 1201,	
.	1210, 1220, 1222, 1225, 1227, 1244,	
.	1249, 1267, 1289, 1292, 1305, 1318,	
.	1323, 1325, 1327, 1329, 1331, 1333,	
.	1335, 1337, 1342, 1347, 1374, 1376,	
.	1380, 1385, 1390, 1400, 1409, 1411,	

D

dim commands:	
<code>\dim_compare:nNnTF</code>	2138, 2143
<code>\dim_compare_p:nNn</code>	2149, 2150
<code>\dim_eval:n</code>	
.	2376, 2479, 2480, 2481, 2750,
.	2841, 2842, 2845, 2871, 3089, 3090,
.	3091, 3148, 3176, 3177, 3178, 3214
<code>\dim_gset:Nn</code>	2854, 2855
<code>\dim_max:nn</code>	2985, 2996
<code>\dim_set:Nn</code>	
.	1872, 1873, 2099, 2100, 2134, 2135
<code>\dim_set_eq:NN</code>	2200
<code>\dim_to_decimal:n</code>	391, 392, 393,

394, 395, 397, 1592, 1597, 1603,
 1604, 1605, 1606, 1615, 1616, 1617,
 1708, 1727, 2244, 2245, 2983, 2994,
 3012, 3013, 3014, 3015, 3019, 3074
 \dim_to_decimal_in_bp:n
 235, 236, 237, 285, 286, 287,
 342, 343, 344, 1189, 1190, 1197,
 1198, 1205, 1206, 1214, 1215, 1216,
 1313, 1317, 1321, 1383, 1388, 1394,
 1395, 1396, 1404, 1405, 1445, 1449,
 1453, 1712, 1789, 1790, 1791, 1792,
 1968, 1969, 1970, 1971, 2023, 2024,
 2025, 2026, 2229, 2230, 2231, 2232
 \dim_zero:N 2132, 2133
 \c_max_dim
 2134, 2135, 2138, 2143, 2149, 2150
 draw internal commands:
 __draw_backend_add_to_path:n . . .
 1589,
 1591, 1596, 1601, 1612, 1620, 1635
 __draw_backend_begin:
 1170, 1170, 1374, 1374, 1583, 1583
 __draw_backend_box_use:Nnnnn . . .
 1347, 1347, 1561, 1561, 1758, 1758
 __draw_backend_cap_but:
 1305, 1325, 1437, 1457, 1702, 1730
 __draw_backend_cap_rectangle: . . .
 1305, 1329, 1437, 1461, 1702, 1734
 __draw_backend_cap_round:
 1305, 1327, 1437, 1459, 1702, 1732
 __draw_backend_clip:
 1225, 1289, 1414, 1430, 1634, 1678
 __draw_backend_closepath:
 1225, 1225,
 1246, 1414, 1414, 1634, 1634, 1671
 __draw_backend_closestroke:
 1225, 1244, 1414, 1418, 1634, 1669
 __draw_backend_curveto:nnnnnn . . .
 1185, 1210, 1380, 1390, 1589, 1610
 __draw_backend_dash:n
 1305, 1311, 1316,
 1437, 1443, 1448, 1702, 1707, 1711
 __draw_backend_dash_aux:nn
 1702, 1706, 1713
 __draw_backend_dash_pattern:nn . . .
 1305, 1305, 1437, 1437, 1702, 1702
 __draw_backend_discardpath:
 1225, 1292, 1414, 1435, 1634, 1681
 __draw_backend_end:
 1170, 1176, 1374, 1376, 1583, 1588
 __draw_backend_evenodd_rule:
 1220, 1220, 1409, 1409, 1630, 1630
 __draw_backend_fill:
 1225, 1249, 1414, 1420, 1634, 1674
 __draw_backend_fillstroke:
 1225, 1267, 1414, 1425, 1634, 1676
 __draw_backend_join_bevel:
 1305, 1335, 1437, 1467, 1702, 1740
 __draw_backend_join_miter:
 1305, 1331, 1437, 1463, 1702, 1736
 __draw_backend_join_round:
 1305, 1333, 1437, 1465, 1702, 1738
 __draw_backend_lineto:nn
 1185, 1193, 1380, 1385, 1589, 1594
 __draw_backend_linewidth:n
 1305, 1318, 1437, 1450, 1702, 1726
 __draw_backend_literal:n
 1168, 1168, 1169, 1172, 1173, 1174,
 1178, 1179, 1182, 1184, 1187, 1195,
 1203, 1212, 1226, 1229, 1230, 1231,
 1232, 1235, 1241, 1251, 1258, 1264,
 1269, 1274, 1275, 1276, 1277, 1280,
 1286, 1296, 1302, 1307, 1320, 1324,
 1326, 1328, 1330, 1332, 1334, 1336,
 1339, 1344, 1349, 1350, 1351, 1352,
 1353, 1354, 1355, 1356, 1357, 1361,
 1362, 1364, 1365, 1366, 1367, 1368,
 1372, 1372, 1373, 1382, 1387, 1392,
 1402, 1415, 1417, 1419, 1422, 1427,
 1432, 1436, 1439, 1452, 1456, 1458,
 1460, 1462, 1464, 1466, 1468, 1514,
 1579, 1579, 1580, 1641, 1660, 1686
 __draw_backend_miterlimit:n
 1305, 1323, 1437, 1455, 1702, 1728
 __draw_backend_moveto:nn
 1185, 1185, 1380, 1380, 1589, 1589
 __draw_backend_nonzero_rule:
 1220, 1222, 1409, 1411, 1630, 1632
 __draw_backend_path:n
 1634, 1636, 1668, 1675, 1677
 \g__draw_backend_path_int 1649, 1666
 \g__draw_backend_path_tl
 1589, 1645, 1661, 1663, 1690, 1699
 __draw_backend_rectangle:nnnn
 1185, 1201, 1380, 1400, 1589, 1599
 __draw_backend_scope_begin: 1181,
 1181, 1375, 1378, 1378, 1581, 1581
 __draw_backend_scope_end: 1181,
 1183, 1377, 1378, 1379, 1581, 1582
 __draw_backend_shift:nn
 1337, 1342, 1469, 1512, 1742, 1750
 __draw_backend_stroke: 1225, 1227,
 1247, 1414, 1416, 1634, 1667, 1672
 __draw_backend_transform:nnnn
 1337, 1337, 1358, 1359,
 1360, 1469, 1469, 1742, 1742, 1761
 __draw_backend_transform_-
 aux:nnnn 1469, 1483, 1488

