

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}{2025-03-10}{}
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2025-03-10}{}
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2025-03-10}{}
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2025-03-10}{}
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2025-03-10}{}
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2025-03-10}{}
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 behaviour 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 `xdvipdmx` (`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 dvipdmtx/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  
\g__kernel_clip_path_int
```

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

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

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

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

```

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

```

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

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

```

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

```
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 </dviptfm | 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: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_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:nw
\__color_backend_separation_init_CIELAB:nnm

```

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

```

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:nnn No support at present.

```

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

```

(End of definition for __color_backend_iccbased_init:nnn.)

```

772 </dvips>

```

```

773 <*dvisvgm>

```

__color_backend_select_separation:nn No support at present.

__color_backend_select_devicen:nn

```

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

```

```

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

```

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

__color_backend_separation_init:nnnnn No support at present.

__color_backend_separation_init_CIELAB:nnn

```

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

```

```

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

```

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

__color_backend_select_iccbased:nn

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

```

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

```

```

779 {
780   \__kernel_backend_literal_svg:e
781   {

```

```

782     <style>

```

```

783       @color-profile ~

```

```

784       \str_if_eq:nnTF {#2} { cmyk }

```

```

785       { device-cmyk }

```

```

786       { --color \int_use:N \g__color_model_int }

```

```

787       \c_space_tl

```

```

788     {

```

```

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:nnnnn
\_color_backend_separation_init:nn
\_color_backend_separation_init_CIELAB:nnn

```

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

```

823 \cs_new_protected:Npn \_color_backend_separation_init:nnnnn #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_pdfname:n {#1} } {#2}
835     \_color_backend_init_resource:n { color \int_use:N \g_color_model_int }
836 }
837 \cs_new_protected:Npn \_color_backend_separation_init:nn #1#2
838 {
839     \use:e
840     {
841         \pdf_object_new:n { color \int_use:N \g_color_model_int }
842         \pdf_object_write:nnn { color \int_use:N \g_color_model_int } { array }
843         { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
844     }
845     \prop_gput:Nne \g_color_backend_colorant_prop { /#1 }
846     { \pdf_object_ref_last: }
847 }

```

For CIELAB colors, we need one object per document for the illuminant, plus initialisation 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:nnm 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/XYTeX` 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 \langle /luatex | pdftex \rangle

1008 \langle *dvips \rangle

`_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.)

`_color_backend_fill_separation:nn`
`_color_backend_stroke_separation:nn`
`_color_backend_fill_devicen:nn`
`_color_backend_stroke_devicen:nn`

At present, these are no-ops.

```

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

```

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

`_color_backend_fill_reset:`
`_color_backend_stroke_reset:`

```

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

```

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

`_color_backend_devicen_init:nnn`
`_color_backend_iccbased_init:nnn`

No support at present.

```

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

```

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

```

1097 </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:
\__draw_backend_stroke:
\__draw_backend_closestroke:
\__draw_backend_fill:
\__draw_backend_fillstroke:
\__draw_backend_clip:
\__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:

```

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

```

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_butt:
 __draw_backend_cap_round:
 __draw_backend_cap_rectangle:
 __draw_backend_join_miter:
 __draw_backend_join_round:
 __draw_backend_join_bevel:

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

```

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

```

```

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

```

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

`_draw_backend_transform: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:Nmmnn`

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:Nmmnn #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
\_draw_backend_curveto:nnnnn
\_draw_backend_rectangle:nnnn
1380 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1381   {
1382     \_draw_backend_literal:e
1383     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1384   }
1385 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1386   {
1387     \_draw_backend_literal:e
1388     { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1389   }
1390 \cs_new_protected:Npn \_draw_backend_curveto:nnnnn #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 {
    \_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 optimise for the case where there is only a scaling.

```

    \cs_new_protected:Npn \_draw_backend_transform:nnnn #1#2#3#4
    \_draw_backend_dash:n      1469 {
    \_draw_backend_dash:n      1470 {
    \_draw_backend_dash:n      1471 \langle *luatex | pdftex \rangle
    \_draw_backend_dash:n      1472   \_kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
    \_draw_backend_dash:n      1473 \langle /luatex | pdftex \rangle
    \_draw_backend_dash:n      1474 \langle *dvipdfmx | xetex \rangle
    \_draw_backend_dash:n      1475   \str_if_eq:nnTF { #2 ~ #3 } { 0 ~ 0 }
    \_draw_backend_dash:n      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
__draw_backend_shift:nn vector.

```

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>

```

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

```

1780 \cs_new_protected:Npn \__graphics_backend_loaded:n #1
1781 {
1782   \cs_if_exist:NTF \hook_gput_code:nnn
1783   {
1784     \hook_gput_code:nnn
1785     { package / l3graphics / after }
1786     { backend }
1787     {#1}
1788   }
1789   {#1}
1790 }

```

(End of definition for __graphics_backend_loaded:n.)

5.1 dvips backend

```
1791 <*dvips>
```

\l_graphics_search_ext_seq

```

1792 \__graphics_backend_loaded:n
1793 { \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_ps:n

```

1794 \__graphics_backend_loaded:n
1795 {
1796   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1797   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
1798 }

```

(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_ps:n
1799 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1800 {
1801   \__kernel_backend_literal:e
1802   {
1803     PSfile = #1 \c_space_tl
1804     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1805     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1806     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1807     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1808   }
1809 }
1810 \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`

```
1811 \__graphics_backend_loaded:n
1812 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }
```

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

```
1813 </dvips>
```

5.2 Lua_T_EX and pdf_T_EX backends

```
1814 < *luatex | pdftex >
```

`\l_graphics_search_ext_seq`

```
1815 \__graphics_backend_loaded:n
1816 {
1817   \seq_set_from_clist:Nn
1818   \l_graphics_search_ext_seq
1819   { .pdf , .eps , .ps , .png , .jpg , .jpeg }
1820 }
```

(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.

```
1821 \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.

```
1822 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
```



```

1823 {
1824   \int_zero:N \l__graphics_page_int
1825   \tl_clear:N \l__graphics_pagebox_tl
1826   \tl_set:Ne \l__graphics_attr_tl
1827   {
1828     \tl_if_empty:NF \l__graphics_decodearray_str
1829     { :D \l__graphics_decodearray_str }
1830     \bool_if:NT \l__graphics_interpolate_bool
1831     { :I }
1832     \str_if_empty:NF \l__graphics_pdf_str
1833     { :X \l__graphics_pdf_str }
1834   }
1835   \__graphics_backend_getbb_auxi:n {#1}
1836 }
1837 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1838 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1839 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1840 {
1841   \tl_clear:N \l__graphics_decodearray_str
1842   \bool_set_false:N \l__graphics_interpolate_bool
1843   \tl_set:Ne \l__graphics_attr_tl
1844   {
1845     : \l__graphics_pagebox_tl
1846     \int_compare:nNnT \l__graphics_page_int > 1
1847     { :P \int_use:N \l__graphics_page_int }
1848     \str_if_empty:NF \l__graphics_pdf_str
1849     { :X \l__graphics_pdf_str }
1850   }
1851   \__graphics_backend_getbb_auxi:n {#1}
1852 }
1853 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1854 {
1855   \__graphics_bb_restore:eF { #1 \l__graphics_attr_tl }
1856   { \__graphics_backend_getbb_auxiii:n {#1} }
1857 }

```

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.

```

1858 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1859 {
1860   \exp_args:Ne \__graphics_backend_getbb_auxiiii:n
1861   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1862   \int_const:cn { c__graphics_ #1 \l__graphics_attr_tl _int }
1863   { \tex_the:D \tex_pdflastximage:D }
1864   \__graphics_bb_save:e { #1 \l__graphics_attr_tl }
1865 }
1866 \cs_new_protected:Npn \__graphics_backend_getbb_auxiiii:n #1
1867 {
1868   \tex_immediate:D \tex_pdfximage:D
1869   \bool_lazy_any:nT
1870   {
1871     { \l__graphics_interpolate_bool }

```

```

1872     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1873     { ! \str_if_empty_p:N \l__graphics_pdf_str }
1874   }
1875   {
1876     attr ~
1877     {
1878       \tl_if_empty:NF \l__graphics_decodearray_str
1879       { /Decode~[ \l__graphics_decodearray_str ] }
1880       \bool_if:NT \l__graphics_interpolate_bool
1881       { /Interpolate~true }
1882       \l__graphics_pdf_str
1883     }
1884   }
1885   \int_compare:nNnT \l__graphics_page_int > 0
1886   { page ~ \int_use:N \l__graphics_page_int }
1887   \tl_if_empty:NF \l__graphics_pagebox_tl
1888   { \l__graphics_pagebox_tl }
1889   {#1}
1890   \hbox_set:Nn \l__graphics_internal_box
1891   { \tex_pdfrefximage:D \tex_pdflastximage:D }
1892   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1893   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1894   }
1895   \cs_new:Npn \__graphics_backend_dequote:w #1 " #2 " #3 \s__graphics_stop {#2}

```

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

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

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

```

1896   \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1897   {
1898     \tex_pdfrefximage:D
1899     \int_use:c { c__graphics_ #1 \l__graphics_attr_tl_int }
1900   }
1901   \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1902   \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1903   \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

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

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

EPS graphics may be included in LuaTeX/pdfTeX by conversion to PDF: this requires restricted shell escape. Modelled on the epstopdf L^AT_EX₂ ϵ package, but simplified, conversion takes place here if we have shell access.

```

1904   \sys_if_shell:T
1905   {
1906     \str_new:N \l__graphics_backend_dir_str
1907     \str_new:N \l__graphics_backend_name_str
1908     \str_new:N \l__graphics_backend_ext_str
1909     \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1910     {
1911       \file_parse_full_name:nNNN {#1}
1912       \l__graphics_backend_dir_str
1913       \l__graphics_backend_name_str

```

```

1914     \l__graphics_backend_ext_str
1915     \exp_args:Ne \__graphics_backend_getbb_eps:nn
1916     {
1917         \exp_args:Ne \__kernel_file_name_quote:n
1918         {
1919             \l__graphics_backend_name_str
1920             - \str_tail:N \l__graphics_backend_ext_str
1921             -converted-to.pdf
1922         }
1923     }
1924     {#1}
1925 }
1926 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_backend_getbb_eps:n
1927 \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1928 {
1929     \file_compare_timestamp:nNnT {#2} > {#1}
1930     {
1931         \sys_shell_now:n
1932         { repstopdf ~ #2 ~ #1 }
1933     }
1934     \tl_set:Nn \l__graphics_final_name_str {#1}
1935     \__graphics_backend_getbb_pdf:n {#1}
1936 }
1937 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1938 {
1939     \file_parse_full_name:nNNN {#1}
1940     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1941     \exp_args:Ne \__graphics_backend_include_pdf:n
1942     {
1943         \exp_args:Ne \__kernel_file_name_quote:n
1944         {
1945             \l__graphics_backend_name_str
1946             - \str_tail:N \l__graphics_backend_ext_str
1947             -converted-to.pdf
1948         }
1949     }
1950 }
1951 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1952 }

```

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

__graphics_backend_get_pagecount:n Simply load and store.

```

1953 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1954 {
1955     \tex_pdfximage:D {#1}
1956     \int_const:cn { c__graphics_ #1 _pages_int }
1957     { \int_use:N \tex_pdflastximagepages:D }
1958 }

```

(End of definition for __graphics_backend_get_pagecount:n.)

1959 </luatex | pdftex>

5.3 dvipdfmx backend

1960 $\langle *dvipdfmx | xetex \rangle$

$\backslash l_graphics_search_ext_seq$

```
1961 \__graphics_backend_loaded:n
1962 {
1963   \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1964   { .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }
1965 }
```

(End of definition for $\backslash l_graphics_search_ext_seq$.)

$\backslash _graphics_backend_getbb_eps:n$

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

$\backslash _graphics_backend_getbb_ps:n$

```
1966 \__graphics_backend_loaded:n
```

$\backslash _graphics_backend_getbb_jpg:n$

```
1967 {
```

$\backslash _graphics_backend_getbb_jpeg:n$

```
1968   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
```

$\backslash _graphics_backend_getbb_pdf:n$

```
1969   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
```

$\backslash _graphics_backend_getbb_png:n$

```
1970 }
```

$\backslash _graphics_backend_getbb_bmp:n$

```
1971  $\langle *dvipdfmx \rangle$ 
```

```
1972 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
```

```
1973 {
```

```
1974   \int_zero:N \l__graphics_page_int
```

```
1975   \tl_clear:N \l__graphics_pagebox_tl
```

```
1976   \__graphics_extract_bb:n {#1}
```

```
1977 }
```

```
1978 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
```

```
1979 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
```

```
1980 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
```

```
1981 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
```

```
1982 {
```

```
1983   \tl_clear:N \l__graphics_decodearray_str
```

```
1984   \bool_set_false:N \l__graphics_interpolate_bool
```

```
1985   \__graphics_extract_bb:n {#1}
```

```
1986 }
```

```
1987  $\langle /dvipdfmx \rangle$ 
```

(End of definition for $\backslash _graphics_backend_getbb_eps:n$ and others.)

$\backslash g_graphics_track_int$

Used to track the object number associated with each graphic.

```
1988 \int_new:N \g_graphics_track_int
```

(End of definition for $\backslash g_graphics_track_int$.)