<code>__draw_backend_transform_-</code>	1526 , 1535 , 1536 , 1537 , 1538 , 1547 ,
<code>decompose:nnnnN</code>	1482 , 1517 , 1518 , 1548 , 1549 , 1550 , 2366 , 2466 , 2743
<code>__draw_backend_transform_-</code>	<code>\fp_new:N</code> 320 , 321
<code>decompose_auxi:nnnnN</code>	<code>\fp_set:Nn</code> 300 , 303
<code>.....</code>	<code>\fp_use:N</code> 306 , 310 , 315
<code>__draw_backend_transform_-</code>	<code>\fp_zero:N</code> 302
<code>decompose_auxii:nnnnN</code>	<code>\c_zero_fp</code> 254 , 301 , 307 , 359 , 1493 , 1506
<code>.....</code>	
<code>__draw_backend_transform_-</code>	
<code>decompose_auxiii:nnnnN</code>	
<code>.....</code>	
<code>\g__draw_draw_clip_bool</code>	1225 , 1634
<code>\g__draw_draw_eor_bool</code>	1220 , 1237 , 1253 , 1260 , 1271 ,
<code>...</code>	1282 , 1298 , 1409 , 1423 , 1428 , 1433
<code>\g__draw_draw_path_int</code>	1634
	G
	graphics commands:
	<code>\l_graphics_search_ext_seq</code> 1781 , 1799 , 1941 , 2122
	graphics internal commands:
	<code>\l_graphics_attr_tl</code> 1801 ,
	1806 , 1823 , 1835 , 1842 , 1844 , 1879
	<code>__graphics_backend_dequote:w</code> 1802 , 1841 , 1875
	<code>\l_graphics_backend_dir_str</code> 1884
	<code>\l_graphics_backend_ext_str</code> 1884
	<code>__graphics_backend_get_pagecount:n</code> 1796 , 1796 , 1933 , 1933 ,
	2042 , 2043 , 2111 , 2111 , 2256 , 2256
	<code>__graphics_backend_getbb_auxi:n</code> 1802 , 1815 , 1831 , 1833
	<code>__graphics_backend_getbb_-</code>
	<code>auxi:nN</code> 2047 , 2051 , 2060 , 2062
	<code>__graphics_backend_getbb_-</code>
	<code>auxii:n</code> 1802 , 1836 , 1838
	<code>__graphics_backend_getbb_-</code>
	<code>auxiii:nnN</code> 2047 , 2065 , 2068 , 2070
	<code>__graphics_backend_getbb_-</code>
	<code>auxiii:n</code> 1802 , 1840 , 1846
	<code>__graphics_backend_getbb_-</code>
	<code>auxiii:nNnn</code> 2047 , 2066 , 2069 , 2071
	<code>__graphics_backend_getbb_-</code>
	<code>auxiv:nnNnn</code> 2047 , 2074 , 2078 , 2090
	<code>__graphics_backend_getbb_-</code>
	<code>auxv:nNnn</code> 2047 , 2075 , 2082 , 2091
	<code>__graphics_backend_getbb_-</code>
	<code>auxvi:nNnn</code> 2094 , 2096
	<code>__graphics_backend_getbb_bmp:n</code> 1943 , 1954 , 2047 , 2055
	<code>__graphics_backend_getbb_eps:n</code> 1782 , 1782 , 1884 ,
	1889 , 1906 , 1943 , 1943 , 2203 , 2203
	<code>__graphics_backend_getbb_eps:nm</code> 1884
	<code>__graphics_backend_getbb_eps:n</code> 1895 , 1907
	<code>__graphics_backend_getbb_jpeg:n</code> 1802 , 1817 ,
	1943 , 1952 , 2047 , 2053 , 2205 , 2211
	<code>__graphics_backend_getbb_jpg:n</code> 1802 , 1802 , 1817 , 1818 , 1943 , 1946 ,