$\backslash _graphics_backend_include_eps:n$

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

$\backslash _graphics_backend_include_ps:n$

$\backslash _graphics_backend_include_jpg:n$

$\backslash _graphics_backend_include_jpeg:n$

$\backslash _graphics_backend_include_pdf:n$

```
1989 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
```

$\backslash _graphics_backend_include_png:n$

```
1990 {
```

$\backslash _graphics_backend_include_bmp:n$

```
1991   \__kernel_backend_literal:e
```

$\backslash _graphics_backend_include_auxi:n$

```
1992   {
```

$\backslash _graphics_backend_include_auxii:nn$

```
1993     PSfile = #1 \c_space_tl
```

$\backslash _graphics_backend_include_auxiii:nn$

```
1994     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
```

```
1995     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
```

```
1996     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
```

```

1997     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1998   }
1999 }
2000 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
2001 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
2002   { \__graphics_backend_include_auxi:nn {#1} { image } }
2003 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
2004 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
2005 \cs_new_eq:NN \__graphics_backend_include_bmp:n \__graphics_backend_include_jpg:n
2006 <*dvipdfmx>
2007 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2008   { \__graphics_backend_include_auxi:nn {#1} { epdf } }
2009 </dvipdfmx>

```

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

```

2010 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
2011   {
2012     \__graphics_backend_include_auxii:enn
2013     {
2014       \tl_if_empty:NF \l__graphics_pagebox_tl
2015       { : \l__graphics_pagebox_tl }
2016       \int_compare:nNnT \l__graphics_page_int > 1
2017       { :P \int_use:N \l__graphics_page_int }
2018       \tl_if_empty:NF \l__graphics_decodearray_str
2019       { :D \l__graphics_decodearray_str }
2020       \bool_if:NT \l__graphics_interpolate_bool
2021       { :I }
2022     }
2023     {#1} {#2}
2024   }
2025 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
2026   {
2027     \int_if_exist:cTF { c__graphics_ #2#1 _int }
2028     {
2029       \__kernel_backend_literal:e
2030       { pdf:usexobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
2031     }
2032     { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
2033   }
2034 \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!

```

2035 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
2036   {
2037     \int_gincr:N \g__graphics_track_int
2038     \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
2039     \__kernel_backend_literal:e
2040     {
2041       pdf:#3~

```

```

2042     @graphic \int_use:c { c__graphics_ #1#2 _int } ~
2043     \int_compare:nNnT \l__graphics_page_int > 1
2044     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2045     \tl_if_empty:NF \l__graphics_pagebox_tl
2046     {
2047         pagebox ~ \l__graphics_pagebox_tl \c_space_tl
2048         bbox ~
2049             \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2050             \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2051             \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2052             \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
2053     }
2054     (#1)
2055     \bool_lazy_or:nnT
2056     { \l__graphics_interpolate_bool }
2057     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
2058     {
2059         <<
2060         \tl_if_empty:NF \l__graphics_decodearray_str
2061         { /Decode~[ \l__graphics_decodearray_str ] }
2062         \bool_if:NT \l__graphics_interpolate_bool
2063         { /Interpolate~true }
2064         >>
2065     }
2066 }
2067 }

```

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

`__graphics_backend_get_pagecount:n`

```

2068 < *dvipdfmx >
2069 \__graphics_backend_loaded:n
2070 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }
2071 < /dvipdfmx >

```

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

```
2072 < /dvipdfmx | xetex >
```

5.4 X_YTeX backend

```
2073 < *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:n
\__graphics_backend_getbb_auxii:n
\__graphics_backend_getbb_auxiii:n
\__graphics_backend_getbb_auxiv:n
\__graphics_backend_getbb_auxv:n
\__graphics_backend_getbb_auxvi:n
\__graphics_backend_getbb_pagebox:w

```

```

2074 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2075 {
2076     \int_zero:N \l__graphics_page_int
2077     \tl_clear:N \l__graphics_pagebox_tl
2078     \__graphics_backend_getbb_auxi:n {#1} \tex_XeTeXpicfile:D
2079 }
2080 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2081 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n

```

```

2082 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2083 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2084 {
2085   \tl_clear:N \l__graphics_decodearray_str
2086   \bool_set_false:N \l__graphics_interpolate_bool
2087   \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2088 }
2089 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2090 {
2091   \int_compare:nNnTF \l__graphics_page_int > 1
2092     { \__graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2093     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2094 }
2095 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2096 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2097 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2098 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2099 {
2100   \tl_if_empty:NTF \l__graphics_pagebox_tl
2101     { \__graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2102     { \__graphics_backend_getbb_auxv:nNnn }
2103     {#1} #2 {#3} {#4}
2104 }
2105 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2106 {
2107   \use:e
2108   {
2109     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2110     {
2111       #5
2112       \tl_if_blank:nF {#1}
2113         { \c_space_tl \__graphics_backend_getbb_pagebox:w #1 }
2114     }
2115   }
2116 }
2117 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2118 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2119 {
2120   \__graphics_bb_restore:nF {#1#3}
2121   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2122 }
2123 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2124 {
2125   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
2126   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2127   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2128   \__graphics_bb_save:n {#1#3}
2129 }
2130 \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_YTeX is best done using the \tex_XeTeXpdffile: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`.

```

2131 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2132 {
2133   \tex_XeTeXpdffile:D #1 ~
2134   \int_compare:nNnT \l__graphics_page_int > 0
2135     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2136     \exp_after:wN \__graphics_backend_getbb_pagebox:w \l__graphics_pagebox_tl
2137 }

```

(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.

```

2138 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
2139 {
2140   \int_const:cn { c__graphics_#1_pages_int }
2141   {
2142     \int_max:nn
2143     { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2144     { 1 }
2145   }
2146 }

```

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

```
2147 </xetex>
```

5.5 dvisvgm backend

```
2148 <*dvisvgm>
```

`\l_graphics_search_ext_seq`

```

2149 \__graphics_backend_loaded:n
2150 {
2151   \seq_set_from_clist:Nn
2152   \l_graphics_search_ext_seq
2153   { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }
2154 }

```

(End of definition for `\l_graphics_search_ext_seq`.)

`__graphics_backend_getbb_svg:n` 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.

```

\__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_auxvi:Nw
\__graphics_backend_getbb_svg_auxvii:w
2155 \cs_new_protected:Npn \__graphics_backend_getbb_svg:n #1
2156 {
2157   \__graphics_bb_restore:nF {#1}
2158   {
2159     \ior_open:Nn \l__graphics_internal_ior {#1}
2160     \ior_if_eof:NTF \l__graphics_internal_ior
2161     { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2162     {
2163       \dim_zero:N \l__graphics_llx_dim
2164       \dim_zero:N \l__graphics_lly_dim

```



```

2165     \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2166     \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2167     \ior_str_map_inline:Nn \l__graphics_internal_ior
2168     {
2169         \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2170         {
2171             \__graphics_backend_getbb_svg_auxi:nNn
2172             { width } \l__graphics_urx_dim {##1}
2173         }
2174         \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2175         {
2176             \__graphics_backend_getbb_svg_auxi:nNn
2177             { height } \l__graphics_ury_dim {##1}
2178         }
2179         \bool_lazy_and:nnF
2180         { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2181         { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2182         { \ior_map_break: }
2183     }
2184     \__graphics_bb_save:n {#1}
2185 }
2186 \ior_close:N \l__graphics_internal_ior
2187 }
2188 }
2189 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2190 {
2191     \use:e
2192     {
2193         \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxii:w
2194         ##1 \tl_to_str:n {#1} = ##2 \tl_to_str:n {#1} = ##3
2195         \s__graphics_stop
2196     }
2197     {
2198         \tl_if_blank:nF {##2}
2199         {
2200             \peek_remove_spaces:n
2201             {
2202                 \peek_meaning:NTF ' % '
2203                 { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2204                 {
2205                     \peek_meaning:NTF " % "
2206                     { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2207                     { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2208                 }
2209             }
2210             ##2 \s__graphics_stop
2211         }
2212     }
2213     \use:e
2214     {
2215         \__graphics_backend_getbb_svg_auxii:w #3
2216         \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2217         \s__graphics_stop
2218     }

```

```

2219 }
2220 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2221 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop
2222 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2223 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2224 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2225 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2226 { \__graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2227 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxvi:Nn #1#2
2228 {
2229   \tex_afterassignment:D \__graphics_backend_getbb_svg_auxvii:w
2230   \l__graphics_internal_dim #2 bp \scan_stop:
2231   \dim_set_eq:NN #1 \l__graphics_internal_dim
2232 }
2233 \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 Simply use the generic function.

```

\__graphics_backend_getbb_ps:n
2234 \__graphics_backend_loaded:n
2235 {
2236   \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
2237   \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
2238 }

```

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

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

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_jpeg:n
2239 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2240 {
2241   \int_zero:N \l__graphics_page_int
2242   \tl_clear:N \l__graphics_pagebox_tl
2243   \__graphics_extract_bb:n {#1}
2244 }
2245 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2246 \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

```

2247 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2248 {
2249   \tl_clear:N \l__graphics_decodearray_str
2250   \bool_set_false:N \l__graphics_interpolate_bool
2251   \__graphics_extract_bb:n {#1}
2252 }

```

(End of definition for __graphics_backend_getbb_pdf:n.)

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

```

\__graphics_backend_include_ps:n
\__graphics_backend_include_pdf:n
\__graphics_backend_include_mn
2253 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
2254 { \__graphics_backend_include:mn { PSfile } {#1} }
2255 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n

```

```

2256 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
2257 { \__graphics_backend_include:nn { pdffile } {#1} }
2258 \cs_new_protected:Npn \__graphics_backend_include:nn #1#2
2259 {
2260   \__kernel_backend_literal:e
2261   {
2262     #1 = #2 \c_space_tl
2263     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2264     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2265     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2266     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2267   }
2268 }

```

(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.

```

2269 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2270 {
2271   \box_move_up:nn { \l__graphics_ury_dim }
2272   {
2273     \hbox:n
2274     {
2275       \__kernel_backend_literal:e
2276       {
2277         dvisvgm:img~
2278         \dim_to_decimal:n { \l__graphics_urx_dim } ~
2279         \dim_to_decimal:n { \l__graphics_ury_dim } ~
2280         \__graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2281       }
2282     }
2283   }
2284 }
2285 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_svg:n
2286 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_svg:n
2287 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_svg:n
2288 \cs_new:Npn \__graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2289 {#2}

```

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

```

\__graphics_backend_get_pagecount:n

```

```

2290 \__graphics_backend_loaded:n
2291 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }

```

(End of definition for __graphics_backend_get_pagecount:n.)

```

2292 </dvisvgm>
2293 </package>

```

6 I3backend-pdf implementation

```
2294 (*package)
2295 (@@=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

```
2296 (*dvips)
```

```
\_pdf_backend_pdfmark:n Used often enough it should be a separate function.
\_pdf_backend_pdfmark:e
2297 \cs_new_protected:Npn \_pdf_backend_pdfmark:n #1
2298 { \_kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2299 \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
2300 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2301 { \_pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2302 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2303 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }
```

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

6.1.2 Objects

```
\_pdf_backend_object_new:
\_pdf_backend_object_ref:n
\_pdf_backend_object_id:n
2304 \cs_new_protected:Npn \_pdf_backend_object_new:
2305 { \int_gincr:N \g_pdf_backend_object_int }
2306 \cs_new:Npn \_pdf_backend_object_ref:n #1 { { pdf.obj #1 } }
2307 \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: some work to get things right. To allow code
\_pdf_backend_object_write:nne sharing with the anonymous version, we use an auxiliary.
```

```
\_pdf_backend_object_write_aux:nnn
\_pdf_backend_object_write_array:nn
\_pdf_backend_object_write_dict:nn
\_pdf_backend_object_write_fstream:nn
\_pdf_backend_object_write_stream:nn
\_pdf_backend_object_write_stream:nnn
2308 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
2309 {
2310   \_pdf_backend_object_write_aux:nnn
2311     { \_pdf_backend_object_ref:n {#1} }
2312     {#2} {#3}
2313 }
2314 \cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nne }
2315 \cs_new_protected:Npn \_pdf_backend_object_write_aux:nnn #1#2#3
2316 {
2317   \_pdf_backend_pdfmark:e
2318   {
2319     /_objdef ~ #1
```

```

2320     /type
2321     \str_case:nn {#2}
2322     {
2323         { array } { /array }
2324         { dict } { /dict }
2325         { fstream } { /stream }
2326         { stream } { /stream }
2327     }
2328     /OBJ
2329 }
2330 \use:c { __pdf_backend_object_write_ #2 :nn } {#1} {#3}
2331 }
2332 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2333 {
2334     \__pdf_backend_pdfmark:e
2335     { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2336 }
2337 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2338 {
2339     \__pdf_backend_pdfmark:e
2340     { #1 << \exp_not:n {#2} >> /PUT }
2341 }
2342 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2343 {
2344     \exp_args:Ne
2345     \__pdf_backend_object_write_fstream:nnn {#1} #2
2346 }
2347 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2348 {
2349     \__kernel_backend_postscript:n
2350     {
2351         SDict ~ begin ~
2352         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2353         mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2354         end
2355     }
2356 }
2357 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2358 {
2359     \exp_args:Ne
2360     \__pdf_backend_object_write_stream:nnn {#1} #2
2361 }
2362 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2363 {
2364     \__kernel_backend_postscript:n
2365     {
2366         mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2367         mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2368     }
2369 }

```

(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

```

2370 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2371 {
2372   \int_gincr:N \g__pdf_backend_object_int
2373   \__pdf_backend_object_write_aux:nnn
2374   { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2375   {#1} {#2}
2376 }
2377 \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.

```

2378 \cs_new:Npn \__pdf_backend_object_last:
2379 { { 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.

```

2380 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2381 { { Page #1 } }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.1.3 Destinations

__pdf_backend_destination:nn
 __pdf_backend_destination:nnnn
 __pdf_backend_destination_aux:nnnn

Here, we need to turn the zoom into a scale. 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.

```

2382 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2383 {
2384   \__kernel_backend_postscript:n { pdf.dest.anchor }
2385   \__pdf_backend_pdfmark:e
2386   {
2387     /View
2388     [
2389       \str_case:nnF {#2}
2390       {
2391         { xyz } { /XYZ ~ pdf.dest.point ~ null }
2392         { fit } { /Fit }
2393         { fitb } { /FitB }
2394         { fitbh } { /FitBH ~ pdf.dest.y }
2395         { fitbv } { /FitBV ~ pdf.dest.x }
2396         { fith } { /FitH ~ pdf.dest.y }
2397         { fitv } { /FitV ~ pdf.dest.x }
2398         { fitr } { /Fit }
2399       }
2400       {
2401         /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2402       }
2403     ]
2404     /Dest ( \exp_not:n {#1} ) cvn
2405     /DEST
2406   }

```

```

2407 }
2408 \cs_new_protected:Npn \__pdf_backend_destination:n nnn #1#2#3#4
2409 {
2410   \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2411     { \dim_eval:n {#2} } {#1} {#3} {#4}
2412 }
2413 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2414 {
2415   \vbox_to_zero:n
2416     {
2417       \__kernel_kern:n {#4}
2418       \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2419       \tex_vss:D
2420     }
2421   \__kernel_kern:n {#1}
2422   \vbox_to_zero:n
2423     {
2424       \__kernel_kern:n { -#3 }
2425       \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2426       \tex_vss:D
2427     }
2428   \__kernel_kern:n { -#1 }
2429   \__pdf_backend_pdfmark:n
2430     {
2431     /View
2432     [
2433       /FitR ~
2434       pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2435       pdf.urx ~ pdf.ury ~ pdf.dest2device
2436     ]
2437     /Dest ( #2 ) cvn
2438     /DEST
2439     }
2440 }

```

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

6.1.4 Structure

__pdf_backend_compresslevel:n
 __pdf_backend_compress_objects:n

Doable for the usual ps2pdf method.

```

2441 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2442 {
2443   \int_compare:nNnT {#1} = 0
2444     {
2445       \__kernel_backend_literal_postscript:n
2446         {
2447           /setdistillerparams ~ where
2448             { pop << /CompressPages ~ false >> setdistillerparams }
2449           if
2450         }
2451     }
2452 }
2453 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1

```

```

2454 {
2455   \bool_if:nF {#1}
2456   {
2457     \__kernel_backend_literal_postscript:n
2458     {
2459       /setdistillerparams ~ where
2460       { pop << /CompressStreams ~ false >> setdistillerparams }
2461       if
2462     }
2463   }
2464 }

```

(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

```

2465 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2466 {
2467   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2468 }
2469 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2470 {
2471   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2472 }

```

(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:

```

2473 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2474 \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:

```

2475 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
2476 { \__pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2477 \cs_new_protected:Npn \__pdf_backend_emc:
2478 { \__pdf_backend_pdfmark:n { /EMC } }

```

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

2479 </dvips>

6.2 LuaTeX and pdfTeX backend

2480 <*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 of an unterminated keyword. The zoom given here is a percentage, but we need to pass it as *per mille*. The rectangle version is also easy as everything is build in.

__pdf_backend_destination:nnnn

```

2481 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2482 {

```



```

2483 <*luatex>
2484   \tex_pdfextension:D dest ~
2485 </luatex>
2486 <*pdftex>
2487   \tex_pdfdest:D
2488 </pdftex>
2489   name {#1}
2490   \str_case:nnF {#2}
2491   {
2492     { xyz } { xyz }
2493     { fit } { fit }
2494     { fitb } { fitb }
2495     { fitbh } { fitbh }
2496     { fitbv } { fitbv }
2497     { fith } { fith }
2498     { fitv } { fitv }
2499     { fitr } { fitr }
2500   }
2501   { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2502   \scan_stop:
2503 }
2504 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2505 {
2506 <*luatex>
2507   \tex_pdfextension:D dest ~
2508 </luatex>
2509 <*pdftex>
2510   \tex_pdfdest:D
2511 </pdftex>
2512   name {#1}
2513   fitr ~
2514   width \dim_eval:n {#2} ~
2515   height \dim_eval:n {#3} ~
2516   depth \dim_eval:n {#4} \scan_stop:
2517 }

```

(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
2518 \cs_new_protected:Npn \__pdf_backend_catalog_gput:nn #1#2
2519 {
2520 <*luatex>
2521   \tex_pdfextension:D catalog
2522 </luatex>
2523 <*pdftex>
2524   \tex_pdfcatalog:D
2525 </pdftex>
2526   { / #1 ~ #2 }
2527 }
2528 \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2529 {
2530 <*luatex>

```

```

2531     \tex_pdfextension:D info
2532 </luatex>
2533 <*pdftex>
2534     \tex_pdfinfo:D
2535 </pdftex>
2536     { / #1 ~ #2 }
2537 }

```

(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 finalisation.

```

2538 \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 2539 \cs_new_protected:Npn \_pdf_backend_object_new:
\_pdf_backend_object_id:n 2540 {
2541 <*luatex>
2542     \tex_pdfextension:D obj ~
2543 </luatex>
2544 <*pdftex>
2545     \tex_pdfobj:D
2546 </pdftex>
2547     reserveobjnum ~
2548     \int_gset:Nn \g_pdf_backend_object_int
2549 <*luatex>
2550     { \tex_pdffeedback:D lastobj }
2551 </luatex>
2552 <*pdftex>
2553     { \tex_pdflastobj:D }
2554 </pdftex>
2555 }
2556 \cs_new:Npn \_pdf_backend_object_ref:n #1 { #1 ~ 0 ~ R }
2557 \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 2558 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
\_pdf_backend_object_write:nn 2559 {
  \_pdf_exp_not_i:nn 2560 <*luatex>
  \_pdf_exp_not_ii:nn 2561     \tex_immediate:D \tex_pdfextension:D obj ~
2562 </luatex>
2563 <*pdftex>
2564     \tex_immediate:D \tex_pdfobj:D
2565 </pdftex>
2566     useobjnum ~ #1
2567     \_pdf_backend_object_write:nn {#2} {#3}
2568 }
2569 \cs_new:Npn \_pdf_backend_object_write:nn #1#2
2570 {

```

```

2571 \str_case:nn {#1}
2572 {
2573   { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2574   { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2575   { fstream }
2576   {
2577     stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2578     file ~ { \_pdf_exp_not_ii:nn #2 }
2579   }
2580   { stream }
2581   {
2582     stream ~ attr ~ { \_pdf_exp_not_i:nn #2 } ~
2583     { \_pdf_exp_not_ii:nn #2 }
2584   }
2585 }
2586 }
2587 \cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nne }
2588 \cs_new:Npn \_pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2589 \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

```

2590 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2591 {
2592   <*luatex>
2593   \tex_immediate:D \tex_pdfextension:D obj ~
2594   </luatex>
2595   <*pdftex>
2596   \tex_immediate:D \tex_pdfobj:D
2597   </pdftex>
2598   \_pdf_backend_object_write:nn {#1} {#2}
2599 }
2600 \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.

```

2601 \cs_new:Npe \_pdf_backend_object_last:
2602 {
2603   \exp_not:N \int_value:w
2604   <*luatex>
2605   \exp_not:N \tex_pdffeedback:D lastobj ~
2606   </luatex>
2607   <*pdftex>
2608   \exp_not:N \tex_pdflastobj:D
2609   </pdftex>
2610   \c_space_tl 0 ~ R
2611 }

```

(End of definition for _pdf_backend_object_last:.)

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

```

2612 \cs_new:Npe \_pdf_backend_pageobject_ref:n #1
2613 {

```

```

2614     \exp_not:N \int_value:w
2615 <*luatex>
2616     \exp_not:N \tex_pdffeedback:D pageref
2617 </luatex>
2618 <*pdftex>
2619     \exp_not:N \tex_pdfpageref:D
2620 </pdftex>
2621     \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2622 }

```

(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 2623 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
\__pdf_backend_compress_objects:n 2624 {
\__pdf_backend_objcompresslevel:n 2625     \tex_global:D
2626 <*luatex>
2627     \tex_pdfvariable:D compresslevel
2628 </luatex>
2629 <*pdftex>
2630     \tex_pdfcompresslevel:D
2631 </pdftex>
2632     \int_value:w \int_eval:n {#1} \scan_stop:
2633 }
2634 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2635 {
2636     \bool_if:nTF {#1}
2637     { \__pdf_backend_objcompresslevel:n { 2 } }
2638     { \__pdf_backend_objcompresslevel:n { 0 } }
2639 }
2640 \cs_new_protected:Npn \__pdf_backend_objcompresslevel:n #1
2641 {
2642     \tex_global:D
2643 <*luatex>
2644     \tex_pdfvariable:D objcompresslevel
2645 </luatex>
2646 <*pdftex>
2647     \tex_pdfobjcompresslevel:D
2648 </pdftex>
2649     #1 \scan_stop:
2650 }

```

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

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

```

\__pdf_backend_version_major_gset:n 2651 \cs_new_protected:Npe \__pdf_backend_version_major_gset:n #1
\__pdf_backend_version_minor_gset:n 2652 {
2653 <*luatex>
2654     \int_compare:nNnT \tex_luatexversion:D > { 106 }
2655     {
2656         \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2657         \exp_not:N \int_eval:n {#1} \scan_stop:

```

```

2658     }
2659 </luatex>
2660 <*pdftex>
2661     \cs_if_exist:NT \tex_pdfmajorversion:D
2662     {
2663         \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2664         \exp_not:N \int_eval:n {#1} \scan_stop:
2665     }
2666 </pdftex>
2667 }
2668 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2669 {
2670     \tex_global:D
2671 <*luatex>
2672     \tex_pdfvariable:D minorversion
2673 </luatex>
2674 <*pdftex>
2675     \tex_pdfminorversion:D
2676 </pdftex>
2677     \int_eval:n {#1} \scan_stop:
2678 }

```

(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:
2679 \cs_new:Npe \__pdf_backend_version_major:
2680 {
2681 <*luatex>
2682     \int_compare:nNnTF \tex luatexversion:D > { 106 }
2683     { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2684     { 1 }
2685 </luatex>
2686 <*pdftex>
2687     \cs_if_exist:NTF \tex_pdfmajorversion:D
2688     { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2689     { 1 }
2690 </pdftex>
2691 }
2692 \cs_new:Npn \__pdf_backend_version_minor:
2693 {
2694     \tex_the:D
2695 <*luatex>
2696     \tex_pdfvariable:D minorversion
2697 </luatex>
2698 <*pdftex>
2699     \tex_pdfminorversion:D
2700 </pdftex>
2701 }

```

(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:`

```
2702 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2703   { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2704 \cs_new_protected:Npn \_pdf_backend_emc:
2705   { \__kernel_backend_literal_page:n { EMC } }
```

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

```
2706 </luatex | pdftex>
```

6.3 dvipdfmx backend

```
2707 < *dvipdfmx | xetex >
```

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

```
\_pdf_backend:e
2708 \cs_new_protected:Npe \_pdf_backend:n #1
2709   { \__kernel_backend_literal:n { pdf: #1 } }
2710 \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
2711 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2712   { \_pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2713 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2714   { \_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 finalisation.

```
2715 \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
\_pdf_backend_object_id:n
2716 \cs_new_protected:Npn \_pdf_backend_object_new:
2717   { \int_gincr:N \g__pdf_backend_object_int }
2718 \cs_new:Npn \_pdf_backend_object_ref:n #1 { @pdf.obj #1 }
2719 \cs_new_eq:NN \_pdf_backend_object_id:n \_pdf_backend_object_ref:n
```

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

```

\__pdf_backend_object_write:nnn
\__pdf_backend_object_write:nne
\__pdf_backend_object_write_array:nn
\__pdf_backend_object_write_dict:nn
\__pdf_backend_object_write_fstream:nn
\__pdf_backend_object_write_stream:nn
\__pdf_backend_object_write_stream:nnnn

```

This is where we choose the actual type.

```

2720 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2721 {
2722   \use:c { __pdf_backend_object_write_ #2 :nn }
2723   { \__pdf_backend_object_ref:n {#1} } {#3}
2724 }
2725 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nne }
2726 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2727 {
2728   \__pdf_backend:e
2729   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2730 }
2731 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2732 {
2733   \__pdf_backend:e
2734   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2735 }
2736 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2737 { \__pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2738 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2739 { \__pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2740 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnnn #1#2#3#4
2741 {
2742   \__pdf_backend:e
2743   {
2744     #1 stream ~ #2 ~
2745     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2746   }
2747 }

```

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

```

\__pdf_backend_object_now:nn
\__pdf_backend_object_now:nne

```

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

```

2748 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2749 {
2750   \int_gincr:N \g__pdf_backend_object_int
2751   \exp_args:Nne \use:c { __pdf_backend_object_write_ #1 :nn }
2752   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2753   {#2}
2754 }
2755 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { ne }

```

(End of definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last:

```

2756 \cs_new:Npn \__pdf_backend_object_last:
2757 { @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_YT_Z.

```

2758 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2759 { @page #1 }

```

(End of definition for __pdf_backend_pageobject_ref:n.)

6.3.3 Destinations

`_pdf_backend_destination:nn
 _pdf_backend_destination:nmmn
 _pdf_backend_destination_aux:nmmn`

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

```

2760 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2761 {
2762   \_pdf_backend:e
2763   {
2764     dest ~ ( \exp_not:n {#1} )
2765     [
2766       @thispage
2767       \str_case:nnF {#2}
2768       {
2769         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
2770         { fit } { /Fit }
2771         { fitb } { /FitB }
2772         { fitbh } { /FitBH }
2773         { fitbv } { /FitBV ~ @xpos }
2774         { fith } { /FitH ~ @ypos }
2775         { fitv } { /FitV ~ @xpos }
2776         { fitr } { /Fit }
2777       }
2778       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
2779     ]
2780   }
2781 }
2782 \cs_new_protected:Npn \_pdf_backend_destination:nmmn #1#2#3#4
2783 {
2784   \exp_args:Ne \_pdf_backend_destination_aux:nmmn
2785   { \dim_eval:n {#2} } {#1} {#3} {#4}
2786 }
2787 \cs_new_protected:Npn \_pdf_backend_destination_aux:nmmn #1#2#3#4
2788 {
2789   \vbox_to_zero:n
2790   {
2791     \_kernel_kern:n {#4}
2792     \hbox:n
2793     {
2794       \_pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
2795       \_pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
2796     }
2797     \tex_vss:D
2798   }
2799   \_kernel_kern:n {#1}
2800   \vbox_to_zero:n
2801   {
2802     \_kernel_kern:n { -#3 }
2803     \hbox:n
2804     {
2805       \_pdf_backend:n
2806       {
2807         dest ~ (#2)
  
```



```

2808         [
2809         @thispage
2810         /FitR ~
2811         @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
2812         @xpos ~ @ypos
2813         ]
2814     }
2815 }
2816 \tex_vss:D
2817 }
2818 \__kernel_kern:n { -#1 }
2819 }

```

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

6.3.4 Structure

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

```

2820 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
2821 { \__kernel_backend_literal:e { dvipdfmx:config~z~ \int_eval:n {#1} } }
2822 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
2823 {
2824   \bool_if:nF {#1}
2825   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
2826 }

```

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

`__pdf_backend_version_major_gset:n`
`__pdf_backend_version_minor_gset:n` We start with the assumption that the default is active.