E

<code>\errmessage</code>	38
<code>\evensidemargin</code>	2952
exp commands:	
<code>\exp_after:wN</code>	2109
<code>\exp_args:Ne</code>	598 ,
652 , 833 , 1840 , 1895 , 1897 , 1921 ,	
1923 , 2309 , 2324 , 2375 , 2749 , 2948	
<code>\exp_args:Nf</code>	1310 , 1442 , 2870
<code>\exp_args:Nne</code>	2716
<code>\exp_args:NNf</code>	247 , 295 , 352
<code>\exp_args:Nno</code>	3348
<code>\exp_args:No</code>	3354
<code>\exp_not:N</code>	565 ,
571 , 572 , 573 , 598 , 600 , 601 , 604 ,	
605 , 610 , 2568 , 2570 , 2573 , 2579 ,	
2581 , 2584 , 2621 , 2622 , 2628 , 2629 ,	
2648 , 2653 , 3096 , 3098 , 3101 , 3131 ,	
3133 , 3136 , 3194 , 3195 , 3196 , 3201	
<code>\exp_not:n</code>	48 , 96 , 116 , 154 ,
922 , 2300 , 2305 , 2369 , 2538 , 2539 ,	
2553 , 2554 , 2694 , 2699 , 2710 , 2729	
<code>\ExplBackendFileDate</code>	1

F

file commands:	
<code>\file_compare_timestamp:nNnTF</code>	1909
<code>\file_parse_full_name:nNNN</code>	1891 , 1919
<code>\fmtversion</code>	51
fp commands:	
<code>\fp_compare:nNnTF</code>	254 , 301 , 307 , 359 , 1493 , 1506 , 1556
<code>\fp_eval:n</code>	247 , 256 , 269 , 270 , 295 , 312 , 327 ,
329 , 352 , 361 , 372 , 373 , 437 , 452 ,	
453 , 1056 , 1069 , 1070 , 1071 , 1495 ,	
1500 , 1501 , 1508 , 1523 , 1524 , 1525 ,	

1952, 1953, 1954, [2047](#), [2047](#), 2053,
 2054, 2055, [2205](#), 2205, 2211, 2212
 _graphics_backend_getbb_
 pagebox:w .. [2047](#), [2086](#), [2103](#), [2109](#)
 _graphics_backend_getbb_pdf:n .
 [1802](#), [1819](#), [1915](#),
[1943](#), [1955](#), [2047](#), [2056](#), [2213](#), [2213](#)
 _graphics_backend_getbb_png:n .
 [1802](#), [1818](#),
[1943](#), [1953](#), [2047](#), [2054](#), [2205](#), [2212](#)
 _graphics_backend_getbb_ps:n ..
 [1782](#), [1783](#),
[1884](#), [1906](#), [1943](#), [1944](#), [2203](#), [2204](#)
 _graphics_backend_getbb_svg:n .
 [2124](#), [2124](#)
 _graphics_backend_getbb_svg_
 auxi:nNn ... [2124](#), [2140](#), [2145](#), [2158](#)
 _graphics_backend_getbb_svg_
 auxii:w [2124](#), [2162](#), [2184](#), [2189](#)
 _graphics_backend_getbb_svg_
 auxiii:Nw [2124](#), [2172](#), [2190](#)
 _graphics_backend_getbb_svg_
 auxiv:Nw [2124](#), [2175](#), [2192](#)
 _graphics_backend_getbb_svg_
 auxv:Nw [2124](#), [2176](#), [2194](#)
 _graphics_backend_getbb_svg_
 auxvi:Nn [2124](#), [2191](#), [2193](#), [2195](#), [2196](#)
 _graphics_backend_getbb_svg_
 auxvii:w [2124](#), [2198](#), [2202](#)
 _graphics_backend_include:nn ..
 [2219](#), [2220](#), [2223](#), [2224](#)
 _graphics_backend_include_
 auxi:nn [1963](#), [1976](#), [1982](#), [1984](#)
 _graphics_backend_include_
 auxii:nnn .. [1963](#), [1986](#), [1999](#), [2008](#)
 _graphics_backend_include_
 auxiii:nnn [1963](#), [2006](#), [2009](#)
 _graphics_backend_include_
 bmp:n [1963](#), [1979](#)
 _graphics_backend_include_
 dequote:w [2235](#), [2246](#), [2254](#)
 _graphics_backend_include_
 eps:n [1784](#),
[1784](#), [1795](#), [1884](#), [1917](#), [1931](#),
[1963](#), [1963](#), [1974](#), [2219](#), [2219](#), [2221](#)
 _graphics_backend_include_
 jpeg:n . [1876](#), [1881](#), [1977](#), [2235](#), [2252](#)
 _graphics_backend_include_
 jpg:n [1876](#),
[1876](#), [1881](#), [1882](#), [1883](#), [1963](#),
[1975](#), [1977](#), [1978](#), [1979](#), [2235](#), [2253](#)
 _graphics_backend_include_
 jpseg:n [1963](#)