```

2827 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
2828 {
2829   \cs_gset:Npe \__pdf_backend_version_major: { \int_eval:n {#1} }
2830   \__kernel_backend_literal:e { pdf:majorversion~ \__pdf_backend_version_major: }
2831 }
2832 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
2833 {
2834   \cs_gset:Npe \__pdf_backend_version_minor: { \int_eval:n {#1} }
2835   \__kernel_backend_literal:e { pdf:minorversion~ \__pdf_backend_version_minor: }
2836 }

```

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

`__pdf_backend_version_major:`
`__pdf_backend_version_minor:` We start with the assumption that the default is active.

```

2837 \cs_new:Npn \__pdf_backend_version_major: { 1 }
2838 \cs_new:Npn \__pdf_backend_version_minor: { 5 }

```

(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:`

```
2839 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2840   { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2841 \cs_new_protected:Npn \_pdf_backend_emc:
2842   { \__kernel_backend_literal_page:n { EMC } }
```

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

```
2843 </dviptfm | xetex>
```

6.4 dvisvgm backend

```
2844 <*dvisvgm>
```

6.4.1 Destinations

`_pdf_backend_destination:nn`
`_pdf_backend_destination:nmmn`

```
2845 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2 { }
2846 \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`
`_pdf_backend_info_gput:nn`

No-op.

```
2847 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2 { }
2848 \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:`
`_pdf_backend_object_ref:n`
`_pdf_backend_object_id:n`
`_pdf_backend_object_write:nmm`
`_pdf_backend_object_write:nne`
`_pdf_backend_object_now:nn`
`_pdf_backend_object_now:nne`
`_pdf_backend_object_last:`
`_pdf_backend_pageobject_ref:n`

All no-ops here.

```
2849 \cs_new_protected:Npn \_pdf_backend_object_new: { }
2850 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
2851 \cs_new:Npn \_pdf_backend_object_id:n #1 { }
2852 \cs_new_protected:Npn \_pdf_backend_object_write:nmm #1#2#3 { }
2853 \cs_new_protected:Npn \_pdf_backend_object_write:nne #1#2#3 { }
2854 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
2855 \cs_new_protected:Npn \_pdf_backend_object_now:nne #1#2 { }
2856 \cs_new:Npn \_pdf_backend_object_last: { }
2857 \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 2858 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1 { }
2859 \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 2860 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1 { }
2861 \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: 2862 \cs_new:Npn \__pdf_backend_version_major: { -1 }
2863 \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: 2864 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2 { }
2865 \cs_new_protected:Npn \__pdf_backend_emc: { }

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

2866 </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.

```
2867 <*dvipdfmx | dvips>
```

```

\__pdf_backend_pagesize_gset:nn This is done as a backend literal, so we deal with it using the shipout hook.
2868 \cs_new_protected:Npn \__pdf_backend_pagesize_gset:nn #1#2
2869 {
2870   \__kernel_backend_first_shipout:n
2871   {
2872     \__kernel_backend_literal:e
2873     {
2874       <*dvipdfmx>
2875         pdf:pagesize ~
2876         width ~ \dim_eval:n {#1} ~
2877         height ~ \dim_eval:n {#2}
2878       </dvipdfmx>
2879       <*dvips>
2880         papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
2881       </dvips>
2882     }
2883   }
2884 }

```

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

```
2885 </dviptfm | dvips>
2886 <*luatex | pdftex | xetex>
```

`_pdf_backend_pagesize_gset:nn` Pass to the primitives.

```
2887 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
2888 {
2889   \dim_gset:Nn \tex_pagewidth:D {#1}
2890   \dim_gset:Nn \tex_pageheight:D {#2}
2891 }
```

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

```
2892 </luatex | pdftex | xetex>
2893 <*dvisvgm>
```

`_pdf_backend_pagesize_gset:nn` A no-op.

```
2894 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2 { }
```

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

```
2895 </dvisvgm>
2896 </package>
```

7 l3backend-pdfannot implementation

```
2897 <*package>
2898 <@@=pdfannot>
```

7.1 dvips backend

```
2899 <*dvips>
```

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

`\l_pdfannot_backend_content_box` The content of an annotation.

```
2900 \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.

```
2901 \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.

```
2902 \int_new:N \g_pdfannot_backend_int
```

(End of definition for `\g_pdfannot_backend_int`.)

`_pdfannot_backend_generic:nmmn`
`_pdfannot_backend_generic_aux:nmmn`

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

```

2903 \cs_new_protected:Npn \_pdfannot_backend_generic:nmmn #1#2#3#4
2904 {
2905   \exp_args:Nf \_pdfannot_backend_generic_aux:nmmn
2906   { \dim_eval:n {#1} } {#2} {#3} {#4}
2907 }
2908 \cs_new_protected:Npn \_pdfannot_backend_generic_aux:nmmn #1#2#3#4
2909 {
2910   \box_move_down:nn {#3}
2911   { \hbox:n { \_kernel_backend_postscript:n { pdfannot.save.ll } } }
2912   \box_move_up:nn {#2}
2913   {
2914     \hbox:n
2915     {
2916       \_kernel_kern:n {#1}
2917       \_kernel_backend_postscript:n { pdfannot.save.ur }
2918       \_kernel_kern:n { -#1 }
2919     }
2920   }
2921   \int_gincr:N \g__pdfannot_backend_int
2922   \_kernel_backend_postscript:e
2923   {
2924     mark
2925     /_objdef { pdfannot \int_use:N \g__pdfannot_backend_int }
2926     pdfannot.rect
2927     #4 ~
2928     /ANN ~
2929     pdfmark
2930   }
2931 }

```

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

`_pdfannot_backend_last:`

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

```

2932 \cs_new:Npn \_pdfannot_backend_last:
2933 { { 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.

```

2934 \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.

```

2935 \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.

```

2936 \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.
2937 `\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.
2938 `\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.
2939 `\tl_new:N \l__pdfannot_backend_breaklink_pdfmark_tl`
2940 `\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.
2941 `\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.
2942 `\cs_new_eq:MN __pdfannot_backend_breaklink_usebox:N \box_use:N`
(End of definition for `__pdfannot_backend_breaklink_usebox:N`.)

`__pdfannot_backend_link_begin_goto:nw` Links are created like annotations but with dedicated code to allow for adjusting the size
`__pdfannot_backend_link_begin_user:nw` of the rectangle. In contrast to `hyperref`, we grab the link content as a box which can
`__pdfannot_backend_link:nw` then unbox: this allows the same interface as for `pdfTeX`.
`__pdfannot_backend_link_aux:nw` Notice that the link setup here uses `/Action` not `/A`. That is because Distiller *requires*
`__pdfannot_backend_link_end:nw` this trigger word, rather than a “raw” PDF dictionary key (Ghostscript can handle either
`__pdfannot_backend_link_end_aux:nw` form).
`__pdfannot_backend_link_minima:nw` Taking the idea of `evenboxes` from `hypdvips`, we implement a minimum box height
`__pdfannot_backend_link_outerbox:nw` and depth for link placement. This means that “underlining” with a hyperlink will
`__pdfannot_backend_link_sf_save:nw` generally give an even appearance. However, to ensure that the full content is always
`__pdfannot_backend_link_sf_restore:nw` 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.

```
2943 \cs_new_protected:Npn \__pdfannot_backend_link_begin_goto:nw #1#2
2944 {
2945   \__pdfannot_backend_link_begin:nw
2946   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2947 }
2948 \cs_new_protected:Npn \__pdfannot_backend_link_begin_user:nw #1#2
2949 { \__pdfannot_backend_link_begin:nw {#1#2} }
2950 \cs_new_protected:Npn \__pdfannot_backend_link_begin:nw #1
2951 {
```

```

2952     \bool_if:NF \g__pdfannot_backend_link_bool
2953     { \__pdfannot_backend_link_begin_aux:nw {#1} }
2954 }

```

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

```

2955 \cs_new_protected:Npn \__pdfannot_backend_link_begin_aux:nw #1
2956 {
2957     \bool_gset_true:N \g__pdfannot_backend_link_bool
2958     \__kernel_backend_postscript:n
2959     { /pdfannot.link.dict ( #1 ) def }
2960     \tl_gset:Nn \g__pdfannot_backend_link_dict_tl {#1}
2961     \__pdfannot_backend_link_sf_save:
2962     \mode_if_math:TF
2963     { \bool_gset_true:N \g__pdfannot_backend_link_math_bool }
2964     { \bool_gset_false:N \g__pdfannot_backend_link_math_bool }
2965     \hbox_set:Nw \l__pdfannot_backend_content_box
2966     \__pdfannot_backend_link_sf_restore:
2967     \bool_if:NT \g__pdfannot_backend_link_math_bool
2968     { \c_math_toggle_token }
2969 }
2970 \cs_new_protected:Npn \__pdfannot_backend_link_end:
2971 {
2972     \bool_if:NT \g__pdfannot_backend_link_bool
2973     { \__pdfannot_backend_link_end_aux: }
2974 }
2975 \cs_new_protected:Npn \__pdfannot_backend_link_end_aux:
2976 {
2977     \bool_if:NT \g__pdfannot_backend_link_math_bool
2978     { \c_math_toggle_token }
2979     \__pdfannot_backend_link_sf_save:
2980     \hbox_set_end:
2981     \__pdfannot_backend_link_minima:
2982     \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
2983     \exp_args:Ne \__pdfannot_backend_link_outerbox:n
2984     {
2985         \int_if_odd:nTF { \value { page } }
2986         { \oddsidemargin }
2987         { \evensidemargin }
2988     }
2989     \box_move_down:nn { \box_dp:N \l__pdfannot_backend_content_box }
2990     { \hbox:n { \__kernel_backend_postscript:n { pdfannot.save.linkll } } }
2991     \__pdfannot_backend_breaklink_postscript:n { pdfannot.bordertracking.begin }
2992     \__pdfannot_backend_breaklink_usebox:N \l__pdfannot_backend_content_box
2993     \__pdfannot_backend_breaklink_postscript:n { pdfannot.bordertracking.end }
2994     \box_move_up:nn { \box_ht:N \l__pdfannot_backend_content_box }
2995     {
2996         \hbox:n
2997         { \__kernel_backend_postscript:n { pdfannot.save.linkur } }
2998     }
2999     \int_gincr:N \g__pdfannot_backend_int
3000     \int_gset_eq:NN \g__pdfannot_backend_link_int \g__pdfannot_backend_int
3001     \__kernel_backend_postscript:e
3002     {

```

```

3003     mark
3004     /_objdef { pdfannot \int_use:N \g__pdfannot_backend_link_int }
3005     \g__pdfannot_backend_link_dict_tl \c_space_tl
3006     pdfannot.rect
3007     /ANN ~ \l__pdfannot_backend_breaklink_pdfmark_tl
3008   }
3009   \__pdfannot_backend_link_sf_restore:
3010   \bool_gset_false:N \g__pdfannot_backend_link_bool
3011 }
3012 \cs_new_protected:Npn \__pdfannot_backend_link_minima:
3013 {
3014   \hbox_set:Nn \l__pdfannot_backend_model_box { Gg }
3015   \__kernel_backend_postscript:e
3016   {
3017     /pdfannot.linkdp.pad ~
3018     \dim_to_decimal:n
3019     {
3020       \dim_max:nn
3021       {
3022         \box_dp:N \l__pdfannot_backend_model_box
3023         - \box_dp:N \l__pdfannot_backend_content_box
3024       }
3025       { Opt }
3026     } ~
3027     pdfannot.pt.dvi ~ def
3028   /pdfannot.linkht.pad ~
3029   \dim_to_decimal:n
3030   {
3031     \dim_max:nn
3032     {
3033       \box_ht:N \l__pdfannot_backend_model_box
3034       - \box_ht:N \l__pdfannot_backend_content_box
3035     }
3036     { Opt }
3037   } ~
3038   pdfannot.pt.dvi ~ def
3039 }
3040 }
3041 \cs_new_protected:Npn \__pdfannot_backend_link_outerbox:n #1
3042 {
3043   \__kernel_backend_postscript:e
3044   {
3045     /pdfannot.outerbox
3046     [
3047       \dim_to_decimal:n {#1} ~
3048       \dim_to_decimal:n { -\box_dp:N \l__pdfannot_backend_model_box } ~
3049       \dim_to_decimal:n { #1 + \textwidth } ~
3050       \dim_to_decimal:n { \box_ht:N \l__pdfannot_backend_model_box }
3051     ]
3052     [ exch { pdfannot.pt.dvi } forall ] def
3053   /pdfannot.baselineskip ~
3054   \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
3055   { pdfannot.pt.dvi ~ def }
3056   { pop ~ pop }

```



```

3057         ifelse
3058     }
3059 }
3060 \cs_new_protected:Npn \__pdfannot_backend_link_sf_save:
3061 {
3062     \int_gset:Nn \g__pdfannot_backend_link_sf_int
3063     {
3064         \mode_if_horizontal:TF
3065         { \tex_spacefactor:D }
3066         { 0 }
3067     }
3068 }
3069 \cs_new_protected:Npn \__pdfannot_backend_link_sf_restore:
3070 {
3071     \mode_if_horizontal:T
3072     {
3073         \int_compare:nNnT \g__pdfannot_backend_link_sf_int > { 0 }
3074         { \int_set:Nn \tex_spacefactor:D \g__pdfannot_backend_link_sf_int }
3075     }
3076 }

```

(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.

```

3077 \use_none:nnn
3078 \cs_if_exist:NT \hook_gput_code:nnn
3079 {
3080     \hook_gput_code:nnn { build/column/after } { backend }
3081     {
3082         \box_if_empty:NF \l_shipout_box
3083         {
3084             \vbox_set:Nn \l_shipout_box
3085             {
3086                 \__kernel_backend_postscript:n
3087                 {
3088                     pdfannot.globaldict /pdfannot.brokenlink.rect ~ known
3089                     { pdfannot.bordertracking.continue }
3090                     if
3091                 }
3092                 \vbox_unpack_drop:N \l_shipout_box
3093                 \__kernel_backend_postscript:n
3094                 { pdfannot.bordertracking.endpage }
3095             }
3096         }
3097     }
3098     \tl_set:Nn \l__pdfannot_backend_breaklink_pdfmark_tl { pdfannot.pdfmark }
3099     \cs_set_eq:NN \__pdfannot_backend_breaklink_postscript:n
3100     \__kernel_backend_postscript:n
3101     \cs_set_eq:NN \__pdfannot_backend_breaklink_usebox:N \hbox_unpack:N
3102 }

```

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

```

3103 \cs_new:Npn \__pdfannot_backend_link_last:
3104 { { pdfannot.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.

```
3105 \cs_new_protected:Npn \_pdfannot_backend_link_margin:n #1
3106   {
3107     \_kernel_backend_postscript:e
3108     {
3109       /pdfannot.linkmargin { \dim_to_decimal:n {#1} ~ pdfannot.pt.dvi } def
3110     }
3111   }
```

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

`_pdfannot_backend_link_on:`

```
\_pdfannot_backend_link_off: 3112 \cs_new_protected:Npn \_pdfannot_backend_link_on: { }
3113 \cs_new_protected:Npn \_pdfannot_backend_link_off: { }
```

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

```
3114 </dvips>
```

7.2 LuaTeX and pdfTeX backend

```
3115 <*luatex | pdftex>
```

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

```
3116 \cs_new_protected:Npn \_pdfannot_backend_generic:nnnn #1#2#3#4
3117   {
3118     <*luatex>
3119     \tex_pdfextension:D annot ~
3120     </luatex>
3121     <*pdftex>
3122     \tex_pdfannot:D
3123     </pdftex>
3124     width ~ \dim_eval:n {#1} ~
3125     height ~ \dim_eval:n {#2} ~
3126     depth ~ \dim_eval:n {#3} ~
3127     {#4}
3128   }
```

(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.

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

(End of definition for `_pdfannot_backend_last:`.)

`_pdfannot_backend_link_begin_goto:nnw`
`_pdfannot_backend_link_begin_user:nnw`
`_pdfannot_backend_link_begin:nnnw`
`_pdfannot_backend_link_end:`

Links are all created using the same internals.

```
3140 \cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nnw #1#2
3141   { \_pdfannot_backend_link_begin:nnnw {#1} { goto~name } {#2} }
3142 \cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nnw #1#2
3143   { \_pdfannot_backend_link_begin:nnnw {#1} { user } {#2} }
3144 \cs_new_protected:Npn \_pdfannot_backend_link_begin:nnnw #1#2#3
3145   {
3146   <*luatex>
3147     \tex_pdfextension:D startlink ~
3148   </luatex>
3149   <*pdftex>
3150     \tex_pdfstartlink:D
3151   </pdftex>
3152     attr {#1}
3153     #2 {#3}
3154   }
3155 \cs_new_protected:Npn \_pdfannot_backend_link_end:
3156   {
3157   <*luatex>
3158     \tex_pdfextension:D endlink \scan_stop:
3159   </luatex>
3160   <*pdftex>
3161     \tex_pdfendlink:D
3162   </pdftex>
3163   }
```

(End of definition for `_pdfannot_backend_link_begin_goto:nnw` and others.)

`_pdfannot_backend_link_last:`

Formatted for direct use.

```
3164 \cs_new:Npe \_pdfannot_backend_link_last:
3165   {
3166     \exp_not:N \int_value:w
3167   <*luatex>
3168     \exp_not:N \tex_pdffeedback:D lastlink ~
3169   </luatex>
3170   <*pdftex>
3171     \exp_not:N \tex_pdflastlink:D
3172   </pdftex>
3173     \c_space_tl 0 ~ R
3174   }
```

(End of definition for `_pdfannot_backend_link_last:`.)

`_pdfannot_backend_link_margin:n`

A simple task: pass the data to the primitive.

```
3175 \cs_new_protected:Npn \_pdfannot_backend_link_margin:n #1
3176   {
3177   <*luatex>
3178     \tex_pdfvariable:D linkmargin
3179   </luatex>
3180   <*pdftex>
3181     \tex_pdflinkmargin:D
3182   </pdftex>
```

```

3183     \dim_eval:n {#1} \scan_stop:
3184   }

```

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

`__pdfannot_backend_link_on:` Separate definitions for the two engines.

```

\__pdfannot_backend_link_off:
3185 \cs_new_protected:Npn \__pdfannot_backend_link_on:
3186 <*luatex>
3187   { \tex_pdfextension:D linkstate 0 ~ }
3188 </luatex>
3189 <*pdfTeX>
3190   { \tex_pdfrunninglinkon:D }
3191 </pdfTeX>
3192 \cs_new_protected:Npn \__pdfannot_backend_link_off:
3193 <*luatex>
3194   { \tex_pdfextension:D linkstate 1 ~ }
3195 </luatex>
3196 <*pdfTeX>
3197   { \tex_pdfrunninglinkoff:D }
3198 </pdfTeX>

```

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

```

3199 </luatex | pdfTeX>

```

7.3 dvipdfmx backend

```

3200 <*dvipdfmx | xetex>

```

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

```

\__pdfannot_backend:e
3201 \cs_new_protected:Npe \__pdfannot_backend:n #1
3202   { \__kernel_backend_literal:n { pdf: #1 } }
3203 \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.

```

3204 \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.

```

3205 \cs_new_protected:Npn \__pdfannot_backend_generic:nnnn #1#2#3#4
3206   {
3207     \int_gincr:N \g__pdfannot_backend_int
3208     \__pdfannot_backend:e
3209     {
3210       ann ~ @pdfannot \int_use:N \g__pdfannot_backend_int \c_space_tl
3211       width ~ \dim_eval:n {#1} ~
3212       height ~ \dim_eval:n {#2} ~
3213       depth ~ \dim_eval:n {#3} ~
3214       << /Type /Annot #4 >>
3215     }
3216   }

```

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

`_pdfannot_backend_last:`

```
3217 \cs_new:Npn \_pdfannot_backend_last:
3218 { @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.

```
3219 \int_new:N \g_pdfannot_backend_link_int
```

(End of definition for `\g_pdfannot_backend_link_int.`)

`_pdfannot_backend_link_begin_goto:nw`

All created using the same internals.

`_pdfannot_backend_link_begin_user:nw`

```
3220 \cs_new_protected:Npn \_pdfannot_backend_link_begin_goto:nw #1#2
```

`_pdfannot_backend_link_begin:n`

```
3221 {
3222   \_pdfannot_backend_link_begin:n
3223   { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> }
3224 }
```

```
3225 \cs_new_protected:Npn \_pdfannot_backend_link_begin_user:nw #1#2
```

```
3226 { \_pdfannot_backend_link_begin:n {#1#2} }
```

```
3227 \cs_new_protected:Npe \_pdfannot_backend_link_begin:n #1
```

```
3228 {
3229   \int_gincr:N \exp_not:N \g_pdfannot_backend_int
3230   \int_gset_eq:NN \exp_not:N \g_pdfannot_backend_link_int
3231   \exp_not:N \g_pdfannot_backend_int
3232   \_pdfannot_backend:e
3233   {
```

```
3234     bann ~
```

```
3235     @pdfannot
```

```
3236     \exp_not:N \int_use:N \exp_not:N \g_pdfannot_backend_link_int
```

```
3237     \c_space_tl
```

```
3238     <<
```

```
3239     /Type /Annot
```

```
3240     #1
```

```
3241     >>
```

```
3242   }
```

```
3243 }
```

```
3244 \cs_new_protected:Npn \_pdfannot_backend_link_end:
```

```
3245 { \_pdfannot_backend:n { eann } }
```

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

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

```
3246 \cs_new:Npn \_pdfannot_backend_link_last:
3247 { @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`.

```
3248 \cs_new_protected:Npn \_pdfannot_backend_link_margin:n #1
3249 { \_kernel_backend_literal:e { dvipdfmx:config~g~ \dim_eval:n {#1} } }
```

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

`_pdfannot_backend_link_on:`

`_pdfannot_backend_link_off:`

```
3250 \cs_new_protected:Npn \_pdfannot_backend_link_on: { \_pdfannot_backend:n { link } }
```

```
3251 \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:.`)
`3252 </dviptdpmx | xetex>`

7.4 dvisvgm backend

`3253 <*dvisvgm>`

`_pdfannot_backend_generic:nnnn`

`3254 \cs_new_protected:Npn _pdfannot_backend_generic:nnnn #1#2#3#4 { }`

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

`_pdfannot_backend_last:`

`3255 \cs_new:Npn _pdfannot_backend_last: { }`

(End of definition for `_pdfannot_backend_last:.`)

`_pdfannot_backend_link_begin_goto:nmw`

`_pdfannot_backend_link_begin_user:nmw`

`_pdfannot_backend_link_begin:nnnw`

`_pdfannot_backend_link_end:`

`3256 \cs_new_protected:Npn _pdfannot_backend_link_begin_goto:nmw #1#2 { }`

`3257 \cs_new_protected:Npn _pdfannot_backend_link_begin_user:nmw #1#2 { }`

`3258 \cs_new_protected:Npn _pdfannot_backend_link_begin:nnnw #1#2#3 { }`

`3259 \cs_new_protected:Npn _pdfannot_backend_link_end: { }`

(End of definition for `_pdfannot_backend_link_begin_goto:nmw` and others.)

`_pdfannot_backend_link_last:`

`3260 \cs_new:Npe _pdfannot_backend_link_last: { }`

(End of definition for `_pdfannot_backend_link_last:.`)

`_pdfannot_backend_link_margin:n`

`3261 \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:`

`3262 \cs_new_protected:Npn _pdfannot_backend_link_on: { }`

`3263 \cs_new_protected:Npn _pdfannot_backend_link_off: { }`

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

`3264 </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.

```
3265 <@@=)
3266 \cs_new_eq:NN \__pdf_backend_annotation:nnnn \__pdfannot_backend_generic:nnnn
3267 \cs_new_eq:NN \__pdf_backend_annotation_last: \__pdfannot_backend_last:
3268 \clist_map_inline:nn
3269 {
3270   begin_goto:nnw ,
3271   begin_user:nnw ,
3272   begin:nnnw ,
3273   end: ,
3274   last: ,
3275   margin:n
3276 }
3277 { \cs_new_eq:cc { __pdf_backend_link_ #1 } { __pdfannot_backend_link_ #1 } }
3278 </package>
```

8 l3backend-opacity implementation

```
3279 <*package>
3280 <@@=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.