_graphics_backend_include_
 pdf:n [1876](#), [1882](#), [1921](#),
[1963](#), [1981](#), [2104](#), [2104](#), [2219](#), [2222](#)
 _graphics_backend_include_
 png:n
 .. [1876](#), [1883](#), [1963](#), [1978](#), [2235](#), [2251](#)
 _graphics_backend_include_ps:n
 [1784](#), [1795](#),
[1884](#), [1931](#), [1963](#), [1974](#), [2219](#), [2221](#)
 _graphics_backend_include_
 svg:n .. [2235](#), [2235](#), [2251](#), [2252](#), [2253](#)
 \l_graphics_backend_name_str . [1884](#)
 _graphics_bb_restore:nTF
 [1835](#), [2093](#), [2126](#)
 _graphics_bb_save:n [1844](#), [2101](#), [2153](#)
 \l_graphics_decodearray_str ...
 [1808](#), [1809](#),
[1821](#), [1852](#), [1858](#), [1859](#), [1957](#), [1992](#),
[1993](#), [2031](#), [2034](#), [2035](#), [2058](#), [2215](#)
 _graphics_extract_bb:n
 [1950](#), [1959](#), [2209](#), [2217](#)
 \l_graphics_final_name_str .. [1914](#)
 _graphics_get_pagecount:n
 [1796](#), [2043](#), [2256](#)
 \l_graphics_interpolate_bool ...
 [1810](#), [1822](#), [1851](#), [1860](#),
[1958](#), [1994](#), [2030](#), [2036](#), [2059](#), [2216](#)
 \l_graphics_llx_dim
 [1789](#), [1968](#), [2023](#), [2132](#), [2229](#)
 \l_graphics_lly_dim
 [1790](#), [1969](#), [2024](#), [2133](#), [2230](#)
 \l_graphics_page_int
 [1804](#), [1826](#), [1827](#), [1865](#),
[1866](#), [1948](#), [1990](#), [1991](#), [2017](#), [2018](#),
[2049](#), [2064](#), [2065](#), [2107](#), [2108](#), [2207](#)
 \l_graphics_pagebox_tl
 [55](#), [1805](#), [1825](#),
[1867](#), [1868](#), [1949](#), [1988](#), [1989](#), [2019](#),
[2021](#), [2050](#), [2073](#), [2074](#), [2109](#), [2208](#)
 \l_graphics_pdf_str
 .. [1812](#), [1813](#), [1828](#), [1829](#), [1853](#), [1862](#)
 _graphics_read_bb:n
 .. [1782](#), [1783](#), [1943](#), [1944](#), [2203](#), [2204](#)
 \l_graphics_tmp_box
 .. [1870](#), [1872](#), [1873](#), [2098](#), [2099](#), [2100](#)
 \l_graphics_tmp_dim [2199](#), [2200](#)
 \l_graphics_tmp_ior
 [2128](#), [2129](#), [2136](#), [2155](#)
 \g_graphics_track_int
 [1962](#), [2011](#), [2012](#)
 \l_graphics_urx_dim
 ... [1791](#), [1872](#), [1970](#), [2025](#), [2099](#),
[2134](#), [2138](#), [2141](#), [2149](#), [2231](#), [2244](#)

`\l_graphics_ury_dim`
 1792, 1873, 1971, 2026, 2100, 2135,
 2143, 2146, 2150, 2232, 2237, 2245
 group commands:
`\group_begin:` 190, 209
`\group_end:` 198
`\group_insert_after:N`
 .. 3251, 3252, 3260, 3268, 3334, 3378

H

hbox commands:
`\hbox:n` 2239, 2383, 2390,
 2757, 2768, 2876, 2879, 2955, 2961
`\hbox_overlap_right:n` 242,
 274, 290, 331, 347, 375, 459, 1363, 1571
`\hbox_set:Nn` .. 1870, 2098, 2647, 2979
`\hbox_set:Nw` 2930
`\hbox_set_end:` 2945
`\hbox_unpack:N` 3066
 hook commands:
`\hook_gput_code:nnn` .. 54, 3043, 3045

I

int commands:
`\int_compare:nNnTF`
 1826, 1865, 1990, 2017,
 2064, 2107, 2408, 2619, 2647, 3038
`\int_const:Nn`
 472, 1842, 1936, 2012, 2113
`\int_eval:n` 492, 502, 648, 657, 670,
 672, 676, 689, 2432, 2436, 2597,
 2622, 2629, 2642, 2786, 2794, 2799
`\int_gincr:N` 216,
 382, 1640, 1685, 2011, 2270, 2337,
 2682, 2715, 2886, 2964, 3172, 3194
`\int_gset:Nn` 191, 210, 2513, 3027
`\int_gset_eq:NN` 199, 2965, 3195
`\int_if_exist:NTF` 2001
`\int_if_odd:nTF` 2950
`\int_max:nn` 2115
`\int_new:N` 182, 183, 429, 467, 1666,
 1962, 2867, 2899, 2901, 3169, 3184
`\int_set:Nn` 3039
`\int_set_eq:NN` 187, 206
`\int_step_function:nnnN` 674
`\int_use:N` 384, 415, 601,
 610, 758, 786, 835, 841, 842, 896,
 897, 906, 930, 1643, 1649, 1656,
 1688, 1696, 1827, 1866, 1879, 1937,
 1991, 2004, 2016, 2018, 2108, 2116,
 2339, 2344, 2717, 2722, 2890, 2898,
 2969, 3069, 3175, 3183, 3201, 3212
`\int_value:w`
 2568, 2579, 2597, 3096, 3131

`\int_zero:N` ... 1804, 1948, 2049, 2207
 ior commands:
`\ior_close:N` 2155
`\ior_if_eof:NTF` 2129
`\ior_map_break:` 2151
`\ior_open:Nn` 2128
`\ior_str_map_inline:Nn` 2136

K

kernel internal commands:
`__kernel_backend_align_begin:` ..
 71, 71, 227, 251, 266
`__kernel_backend_align_end:` ...
 71, 77, 241, 259, 273
`__kernel_backend_first_shipout:n`
 49, 53, 56, 58, 68, 598, 2835
`\g__kernel_backend_header_bool` ..
 66, 596
`__kernel_backend_literal:n`
 46, 46, 47, 48, 61, 64, 69,
 73, 80, 83, 85, 169, 172, 174, 176,
 180, 356, 369, 516, 522, 546, 551,
 618, 754, 798, 950, 955, 961, 966,
 1017, 1043, 1477, 1478, 1479, 1490,
 1497, 1503, 1568, 1573, 1786, 1965,
 2003, 2013, 2226, 2241, 2674, 2786,
 2790, 2795, 2800, 2837, 3167, 3214
`__kernel_backend_literal_page:n`
 108, 108,
 118, 171, 171, 2668, 2670, 2805, 2807
`__kernel_backend_literal_pdf:n` .
 88, 88, 98, 168, 168, 170,
 282, 339, 1372, 1373, 3328, 3339, 3372
`__kernel_backend_literal_-
 postscript:n` 60,
 60, 62, 74, 75, 79, 228, 229, 231,
 232, 240, 252, 267, 1168, 2410, 2422
`__kernel_backend_literal_svg:n` .
 . 179, 179, 181, 186, 197, 205, 215,
 383, 385, 402, 780, 1579, 1762, 1773
`__kernel_backend_matrix:n`
 .. 146, 146, 156, 304, 325, 1472, 1565
`__kernel_backend_postscript:n` ..
 63, 63, 65, 518,
 1020, 1022, 1024, 1028, 2263, 2314,
 2329, 2349, 2383, 2390, 2876, 2882,
 2887, 2923, 2955, 2962, 2966, 2980,
 3008, 3051, 3058, 3065, 3072, 3272
`__kernel_backend_scope:n`
 ... 184, 213, 218, 412, 417, 1048,
 1076, 1586, 1631, 1633, 1653, 1693,
 1715, 1727, 1729, 1731, 1733, 1735,
 1737, 1739, 1741, 1744, 1752, 3397

<code>__kernel_backend_scope_begin:</code> ...	<code>__opacity_backend_fill:n</code>
82, 82, 128 , 128, 173 , 173 , 184 , 184,	.. 3247 , 3254 , 3346 , 3346 , 3390 , 3392
226, 250, 265, 281, 298, 324, 338,	<code>__opacity_backend_fill_stroke:nn</code>
355, 368, 1378, 1563, 1581, 1585, 1760	.. 3346 , 3348 , 3354 , 3358 , 3381 , 3386
<code>__kernel_backend_scope_begin:n</code> ..	<code>\l__opacity_backend_fill_tl</code>
..... 184 , 203 , 212 , 404 , 432 , 445 3316 , 3322 , 3355 , 3363
<code>__kernel_backend_scope_end:</code> ...	<code>__opacity_backend_reset:</code>
..... 82 , 84 , 128 , 137 , 3320 , 3334 , 3336 , 3378
173 , 175 , 184 , 193 , 243 , 261 , 275 ,	<code>__opacity_backend_reset_fill:</code> ..
291 , 318 , 332 , 348 , 364 , 376 , 427 , 3247 , 3251 , 3260 , 3289
441 , 460 , 1379 , 1575 , 1582 , 1588 , 1774	<code>__opacity_backend_reset_stroke:</code>
<code>\g__kernel_backend_scope_int</code> 3247 , 3252 , 3268 , 3296
182 , 189 , 191 , 196 , 200 , 208 , 210 , 216	<code>__opacity_backend_select:n</code>
<code>\l__kernel_backend_scope_int</code> 3247 , 3247 , 3320 ,
..... 182 , 188 , 201 , 207	3320 , 3361 , 3381 , 3385 , 3390 , 3390
<code>\g__kernel_clip_path_int</code>	<code>\c__opacity_backend_stack_int</code> ...
380 , 1640 , 1643 , 1656 , 1685 , 1688 , 1696 3305 , 3331 , 3343 , 3375
<code>__kernel_color_backend_stack_-</code>	<code>__opacity_backend_stroke:n</code>
init:Nnn 3247 , 3262 , 3346 , 3352 , 3390 , 3394
..... 470 , 470 , 3310	<code>\l__opacity_backend_stroke_tl</code> ..
<code>__kernel_color_backend_stack_-</code> 3316 , 3323 , 3350 , 3364
pop:n	
..... 484 , 494 , 542 , 3343	
<code>__kernel_color_backend_stack_-</code>	
push:nn	
.. 484 , 484 , 539 , 984 , 996 , 3331 , 3375	
<code>__kernel_dependency_version_-</code>	
check:Nn	
..... 1	
<code>__kernel_dependency_version_-</code>	
check:nn	
..... 27 , 29	
<code>__kernel_file_name_quote:n</code>	
..... 1897 , 1923	
<code>__kernel_kern:n</code>	
..... 2382 , 2386 , 2389 , 2393 ,	
2756 , 2764 , 2767 , 2783 , 2881 , 2883	
L	
lua commands:	
<code>\lua_load_module:n</code>	
..... 1162	
M	
<code>\MessageBreak</code>	
..... 40	
mode commands:	
<code>\mode_if_horizontal:TF</code> ...	
..... 3029 , 3036	
<code>\mode_if_math:TF</code>	
..... 2927	
msg commands:	
<code>\msg_error:nnn</code>	
..... 556 , 2130	
<code>\msg_new:nnn</code>	
..... 558	
O	
<code>\oddsidemargin</code>	
..... 2951	
opacity internal commands:	
<code>__opacity_backend:nn</code>	
..... 3390 , 3391 , 3393 , 3395 , 3396	
<code>__opacity_backend:nnn</code>	
..... 3247 , 3249 ,	
3250 , 3256 , 3264 , 3270 , 3291 , 3298	
<code>__opacity_backend_fill:n</code>	
.. 3247 , 3254 , 3346 , 3346 , 3390 , 3392	
<code>__opacity_backend_fill_stroke:nn</code>	
.. 3346 , 3348 , 3354 , 3358 , 3381 , 3386	
<code>\l__opacity_backend_fill_tl</code>	
..... 3316 , 3322 , 3355 , 3363	
<code>__opacity_backend_reset:</code>	
..... 3320 , 3334 , 3336 , 3378	
<code>__opacity_backend_reset_fill:</code> ..	
..... 3247 , 3251 , 3260 , 3289	
<code>__opacity_backend_reset_stroke:</code>	
..... 3247 , 3252 , 3268 , 3296	
<code>__opacity_backend_select:n</code>	
..... 3247 , 3247 , 3320 ,	
3320 , 3361 , 3381 , 3385 , 3390 , 3390	
<code>\c__opacity_backend_stack_int</code> ...	
..... 3305 , 3331 , 3343 , 3375	
<code>__opacity_backend_stroke:n</code>	
.. 3247 , 3262 , 3346 , 3352 , 3390 , 3394	
<code>\l__opacity_backend_stroke_tl</code> ..	
..... 3316 , 3323 , 3350 , 3364	
P	
pdf commands:	
<code>\pdf_object_if_exist:nTF</code>	
..... 850 , 916 , 934	
<code>\pdf_object_new:n</code>	
..... 841 , 852 , 896 , 918 , 936	
<code>\pdf_object_ref:n</code>	
..... 798 , 865 , 929 , 944 , 962 , 967	
<code>\pdf_object_ref_last:</code>	
..... 818 , 843 , 846 , 902	
<code>\pdf_object_unnamed_write:nn</code> ...	
..... 825 , 872 , 928 , 943	
<code>\pdf_object_write:nnn</code>	
..... 842 , 853 , 897 , 919 , 937	
pdf internal commands:	
<code>__pdf_backend:n</code>	
..... 2673 , 2673 , 2675 , 2677 , 2679 , 2693 ,	
2698 , 2707 , 2727 , 2759 , 2760 , 2770	
<code>__pdf_backend_annotation:nnnn</code>	
..... 3231	
<code>__pdf_backend_annotation_last:</code>	
..... 3232	
<code>__pdf_backend_bdc:nn</code>	
..... 2440 , 2440 ,	
2667 , 2667 , 2804 , 2804 , 2829 , 2829	
<code>__pdf_backend_catalog_gput:nn</code> ..	
..... 2265 , 2265 ,	
2483 , 2483 , 2676 , 2676 , 2812 , 2812	
<code>__pdf_backend_compress_objects:n</code>	
..... 2406 , 2418 ,	
2588 , 2599 , 2785 , 2787 , 2823 , 2824	
<code>__pdf_backend_compresslevel:n</code> ..	
..... 2406 , 2406 ,	
2588 , 2588 , 2785 , 2785 , 2823 , 2823	