```
3281 <*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.

```
3282 \cs_new_protected:Npn \__opacity_backend_select:n #1
3283 {
3284   \__opacity_backend:nnn {#1} { fill } { ca }
3285   \__opacity_backend:nnn {#1} { stroke } { CA }
3286 }
3287 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3288 {
3289   \__opacity_backend:nnn
3290     { #1 }
3291     { fill }
3292     { ca }
3293 }
3294 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3295 {
3296   \__opacity_backend:nnn
3297     { #1 }
```

```

3298     { stroke }
3299     { CA }
3300 }
3301 \cs_new_protected:Npn \__opacity_backend:nnn #1#2#3
3302 {
3303   \__kernel_backend_postscript:n
3304   {
3305     product ~ (Ghostscript) ~ search
3306     {
3307       pop ~ pop ~ pop ~
3308       #1 ~ .set #2 constantalpha
3309     }
3310     {
3311       pop ~
3312       mark ~
3313       /#3 ~ #1
3314       /SetTransparency ~
3315       pdfmark
3316     }
3317     ifelse
3318   }
3319 }

```

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

```

3320 </dvips>
3321 <*dvipdfmx | luatex | pdftex | xetex>

```

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

```

3322 \bool_lazy_and:nnT
3323 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3324 { \pdfmanagement_if_active_p: }
3325 {
3326 <*luatex | pdftex>
3327   \__kernel_color_backend_stack_init:Nnn \c_opacity_backend_stack_int
3328   { page ~ direct } { /opacity 1 ~ gs }
3329 </luatex | pdftex>
3330   \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3331   { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3332 }

```

(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.

```

3333 \tl_new:N \l__opacity_backend_fill_tl
3334 \tl_new:N \l__opacity_backend_stroke_tl
3335 \tl_set:Nn \l__opacity_backend_fill_tl { 1 }
3336 \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: 3337 \cs_new_protected:Npn \__opacity_backend_select:n #1
3338 {

```



```

3339 \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3340 \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3341 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3342 { opacity #1 }
3343 { << /ca ~ #1 /CA ~ #1 >> }
3344 <*/dvipdfmx|xetex>
3345 \__kernel_backend_literal_pdf:n
3346 </dvipdfmx|xetex>
3347 <*/luatex|pdfTeX>
3348 \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3349 </luatex|pdfTeX>
3350 { /opacity #1 ~ gs }
3351 \group_insert_after:N \__opacity_backend_reset:
3352 }
3353 \cs_new_protected:Npn \__opacity_backend_reset:
3354 {
3355 <*/dvipdfmx|xetex>
3356 \__kernel_backend_literal_pdf:n
3357 { /opacity1 ~ gs }
3358 </dvipdfmx|xetex>
3359 <*/luatex|pdfTeX>
3360 \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int
3361 </luatex|pdfTeX>
3362 }

```

(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
3363 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3364 {
3365 \exp_args:Nno \__opacity_backend_fill_stroke:nn
3366 { #1 }
3367 { \l__opacity_backend_stroke_tl }
3368 }
3369 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3370 {
3371 \exp_args:No \__opacity_backend_fill_stroke:nn
3372 { \l__opacity_backend_fill_tl }
3373 { #1 }
3374 }
3375 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3376 {
3377 \str_if_eq:nnTF {#1} {#2}
3378 { \__opacity_backend_select:n {#1} }
3379 {
3380 \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3381 \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3382 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3383 { opacity.fill #1 }
3384 { << /ca ~ #1 >> }
3385 \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3386 { opacity.stroke #2 }
3387 { << /CA ~ #2 >> }

```

```

3388 <*dvipdfmx | xetex>
3389     \__kernel_backend_literal_pdf:n
3390 </dvipdfmx | xetex>
3391 <*luatex | pdftex>
3392     \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3393 </luatex | pdftex>
3394     { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3395     \group_insert_after:N \__opacity_backend_reset:
3396 }
3397 }

```

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

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

```

\__opacity_backend_fill_stroke:nn
3398 \bool_lazy_and:nnF
3399 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3400 { \pdfmanagement_if_active_p: }
3401 {
3402   \cs_gset_protected:Npn \__opacity_backend_select:n #1 { }
3403   \cs_gset_protected:Npn \__opacity_backend_fill_stroke:nn #1#2 { }
3404 }

```

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

```

3405 </dvipdfmx | luatex | pdftex | xetex>

```

```

3406 <*dvisvgm>

```

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

```

\__opacity_backend_fill:n
\__opacity_backend_stroke:n
\__opacity_backend:nn
3407 \cs_new_protected:Npn \__opacity_backend_select:n #1
3408 { \__opacity_backend:nn {#1} { } }
3409 \cs_new_protected:Npn \__opacity_backend_fill:n #1
3410 { \__opacity_backend:nn {#1} { fill- } }
3411 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3412 { \__opacity_backend:nn {#1} { stroke- } }
3413 \cs_new_protected:Npn \__opacity_backend:nn #1#2
3414 { \__kernel_backend_scope:e { #2 opacity = " #1 " } }

```

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

```

3415 </dvisvgm>

```

```

3416 </package>

```

8.1 Font handling integration

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

```

3417 <*lua>

```

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

```

3418 local pdfmanagement_active do
3419   local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3420   local cmd = pdfmanagement_if_active_p.cmdname
3421   if cmd == 'undefined_cs' then

```

```

3422     pdfmanagement_active = false
3423 else
3424     token.put_next(pdfmanagement_if_active_p)
3425     pdfmanagement_active = token.scan_int() ~= 0
3426 end
3427 end
3428
3429 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3430     luaotfload.set_transparent_colorstack(function() return token.create'c_opacity_backend_st
3431
3432     local transparent_register = {
3433         token.create'pdfmanagement_add:nm',
3434         token.new(0, 1),
3435         'Page/Resources/ExtGState',
3436         token.new(0, 2),
3437         token.new(0, 1),
3438         '',
3439         token.new(0, 2),
3440         token.new(0, 1),
3441         '<</ca ',
3442         '',
3443         '/CA ',
3444         '',
3445         '>>',
3446         token.new(0, 2),
3447     }
3448     luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)
3449         value = (octet * -1):match(value)
3450         if not value then
3451             tex.error'Invalid transparency value'
3452             return
3453         end
3454         value = value:sub(1, -2)
3455         local result = 'opacity' .. value
3456         tex.runtoks(function()
3457             transparent_register[6], transparent_register[10], transparent_register[12] = result,
3458             tex.sprint(-2, transparent_register)
3459         end)
3460         return '/' .. result .. ' gs'
3461     end, 'l3opacity')
3462 end
3463 </lua>

```

9 l3backend-header implementation

```

3464 <*dvips & header>

```

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

```

3465 /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 stack.
separation

```

3466 TeXDict begin
3467 /TeXcolorseparation { setcolor } def
3468 end

```

(End of definition for TeXcolorseparation and separation.)

pdf.globaldict A small global dictionary for backend use.

```

3469 true setglobal
3470 /pdf.globaldict 4 dict def
3471 false setglobal

```

(End of definition for pdf.globaldict.)

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

```

pdf.rect.ht 3472 /pdf.cvs { 65534 string cvs } def
3473 /pdf.dvi.pt { 72.27 mul Resolution div } def
3474 /pdf.pt.dvi { 72.27 div Resolution mul } def
3475 /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 3476 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad 3477 /pdf.linkdp.pad { 0 } def
3478 /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
pdf.save.ll separate links for generic annotations to support adding a margin and setting a minimal
pdf.save.ur size.

```

pdf.rect 3479 /pdf.rect
pdf.save.ll 3480 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.save.linkur 3481 /pdf.save.ll
pdf.llx 3482 {
pdf.lly 3483 currentpoint
pdf.urx 3484 /pdf.lly exch def
pdf.ury 3485 /pdf.llx exch def
3486 }
3487 def
3488 /pdf.save.ur
3489 {
3490 currentpoint
3491 /pdf.ury exch def
3492 /pdf.urx exch def
3493 }
3494 def
3495 /pdf.save.linkll
3496 {
3497 currentpoint
3498 pdf.linkmargin add
3499 pdf.linkdp.pad add
3500 /pdf.lly exch def

```

```

3501     pdf.linkmargin sub
3502     /pdf.llx exch def
3503   }
3504   def
3505 /pdf.save.linkur
3506 {
3507   currentpoint
3508   pdf.linkmargin sub
3509   pdf.linkht.pad sub
3510   /pdf.ury exch def
3511   pdf.linkmargin add
3512   /pdf.urx exch def
3513 }
3514 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 3515 /pdf.dest.anchor
pdf.dev.y 3516 {
pdf.tmpa 3517   currentpoint exch
pdf.tmpb 3518   pdf.dvi.pt 72 add
pdf.tmpc 3519   /pdf.dest.x exch def
pdf.tmpd 3520   pdf.dvi.pt
3521   vsize 72 sub exch sub
3522   /pdf.dest.y exch def
3523 }
3524 def
3525 /pdf.dest.point
3526 { pdf.dest.x pdf.dest.y } def
3527 /pdf.dest2device
3528 {
3529   /pdf.dest.y exch def
3530   /pdf.dest.x exch def
3531   matrix currentmatrix
3532   matrix defaultmatrix
3533   matrix invertmatrix
3534   matrix concatmatrix
3535   cvx exec
3536   /pdf.dev.y exch def
3537   /pdf.dev.x exch def
3538   /pdf.tmpd exch def
3539   /pdf.tmpc exch def
3540   /pdf.tmpb exch def
3541   /pdf.tmpa exch def
3542   pdf.dest.x pdf.tmpa mul
3543   pdf.dest.y pdf.tmpc mul add
3544   pdf.dev.x add
3545   pdf.dest.x pdf.tmpb mul
3546   pdf.dest.y pdf.tmpd mul add

```

```

3547     pdf.dev.y add
3548   }
3549   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 3550 /pdf.bordertracking false def
pdf.brokenlink.skip 3551 /pdf.bordertracking.begin
pdf.brokenlink.dict 3552 {
pdf.bordertracking.endpage 3553   SDict /pdf.bordertracking true put
pdf.bordertracking.continue 3554   SDict /pdf.leftboundary undef
pdf.originx 3555   SDict /pdf.rightboundary undef
pdf.originy 3556   /a where
3557     {
3558       /a
3559       {
3560         currentpoint pop
3561         SDict /pdf.rightboundary known dup
3562         {
3563           SDict /pdf.rightboundary get 2 index lt
3564             { not }
3565             if
3566           }
3567         if
3568           { pop }
3569           { SDict exch /pdf.rightboundary exch put }
3570         ifelse
3571         moveto
3572         currentpoint pop
3573         SDict /pdf.leftboundary known dup
3574         {
3575           SDict /pdf.leftboundary get 2 index gt
3576             { not }
3577             if
3578           }
3579         if
3580           { pop }
3581           { SDict exch /pdf.leftboundary exch put }
3582         ifelse
3583       }
3584     put
3585   }
3586   if
3587 }
3588 def
3589 /pdf.bordertracking.end
3590 {
3591   /a where { /a { moveto } put } if
3592   /x where { /x { 0 exch rmoveto } put } if
3593   SDict /pdf.leftboundary known

```

```

3594     { pdf.outerbox 0 pdf.leftboundary put }
3595   if
3596   SDict /pdf.rightboundary known
3597     { pdf.outerbox 2 pdf.rightboundary put }
3598   if
3599   SDict /pdf.bordertracking false put
3600 }
3601 def
3602 /pdf.bordertracking.endpage
3603 {
3604 pdf.bordertracking
3605   {
3606     pdf.bordertracking.end
3607     true setglobal
3608     pdf.globaldict
3609     /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3610     pdf.globaldict
3611     /pdf.brokenlink.skip pdf.baselineskip put
3612     pdf.globaldict
3613     /pdf.brokenlink.dict
3614     pdf.link.dict pdf.cvs put
3615     false setglobal
3616     mark pdf.link.dict cvx exec /Rect
3617     [
3618       pdf.llx
3619       pdf.lly
3620       pdf.outerbox 2 get pdf.linkmargin add
3621       currentpoint exch pop
3622       pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3623     ]
3624     /ANN pdf.pdfmark
3625   }
3626   if
3627 }
3628 def
3629 /pdf.bordertracking.continue
3630 {
3631   /pdf.link.dict pdf.globaldict
3632   /pdf.brokenlink.dict get def
3633   /pdf.outerbox pdf.globaldict
3634   /pdf.brokenlink.rect get def
3635   /pdf.baselineskip pdf.globaldict
3636   /pdf.brokenlink.skip get def
3637   pdf.globaldict dup dup
3638   /pdf.brokenlink.dict undef
3639   /pdf.brokenlink.skip undef
3640   /pdf.brokenlink.rect undef
3641   currentpoint
3642   /pdf.originy exch def
3643   /pdf.originx exch def
3644   /a where
3645     {
3646       /a
3647       {

```

```

3648         moveto
3649         SDict
3650         begin
3651         currentpoint pdf.originy ne exch
3652         pdf.originx ne or
3653         {
3654         pdf.save.linkll
3655         /pdf.lly
3656         pdf.lly pdf.outerbox 1 get sub def
3657         pdf.bordertracking.begin
3658         }
3659         if
3660         end
3661     }
3662     put
3663 }
3664 if
3665 /x where
3666 {
3667     /x
3668     {
3669         0 exch rmoveto
3670         SDict
3671         begin
3672         currentpoint
3673         pdf.originy ne exch pdf.originx ne or
3674         {
3675         pdf.save.linkll
3676         /pdf.lly
3677         pdf.lly pdf.outerbox 1 get sub def
3678         pdf.bordertracking.begin
3679         }
3680         if
3681         end
3682     }
3683     put
3684 }
3685 if
3686 }
3687 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.

```

3688 /pdf.breaklink
3689 {
3690     pop
3691     counttomark 2 mod 0 eq
3692     {
3693         counttomark /pdf.count exch def

```



```

3694 {
3695 pdf.count 0 eq { exit } if
3696 counttomark 2 roll
3697 1 index /Rect eq
3698 {
3699   dup 4 array copy
3700   dup dup
3701     1 get
3702     pdf.outerbox pdf.rect.ht
3703     pdf.linkmargin 2 mul add sub
3704     3 exch put
3705   dup
3706     pdf.outerbox 2 get
3707     pdf.linkmargin add
3708     2 exch put
3709   dup dup
3710     3 get
3711     pdf.outerbox pdf.rect.ht
3712     pdf.linkmargin 2 mul add add
3713     1 exch put
3714   /pdf.currentrect exch def
3715   pdf.breaklink.write
3716   {
3717     pdf.currentrect
3718     dup
3719       pdf.outerbox 0 get
3720       pdf.linkmargin sub
3721       0 exch put
3722     dup
3723       pdf.outerbox 2 get
3724       pdf.linkmargin add
3725       2 exch put
3726     dup dup
3727       1 get
3728       pdf.baselineskip add
3729       1 exch put
3730     dup dup
3731       3 get
3732       pdf.baselineskip add
3733       3 exch put
3734     /pdf.currentrect exch def
3735     pdf.breaklink.write
3736   }
3737   1 index 3 get
3738   pdf.linkmargin 2 mul add
3739   pdf.outerbox pdf.rect.ht add
3740   2 index 1 get sub
3741   pdf.baselineskip div round cvi 1 sub
3742   exch
3743   repeat
3744   pdf.currentrect
3745   dup
3746     pdf.outerbox 0 get
3747     pdf.linkmargin sub

```

```

3748         0 exch put
3749     dup dup
3750         1 get
3751         pdf.baselineskip add
3752         1 exch put
3753     dup dup
3754         3 get
3755         pdf.baselineskip add
3756         3 exch put
3757     dup 2 index 2 get 2 exch put
3758     /pdf.currentrect exch def
3759     pdf.breaklink.write
3760     SDict /pdf.pdfmark.good false put
3761     exit
3762     }
3763     { pdf.count 2 sub /pdf.count exch def }
3764     ifelse
3765     }
3766     loop
3767     }
3768     if
3769     /ANN
3770     }
3771     def
3772 /pdf.breaklink.write
3773     {
3774     counttomark 1 sub
3775     index /_objdef eq
3776     {
3777         counttomark -2 roll
3778         dup wcheck
3779         {
3780             readonly
3781             counttomark 2 roll
3782         }
3783         { pop pop }
3784         ifelse
3785     }
3786     if
3787     counttomark 1 add copy
3788     pop pdf.currentrect
3789     /ANN pdfmark
3790     }
3791     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.