_pdf_backend_destination:nn . . .	_pdf_backend_object_write_-
. 2347 , 2347 ,	stream:nnn 2273 , 2325 , 2327
2446 , 2446 , 2725 , 2725 , 2810 , 2810	_pdf_backend_object_write_-
_pdf_backend_destination:nnnn .	stream:nnnn . 2685 , 2702 , 2704 , 2705
. 2347 , 2373 ,	_pdf_backend_pageobject_ref:n .
2446 , 2469 , 2725 , 2747 , 2810 , 2811 2345 , 2345 ,
_pdf_backend_destination_-	2577 , 2577 , 2723 , 2723 , 2814 , 2822
aux:nnnn	_pdf_backend_pagesize_gset:nn .
. 2347 , 2375 , 2378 , 2725 , 2749 , 2752 2833 , 2833 , 2852 , 2852 , 2859 , 2859
_pdf_backend_emc:	_pdf_backend_pdfmark:n
. 2440 , 2442 , 2262 , 2262 , 2264 , 2266 , 2268 , 2282 ,
2667 , 2669 , 2804 , 2806 , 2829 , 2830	2299 , 2304 , 2350 , 2394 , 2441 , 2443
_pdf_backend_info_gput:nn	_pdf_backend_version_major:
. 2265 , 2267 , 2432 , 2438 , 2438 , 2644 , 2644 ,
2483 , 2493 , 2676 , 2678 , 2812 , 2813	2794 , 2795 , 2802 , 2802 , 2827 , 2827
_pdf_backend_objcompresslevel:n	_pdf_backend_version_major_-
. 2588 , 2602 , 2603 , 2605	gset:n 2430 , 2430 ,
_pdf_backend_object_id:n	2616 , 2616 , 2792 , 2792 , 2825 , 2825
. 2269 , 2272 ,	_pdf_backend_version_minor:
2504 , 2522 , 2681 , 2684 , 2814 , 2816 2436 , 2438 , 2439 , 2644 , 2657 ,
\g_pdf_backend_object_int	2799 , 2800 , 2802 , 2803 , 2827 , 2828
. 2270 , 2337 , 2339 ,	_pdf_backend_version_minor_-
2344 , 2513 , 2682 , 2715 , 2717 , 2722	gset:n 2430 , 2434 ,
_pdf_backend_object_last:	2616 , 2633 , 2792 , 2797 , 2825 , 2826
. 2343 , 2343 ,	_pdf_exp_not_i:nn
2566 , 2566 , 2721 , 2721 , 2814 , 2821 2523 , 2542 , 2547 , 2553
_pdf_backend_object_new:	_pdf_exp_not_ii:nn
. 2269 , 2269 , 2523 , 2543 , 2548 , 2554
2504 , 2504 , 2681 , 2681 , 2814 , 2814	pdf.baselineskip 3775
_pdf_backend_object_now:nn	pdf.bordertracking 3533
. 2335 , 2335 , 2342 , 2555 , 2555 , 2565 ,	pdf.bordertracking.begin 3533
2713 , 2713 , 2720 , 2814 , 2819 , 2820	pdf.bordertracking.continue 3533
\g_pdf_backend_object_prop	pdf.bordertracking.end 3533
. 2503 , 2680	pdf.bordertracking.endpage 3533
_pdf_backend_object_ref:n	pdf.breaklink 3671
. 2269 , 2271 , 2272 , 2276 , 2504 , 2521 ,	pdf.breaklink.write 3671
2681 , 2683 , 2684 , 2688 , 2814 , 2815	pdf.brokenlink.dict 3533
_pdf_backend_object_write:nn	pdf.brokenlink.rect 3533
. 2523 , 2532 , 2534 , 2563 , 2814	pdf.brokenlink.skip 3533
_pdf_backend_object_write:nnn	pdf.count 3671
. 2273 , 2273 , 2279 , 2523 , 2523 , 2552 ,	pdf.currentrect 3671
2685 , 2685 , 2690 , 2814 , 2817 , 2818	pdf.cvs 3455
_pdf_backend_object_write_-	pdf.dest.anchor 3498
array:nn 2273 , 2297 , 2685 , 2691	pdf.dest.point 3498
_pdf_backend_object_write_-	pdf.dest.x 3498
aux:nnn 2273 , 2275 , 2280 , 2338	pdf.dest.y 3498
_pdf_backend_object_write_-	pdf.dest2device 3498
dict:nn 2273 , 2302 , 2685 , 2696	pdf.dev.x 3498
_pdf_backend_object_write_-	pdf.dev.y 3498
fstream:nn 2273 , 2307 , 2685 , 2701	pdf.dvi.pt 3455
_pdf_backend_object_write_-	pdf.globaldict 3452
fstream:nnn 2310 , 2312	pdf.leftboundary 3533
_pdf_backend_object_write_-	pdf.linkdp.pad 3459
stream:nn 2273 , 2322 , 2685 , 2703	pdf.linkht.pad 3459

pdf.linkmargin	3459	_pdfannot_backend_link_-	
pdf.llx	3462	begin:nw	2910, 2914, 2915
pdf.lly	3462	_pdfannot_backend_link_begin_-	
pdf.originx	3533	aux:nw	2918, 2920
pdf.originy	3533	_pdfannot_backend_link_begin_-	
pdf.outerbox	3775	goto:nnw	2908, 2908,
pdf.pdfmark	3775		3105, 3105, 3185, 3185, 3221, 3221
pdf.pdfmark.dict	3775	_pdfannot_backend_link_begin_-	
pdf.pdfmark.good	3775	user:nnw	2908, 2913,
pdf.pt.dvi	3455		3105, 3107, 3185, 3190, 3221, 3222
pdf.rect	3462	\\g_pdfannot_backend_link_bool	..
pdf.rect.ht	3455		2903, 2917, 2922, 2937, 2975
pdf.rightboundary	3533	\\g_pdfannot_backend_link_dict_-	
pdf.save.linkll	3462	tl	2900, 2925, 2970
pdf.save.linkur	3462	_pdfannot_backend_link_end:...	
pdf.save.ll	3462		2908, 2935,
pdf.save.ur	3462		3105, 3120, 3185, 3209, 3221, 3224
pdf.tmpa	3498	_pdfannot_backend_link_end_-	
pdf.tmpb	3498	aux:	2908, 2938, 2940
pdf.tmpc	3498	\\g_pdfannot_backend_link_int...	
pdf.tmpd	3498		2899, 2965,
pdf.urx	3462		2969, 3069, 3184, 3195, 3201, 3212
pdf.ury	3462	_pdfannot_backend_link_last:..	
pdfannot internal commands:			3068, 3068,
_pdfannot_backend:n	3166, 3166,		3129, 3129, 3211, 3211, 3225, 3225
3168, 3173, 3197, 3210, 3215, 3216		_pdfannot_backend_link_-	
\\l_pdfannot_backend_breaklink_-		margin:n	3070, 3070,
pdfmark_tl	2904, 2972, 3063		3140, 3140, 3213, 3213, 3226, 3226
_pdfannot_backend_breaklink_-		\\g_pdfannot_backend_link_math_-	
postscript:n		bool	2902, 2928, 2929, 2932, 2942
2906, 2906, 2956, 2958, 3064		_pdfannot_backend_link_minima:	
_pdfannot_backend_breaklink_-			2908, 2946, 2977
usebox:N	2907, 2907, 2957, 3066	_pdfannot_backend_link_off:...	
\\l_pdfannot_backend_content_box			3077, 3078,
2865,			3150, 3157, 3215, 3216, 3227, 3228
2930, 2954, 2957, 2959, 2988, 2999		_pdfannot_backend_link_on:...	
_pdfannot_backend_generic:nnnn			3077, 3077,
2868, 2868, 3081,			3150, 3150, 3215, 3215, 3227, 3227
3081, 3170, 3170, 3219, 3219, 3231		_pdfannot_backend_link_-	
_pdfannot_backend_generic_-		outerbox:n	2908, 2948, 3006
aux:nnnn	2868, 2870, 2873	\\g_pdfannot_backend_link_sf_int	
\\g_pdfannot_backend_int			2901, 3027, 3038, 3039
2867, 2886, 2890, 2898, 2964, 2965,		_pdfannot_backend_link_sf_-	
3169, 3172, 3175, 3183, 3194, 3196		restore:	2908, 2931, 2974, 3034
_pdfannot_backend_last:.....		_pdfannot_backend_link_sf_-	
2897, 2897, 3094,		save:	2908, 2926, 2944, 3025
3094, 3182, 3182, 3220, 3220, 3232		\\l_pdfannot_backend_model_box..	
_pdfannot_backend_link:nw	2908		2866,
_pdfannot_backend_link_aux:nw	2908		2947, 2979, 2987, 2998, 3013, 3015
_pdfannot_backend_link_begin:n		pdfmanagement commands:	
3185, 3187, 3191, 3192		\\pdfmanagement_add:nnn
_pdfannot_backend_link_-			815, 3313, 3324, 3365, 3368
begin:nnw		\\pdfmanagement_if_active_p:...	
3105, 3106, 3108, 3109, 3221, 3223			810, 811, 3306, 3307, 3382, 3383