```

3792 /pdf.pdfmark
3793     {

```

```

3794 SDict /pdf.pdfmark.good true put
3795 dup /ANN eq
3796 {
3797   pdf.pdfmark.store
3798   pdf.pdfmark.dict
3799   begin
3800     Subtype /Link eq
3801     currentdict /Rect known and
3802     SDict /pdf.outerbox known and
3803     SDict /pdf.baselineskip known and
3804     {
3805       Rect 3 get
3806       pdf.linkmargin 2 mul add
3807       pdf.outerbox pdf.rect.ht add
3808       Rect 1 get sub
3809       pdf.baselineskip div round cvi 0 gt
3810       { pdf.breaklink }
3811       if
3812     }
3813     if
3814     end
3815     SDict /pdf.outerbox undef
3816     SDict /pdf.baselineskip undef
3817     currentdict /pdf.pdfmark.dict undef
3818   }
3819   if
3820   pdf.pdfmark.good
3821   { pdfmark }
3822   { cleartomark }
3823   ifelse
3824 }
3825 def
3826 /pdf.pdfmark.store
3827 {
3828   /pdf.pdfmark.dict 65534 dict def
3829   counttomark 1 add copy
3830   pop
3831   {
3832     dup mark eq
3833     {
3834       pop
3835       exit
3836     }
3837     {
3838       pdf.pdfmark.dict
3839       begin def end
3840     }
3841     ifelse
3842   }
3843   loop
3844 }
3845 def

```

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

3846 </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, 2964, 3010
<code>\bool_gset_true:N</code>	1221, 1290, 1410, 1679, 2957, 2963
<code>\bool_if:NTF</code>	66, 596, 1233, 1237, 1253, 1256, 1260, 1271, 1278, 1282, 1294, 1298, 1423, 1428, 1433, 1638, 1683, 1830, 1880, 2020, 2062, 2952, 2967, 2972, 2977
<code>\bool_if:nTF</code>	2455, 2636, 2824
<code>\bool_lazy_and:nnTF</code>	809, 2179, 3322, 3398
<code>\bool_lazy_any:nTF</code>	1869
<code>\bool_lazy_or:nnTF</code>	2055
<code>\bool_new:N</code>	1224, 1291, 1413, 1680, 2937, 2938
<code>\bool_set_false:N</code>	1842, 1984, 2086, 2250
box commands:	
<code>\box_dp:N</code>	235, 237, 285, 287, 342, 344, 391, 393, 395, 397, 2989, 3022, 3023, 3048
<code>\box_ht:N</code>	237, 287, 344, 395, 397, 1893, 2127, 2994, 3033, 3034, 3050
<code>\box_if_empty:N</code>	3082
<code>\box_move_down:nn</code>	2910, 2989
<code>\box_move_up:nn</code>	2271, 2912, 2994
<code>\box_new:N</code>	2900, 2901
<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, 2942
<code>\box_wd:N</code>	236, 244, 286, 292, 343, 349, 392, 394, 1892, 2126
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>	3268
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

<code>\l_color_backend_fill_tl</code>	<code>_color_backend_separation_</code>
..... 525 , 537 , 983 , 997	<code>init_aux:nnnnnn</code> 594 , 600 , 616
<code>_color_backend_iccbased_</code>	<code>_color_backend_separation_</code>
<code>device:mnn</code> 932 , 932	<code>init_CIELAB:mnn</code>
<code>_color_backend_iccbased_</code> 594 , 706 , 776 , 823 , 848
<code>init:nnn</code>	<code>_color_backend_separation_</code>
..... 771 , 771 , 914 , 914 , 1095 , 1096	<code>init_CIELAB:nnnnnn</code> 777
<code>_color_backend_init_resource:n</code>	<code>_color_backend_separation_</code>
..... 806 , 806 , 835 , 906 , 930 , 945	<code>init_count:n</code> 594 , 653 , 656
<code>_color_backend_reset:</code>	<code>_color_backend_separation_</code>
.... 506 , 521 , 529 , 541 , 545 , 550 ,	<code>init_count:w</code> ... 594 , 657 , 658 , 662
971 , 972 , 1005 , 1006 , 1031 , 1049 , 1093	<code>_color_backend_separation_</code>
<code>_color_backend_rgb:w</code> 1062	<code>init_Device:Nn</code>
<code>_color_backend_select:n</code> 594 , 638 , 640 , 642 , 643
..... 506 , 507 , 509 , 511 ,	<code>\l_color_backend_stack_int</code>
513 , 514 , 545 , 545 , 547 , 548 , 549 , 591 467 , 542 , 984 , 996
<code>_color_backend_select:mn</code>	<code>_color_backend_stroke:n</code>
..... 529 , 530 , 532 , 534 , 535 , 802 949 , 954 , 956 ,
<code>_color_backend_select_cmyk:n</code> ..	957 , 958 , 975 , 988 , 990 , 992 , 993 , 1002
..... 506 , 506 , 529 , 529 , 545 , 547	<code>_color_backend_stroke_cmyk:n</code> ..
<code>_color_backend_select_devicen:mnn</code> 949 ,
..... 590 , 592 , 774 , 775 , 796 , 804	956 , 975 , 987 , 1009 , 1019 , 1045 , 1045
<code>_color_backend_select_gray:n</code> ..	<code>_color_backend_stroke_devicen:mnn</code>
.... 506 , 508 , 529 , 531 , 545 , 548 , 555 959 ,
<code>_color_backend_select_iccbased:mnn</code>	970 , 999 , 1004 , 1025 , 1030 , 1089 , 1092
..... 593 , 593 , 778 , 778 , 796 , 805	<code>_color_backend_stroke_gray:n</code> ..
<code>_color_backend_select_named:n</code> 949 ,
..... 506 , 510 , 552 , 552	957 , 975 , 989 , 1009 , 1021 , 1045 , 1051
<code>_color_backend_select_rgb:n</code> ...	<code>_color_backend_stroke_gray_</code>
..... 506 , 512 , 529 , 533 , 545 , 549	<code>aux:n</code> 1045 , 1055 , 1059
<code>_color_backend_select_separation:mnn</code>	<code>_color_backend_stroke_reset:</code> ..
..... 590 , 590 , 592 , 971 ,
774 , 774 , 775 , 796 , 797 , 801 , 804 , 805	972 , 1005 , 1006 , 1031 , 1032 , 1093 , 1094
<code>_color_backend_separation_</code>	<code>_color_backend_stroke_rgb:n</code> ...
<code>init:n</code> 594 , 675 , 688 949 ,
<code>_color_backend_separation_</code>	958 , 975 , 991 , 1009 , 1023 , 1045 , 1061
<code>init:nn</code> 823 , 833 , 837	<code>_color_backend_stroke_rgb:w</code> ...
<code>_color_backend_separation_</code> 1045 , 1063
<code>init:nnn</code> 594 , 629 , 650	<code>_color_backend_stroke_separation:mnn</code>
<code>_color_backend_separation_</code>	.. 959 , 964 , 970 , 999 , 1001 , 1004 ,
<code>init:nnnn</code> 594 , 652 , 664	1025 , 1027 , 1030 , 1089 , 1090 , 1092
<code>_color_backend_separation_</code>	<code>\l_color_backend_stroke_tl</code>
<code>init:nnnnn</code> 594 , 525 , 538 , 985 , 995
594 , 615 , 708 , 776 , 776 , 823 , 823 , 863	<code>\g_color_model_int</code> 601 , 610 , 758 ,
<code>_color_backend_separation_</code>	786 , 835 , 841 , 842 , 896 , 897 , 906 , 930
<code>init:nw</code> 594 , 679 , 690 , 704	<code>\c_color_model_range_CIELAB_tl</code> .
<code>_color_backend_separation_</code> 713 , 748 , 859 , 866
<code>init:w</code> 594 , 666 , 681 , 686	<code>color.sc</code> 3465
<code>_color_backend_separation_</code>	<code>cs commands:</code>
<code>init_/DeviceCMYK:nnn</code> 594	<code>\cs_generate_variant:Nn</code> .. 62 , 65 ,
<code>_color_backend_separation_</code>	170 , 181 , 212 , 218 , 615 , 1169 , 1580 ,
<code>init_/DeviceGray:nnn</code> 594	2034 , 2097 , 2117 , 2299 , 2314 , 2377 ,
<code>_color_backend_separation_</code>	2587 , 2600 , 2710 , 2725 , 2755 , 3203
<code>init_/DeviceRGB:nnn</code> 594	<code>\cs_gset:Npe</code> .. 2467 , 2471 , 2829 , 2834

<code>\cs_gset_protected:Npn</code> . . .	3402, 3403	1430, 1435, 1437, 1450, 1455, 1457,
<code>\cs_if_exist:NTF</code>		1459, 1461, 1463, 1465, 1467, 1469,
.	27, 49, 1782, 2661, 2687, 3078	1488, 1512, 1518, 1530, 1542, 1554,
<code>\cs_if_exist_p:N</code>	810, 3323, 3399	1561, 1583, 1589, 1594, 1599, 1610,
<code>\cs_if_exist_use:NTF</code>	38, 628	1620, 1630, 1632, 1634, 1636, 1667,
<code>\cs_new:Npe</code>		1669, 1674, 1676, 1678, 1681, 1702,
563, 2601, 2612, 2679, 3129, 3164, 3260		1713, 1726, 1728, 1730, 1732, 1734,
<code>\cs_new:Npn</code>	578, 637, 639,	1736, 1738, 1740, 1742, 1750, 1758,
641, 643, 650, 656, 658, 664, 681,		1780, 1799, 1822, 1839, 1853, 1858,
688, 690, 908, 1316, 1448, 1711,		1866, 1896, 1909, 1927, 1937, 1953,
1895, 2130, 2288, 2306, 2378, 2380,		1972, 1981, 1989, 2001, 2007, 2010,
2473, 2474, 2556, 2557, 2569, 2588,		2025, 2035, 2074, 2083, 2089, 2095,
2589, 2692, 2718, 2756, 2758, 2837,		2098, 2105, 2118, 2123, 2131, 2138,
2838, 2850, 2851, 2856, 2857, 2862,		2155, 2189, 2220, 2221, 2223, 2225,
2863, 2932, 3103, 3217, 3246, 3255		2227, 2233, 2239, 2247, 2253, 2256,
<code>\cs_new_eq:NN</code>	46, 56, 58, 547,	2258, 2269, 2297, 2300, 2302, 2304,
548, 549, 592, 775, 804, 805, 951,		2308, 2315, 2332, 2337, 2342, 2347,
952, 953, 956, 957, 958, 969, 970,		2357, 2362, 2370, 2382, 2408, 2413,
971, 972, 1003, 1004, 1005, 1006,		2441, 2453, 2465, 2469, 2475, 2477,
1029, 1030, 1031, 1091, 1092, 1093,		2481, 2504, 2518, 2528, 2539, 2558,
1168, 1372, 1373, 1378, 1379, 1579,		2590, 2623, 2634, 2640, 2668, 2702,
1581, 1582, 1588, 1796, 1797, 1810,		2704, 2711, 2713, 2716, 2720, 2726,
1812, 1837, 1838, 1901, 1902, 1903,		2731, 2736, 2738, 2740, 2748, 2760,
1926, 1951, 1968, 1969, 1978, 1979,		2782, 2787, 2820, 2822, 2827, 2832,
1980, 2000, 2003, 2004, 2005, 2070,		2839, 2841, 2845, 2846, 2847, 2848,
2080, 2081, 2082, 2236, 2237, 2245,		2849, 2852, 2853, 2854, 2855, 2858,
2246, 2255, 2285, 2286, 2287, 2291,		2859, 2860, 2861, 2864, 2865, 2868,
2307, 2719, 2942, 3266, 3267, 3277		2887, 2894, 2903, 2908, 2941, 2943,
<code>\cs_new_protected:Npe</code>		2948, 2950, 2955, 2970, 2975, 3012,
. . .	594, 1074, 2651, 2708, 3201, 3227	3041, 3060, 3069, 3105, 3112, 3113,
<code>\cs_new_protected:Npn</code> 47, 53, 60, 63,		3116, 3140, 3142, 3144, 3155, 3175,
71, 77, 82, 84, 88, 98, 108, 118, 128,		3185, 3192, 3205, 3220, 3225, 3244,
137, 146, 156, 168, 171, 173, 175,		3248, 3250, 3251, 3254, 3256, 3257,
179, 184, 193, 203, 213, 224, 246,		3258, 3259, 3261, 3262, 3263, 3282,
248, 263, 279, 294, 296, 322, 336,		3287, 3294, 3301, 3337, 3353, 3363,
351, 353, 366, 380, 430, 443, 470,		3369, 3375, 3407, 3409, 3411, 3413
484, 494, 506, 508, 510, 512, 514,		<code>\cs_set_eq:NN</code>
521, 529, 531, 533, 535, 541, 545,		3099, 3101
550, 552, 590, 593, 616, 706, 752,		<code>\cs_set_protected:Npn</code>
771, 774, 776, 777, 778, 797, 801,		2193
806, 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, 1414, 1416, 1418, 1420, 1425,		

D

dim commands:

<code>\dim_compare:nNnTF</code>	2169, 2174
<code>\dim_compare_p:nNn</code>	2180, 2181
<code>\dim_eval:n</code>	
. . .	2411, 2514, 2515, 2516, 2785,
2876, 2877, 2880, 2906, 3124, 3125,	
3126, 3183, 3211, 3212, 3213, 3249	