peek commands:		
\peek_meaning:NTF	2171, 2174	
\peek_remove_spaces:n	2169	
prg commands:		
\prg_replicate:nn	195, 646, 667, 677, 878	
prop commands:		
\prop_gput:Nnn	604, 845	
\prop_if_in:NnTF	581	
\prop_item:Nn	584	
\prop_new:N	562, 2503, 2680	
\ProvidesExplFile	2	
		Q
quark commands:		
\quark_if_recursion_tail_stop:n	580	
\q_recursion_stop	573	
\q_recursion_tail	572	
		S
scan commands:		
\scan_stop	131, 140, 502, 2199, 2202, 2467, 2481, 2597, 2614, 2622, 2629, 2642, 3123, 3148	
scan internal commands:		
\s__color_stop	657, 658, 662, 666, 679, 682, 686, 690, 704, 879, 908, 912, 1062, 1064	
\s__graphics_stop	1841, 1875, 2164, 2179, 2186, 2190, 2192, 2194, 2246, 2254	
separation	<u>3449</u>	
seq commands:		
\seq_set_from_clist:Nn	1781, 1799, 1941, 2122	
shipout commands:		
\l_shipout_box	3047, 3049, 3057	
skip commands:		
\skip_horizontal:n	244, 292, 349	
str commands:		
\c_hash_str	415, 1649, 1656, 1696	
\c_percent_str	1082, 1083, 1084	
\str_case:nn	884, 2286, 2536	
\str_case:nnTF	2354, 2455, 2732	
\str_convert_pdfname:n	605, 625, 834	
\str_if_empty:NTF	1812, 1828	
\str_if_empty_p:N	1853	
\str_if_eq:nnTF	554, 784, 1475, 3360	
\str_new:N	1886, 1887, 1888	
\str_tail:N	1900, 1926	
sys commands:		
\sys_if_shell:TF	1884	
\sys_shell_now:n	1911	
		T
TeX and L ^A T _E X 2 _ε commands:		
\@ifl@t@r	49, 51	
\special	2	
tex commands:		
\tex_afterassignment:D	2198	
\tex_baselineskip:D	3019	
\tex_endinput:D	44	
\tex_global:D	2590, 2607, 2621, 2628, 2635	
\tex_immediate:D	1848, 2526, 2529, 2558, 2561	
\tex_luatexversion:D	2619, 2647	
\tex_pageheight:D	2855	
\tex_pagewidth:D	2854	
\tex_pdfannot:D	3087	
\tex_pdfcatalog:D	2489	
\tex_pdfcolorstack:D	490, 500	
\tex_pdfcolorstackinit:D	478	
\tex_pdfcompresslevel:D	2595	
\tex_pdfdest:D	2452, 2475	
\tex_pdfendlink:D	3126	
\tex_pdfextension:D	91, 101, 111, 121, 131, 140, 149, 159, 487, 497, 2449, 2472, 2486, 2496, 2507, 2526, 2558, 3084, 3112, 3123, 3152, 3159	
\tex_pdffeedback:D	475, 2515, 2570, 2581, 3098, 3133	
\tex_pdfinfo:D	2499	
\tex_pdflastannot:D	3101	
\tex_pdflastlink:D	3136	
\tex_pdflastobj:D	2518, 2573	
\tex_pdflastximage:D	1843, 1871	
\tex_pdflastximagepages:D	1937	
\tex_pdflinkmargin:D	3146	
\tex_pdfliteral:D	94, 104, 114, 124	
\tex_pdfmajorversion:D	2626, 2628, 2652, 2653	
\tex_pdfminorversion:D	2640, 2664	
\tex_pdfobj:D	2510, 2529, 2561	
\tex_pdfobjcompresslevel:D	2612	
\tex_pdfpageref:D	2584	
\tex_pdfrefximage:D	1871, 1878	
\tex_pdfrestore:D	143	
\tex_pdfrunninglinkoff:D	3162	
\tex_pdfrunninglinkon:D	3155	
\tex_pdfsave:D	134	
\tex_pdfsetmatrix:D	152, 162	
\tex_pdfstartlink:D	3115	
\tex_pdfvariable:D	2592, 2609, 2621, 2637, 2648, 2661, 3143	
\tex_pdfximage:D	1848, 1935	
\tex_spacefactor:D	3030, 3039	
\tex_special:D	46	