<code>\dim_gset:Nn</code>	2889, 2890
<code>\dim_max:nn</code>	3020, 3031
<code>\dim_set:Nn</code>	
. .	1892, 1893, 2126, 2127, 2165, 2166
<code>\dim_set_eq:NN</code>	2231
<code>\dim_to_decimal:n</code> . .	391, 392, 393,
394, 395, 397, 1592, 1597, 1603,	

1604, 1605, 1606, 1615, 1616, 1617,
 1708, 1727, 2278, 2279, 3018, 3029,
 3047, 3048, 3049, 3050, 3054, 3109
 \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, 1804, 1805, 1806, 1807,
 1994, 1995, 1996, 1997, 2049, 2050,
 2051, 2052, 2263, 2264, 2265, 2266
 \dim_zero:N 2163, 2164
 \c_max_dim
 .. 2165, 2166, 2169, 2174, 2180, 2181
 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:nnnnn ..
 .. 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, 1548, 1549, 1550, 2401, 2501, 2778
<code>decompose:nmmN</code>	1482, 1517 , 1518
<code>__draw_backend_transform_-</code>	
<code>decompose_auxi:nmmN</code>
.....	1517 , 1522, 1530
<code>__draw_backend_transform_-</code>	
<code>decompose_auxii:nmmN</code>
.....	1517 , 1534, 1542
<code>__draw_backend_transform_-</code>	
<code>decompose_auxiii:nmmN</code>
.....	1517 , 1546, 1554
<code>\g__draw_draw_clip_bool</code>	.. 1225 , 1634
<code>\g__draw_draw_eor_bool</code>
...	1220 , 1237 , 1253 , 1260 , 1271 , 1282 , 1298 , 1409 , 1423 , 1428 , 1433
<code>\g__draw_draw_path_int</code> 1634
E	
<code>\errmessage</code> 38
<code>\evensidemargin</code> 2987
exp commands:	
<code>\exp_after:wN</code> 2136
<code>\exp_args:Ne</code> 598, 652, 833, 1860, 1915, 1917, 1941, 1943, 2344, 2359, 2410, 2784, 2983
<code>\exp_args:Nf</code> 1310, 1442, 2905
<code>\exp_args:Nne</code> 2751
<code>\exp_args:NNf</code> 247, 295, 352
<code>\exp_args:Nno</code> 3365
<code>\exp_args:No</code> 3371
<code>\exp_not:N</code> 565, 571, 572, 573, 598, 600, 601, 604, 605, 610, 2603, 2605, 2608, 2614, 2616, 2619, 2656, 2657, 2663, 2664, 2683, 2688, 3131, 3133, 3136, 3166, 3168, 3171, 3229, 3230, 3231, 3236
<code>\exp_not:n</code> 48, 96, 116, 154, 922, 2335, 2340, 2404, 2573, 2574, 2588, 2589, 2729, 2734, 2745, 2764
<code>\ExplBackendFileDate</code> 1
F	
file commands:	
<code>\file_compare_timestamp:nNnTF</code>	.. 1929
<code>\file_parse_full_name:nNNN</code>	1911, 1939
<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,
<code>\fp_new:N</code> 320, 321
<code>\fp_set:Nn</code> 300, 303
<code>\fp_use:N</code> 306, 310, 315
<code>\fp_zero:N</code> 302
<code>\c_zero_fp</code>	254, 301, 307, 359, 1493, 1506
G	
graphics commands:	
<code>\l_graphics_search_ext_seq</code> 1792 , 1815 , 1961 , 2149
graphics internal commands:	
<code>\l_graphics_attr_tl</code> 1821 , 1826, 1843, 1855, 1862, 1864, 1899
<code>__graphics_backend_dequote:w</code> 1822 , 1861, 1895
<code>\l_graphics_backend_dir_str</code>	.. 1904
<code>\l_graphics_backend_ext_str</code>	.. 1904
<code>__graphics_backend_get_pagecount:n</code> 1811 , 1812, 1953 , 1953, 2068 , 2070, 2138 , 2138, 2290 , 2291
<code>__graphics_backend_getbb_auxi:n</code> 1822 , 1835, 1851, 1853
<code>__graphics_backend_getbb_-</code>	
<code>auxi:nN</code>	... 2074 , 2078, 2087, 2089
<code>__graphics_backend_getbb_-</code>	
<code>auxii:n</code> 1822 , 1856, 1858
<code>__graphics_backend_getbb_-</code>	
<code>auxiii:nmmN</code>	.. 2074 , 2092, 2095, 2097
<code>__graphics_backend_getbb_-</code>	
<code>auxiiii:n</code> 1822 , 1860, 1866
<code>__graphics_backend_getbb_-</code>	
<code>auxiii:nNnn</code>	.. 2074 , 2093, 2096, 2098
<code>__graphics_backend_getbb_-</code>	
<code>auxiv:nmNnn</code>	.. 2074 , 2101, 2105, 2117
<code>__graphics_backend_getbb_-</code>	
<code>auxv:nmNnn</code>	.. 2074 , 2102, 2109, 2118
<code>__graphics_backend_getbb_-</code>	
<code>auxvi:nmNnn</code> 2121 , 2123
<code>__graphics_backend_getbb_bmp:n</code> 1966 , 1980, 2074 , 2082
<code>__graphics_backend_getbb_eps:n</code> 1794 , 1796 , 1904 , 1909 , 1926, 1966 , 1968, 2234 , 2236
<code>__graphics_backend_getbb_eps:nm</code> 1904
<code>__graphics_backend_getbb_eps:nm</code> 1915, 1927
<code>__graphics_backend_getbb_jpeg:n</code> 1822 , 1837, 1966 , 1978, 2074 , 2080, 2239 , 2245
<code>__graphics_backend_getbb_jpg:n</code>	.. 1822 , 1822, 1837, 1838, 1966 , 1972,

1978, 1979, 1980, [2074](#), [2074](#), 2080,
 2081, 2082, [2239](#), [2239](#), [2245](#), [2246](#)
 _graphics_backend_getbb_
 pagebox:w .. [2074](#), [2113](#), [2130](#), [2136](#)
 _graphics_backend_getbb_pdf:n .
 [1822](#), [1839](#), [1935](#),
 [1966](#), [1981](#), [2074](#), [2083](#), [2247](#), [2247](#)
 _graphics_backend_getbb_png:n .
 [1822](#), [1838](#),
 [1966](#), [1979](#), [2074](#), [2081](#), [2239](#), [2246](#)
 _graphics_backend_getbb_ps:n ..
 [1794](#), [1797](#),
 [1904](#), [1926](#), [1966](#), [1969](#), [2234](#), [2237](#)
 _graphics_backend_getbb_svg:n .
 [2155](#), [2155](#)
 _graphics_backend_getbb_svg_
 auxi:nNn ... [2155](#), [2171](#), [2176](#), [2189](#)
 _graphics_backend_getbb_svg_
 auxii:w [2155](#), [2193](#), [2215](#), [2220](#)
 _graphics_backend_getbb_svg_
 auxiii:Nw [2155](#), [2203](#), [2221](#)
 _graphics_backend_getbb_svg_
 auxiv:Nw [2155](#), [2206](#), [2223](#)
 _graphics_backend_getbb_svg_
 auxv:Nw [2155](#), [2207](#), [2225](#)
 _graphics_backend_getbb_svg_
 auxvi:Nn [2155](#), [2222](#), [2224](#), [2226](#), [2227](#)
 _graphics_backend_getbb_svg_
 auxvii:w [2155](#), [2229](#), [2233](#)
 _graphics_backend_include:nn ..
 [2253](#), [2254](#), [2257](#), [2258](#)
 _graphics_backend_include_
 auxi:nn [1989](#), [2002](#), [2008](#), [2010](#)
 _graphics_backend_include_
 auxii:nnn .. [1989](#), [2012](#), [2025](#), [2034](#)
 _graphics_backend_include_
 auxiii:nnn [1989](#), [2032](#), [2035](#)
 _graphics_backend_include_
 bmp:n [1989](#), [2005](#)
 _graphics_backend_include_
 dequote:w [2269](#), [2280](#), [2288](#)
 _graphics_backend_include_
 eps:n [1799](#),
 [1799](#), [1810](#), [1904](#), [1937](#), [1951](#),
 [1989](#), [1989](#), [2000](#), [2253](#), [2253](#), [2255](#)
 _graphics_backend_include_
 jpeg:n . [1896](#), [1901](#), [2003](#), [2269](#), [2286](#)
 _graphics_backend_include_
 jpg:n [1896](#),
 [1896](#), [1901](#), [1902](#), [1903](#), [1989](#),
 [2001](#), [2003](#), [2004](#), [2005](#), [2269](#), [2287](#)
 _graphics_backend_include_
 jpspeg:n [1989](#)
 _graphics_backend_include_
 pdf:n [1896](#), [1902](#), [1941](#),
 [1989](#), [2007](#), [2131](#), [2131](#), [2253](#), [2256](#)
 _graphics_backend_include_
 png:n
 .. [1896](#), [1903](#), [1989](#), [2004](#), [2269](#), [2285](#)
 _graphics_backend_include_ps:n
 [1799](#), [1810](#),
 [1904](#), [1951](#), [1989](#), [2000](#), [2253](#), [2255](#)
 _graphics_backend_include_
 svg:n .. [2269](#), [2269](#), [2285](#), [2286](#), [2287](#)
 _graphics_backend_loaded:n ...
 [1780](#), [1780](#), [1792](#), [1794](#), [1811](#), [1815](#),
 [1961](#), [1966](#), [2069](#), [2149](#), [2234](#), [2290](#)
 \l_graphics_backend_name_str . [1904](#)
 _graphics_bb_restore:nTF
 [1855](#), [2120](#), [2157](#)
 _graphics_bb_save:n [1864](#), [2128](#), [2184](#)
 \l_graphics_decodearray_str ...
 [1828](#), [1829](#),
 [1841](#), [1872](#), [1878](#), [1879](#), [1983](#), [2018](#),
 [2019](#), [2057](#), [2060](#), [2061](#), [2085](#), [2249](#)
 _graphics_extract_bb:n
 [1976](#), [1985](#), [2243](#), [2251](#)
 \l_graphics_final_name_str .. [1934](#)
 _graphics_get_pagecount:n
 [1812](#), [2070](#), [2291](#)
 \l_graphics_internal_box
 .. [1890](#), [1892](#), [1893](#), [2125](#), [2126](#), [2127](#)
 \l_graphics_internal_dim [2230](#), [2231](#)
 \l_graphics_internal_ior
 [2159](#), [2160](#), [2167](#), [2186](#)
 \l_graphics_interpolate_bool ...
 [1830](#), [1842](#), [1871](#), [1880](#),
 [1984](#), [2020](#), [2056](#), [2062](#), [2086](#), [2250](#)
 \l_graphics_llx_dim
 [1804](#), [1994](#), [2049](#), [2163](#), [2263](#)
 \l_graphics_lly_dim
 [1805](#), [1995](#), [2050](#), [2164](#), [2264](#)
 \l_graphics_page_int
 [1824](#), [1846](#), [1847](#), [1885](#),
 [1886](#), [1974](#), [2016](#), [2017](#), [2043](#), [2044](#),
 [2076](#), [2091](#), [2092](#), [2134](#), [2135](#), [2241](#)
 \l_graphics_pagebox_tl
 [56](#), [1825](#), [1845](#),
 [1887](#), [1888](#), [1975](#), [2014](#), [2015](#), [2045](#),
 [2047](#), [2077](#), [2100](#), [2101](#), [2136](#), [2242](#)
 \l_graphics_pdf_str
 .. [1832](#), [1833](#), [1848](#), [1849](#), [1873](#), [1882](#)
 _graphics_read_bb:n
 .. [1796](#), [1797](#), [1968](#), [1969](#), [2236](#), [2237](#)
 \g_graphics_track_int
 [1988](#), [2037](#), [2038](#)

\l__graphics_urx_dim	2374, 2379, 2752, 2757, 2925, 2933, 3004, 3104, 3210, 3218, 3236, 3247
...	1806, 1892, 1996, 2051, 2126, 2165, 2169, 2172, 2180, 2265, 2278
\l__graphics_ury_dim	2603, 2614, 2632, 3131, 3166
...	1807, 1893, 1997, 2052, 2127, 2166, 2174, 2177, 2181, 2266, 2271, 2279
group commands:	
\group_begin:	190, 209
\group_end:	198
\group_insert_after:N	3351, 3395
\int_value:w	
\int_zero:N	1824, 1974, 2076, 2241
ior commands:	
\ior_close:N	2186
\ior_if_eof:NTF	2160
\ior_map_break:	2182
\ior_open:Nn	2159
\ior_str_map_inline:Nn	2167

H

hbox commands:

\hbox:n	2273, 2418, 2425, 2792, 2803, 2911, 2914, 2990, 2996
\hbox_overlap_right:n	242, 274, 290, 331, 347, 375, 459, 1363, 1571
\hbox_set:Nn	1890, 2125, 2982, 3014
\hbox_set:Nw	2965
\hbox_set_end:	2980
\hbox_unpack:N	3101

hook commands:

\hook_gput_code:nmn	54, 1782, 1784, 3078, 3080
---------------------	----------------------------

I

int commands:

\int_compare:nNnTF	1846, 1885, 2016, 2043, 2091, 2134, 2443, 2654, 2682, 3073
\int_const:Nn	472, 1862, 1956, 2038, 2140
\int_eval:n	492, 502, 648, 657, 670, 672, 676, 689, 2467, 2471, 2632, 2657, 2664, 2677, 2821, 2829, 2834
\int_gincr:N	216, 382, 1640, 1685, 2037, 2305, 2372, 2717, 2750, 2921, 2999, 3207, 3229
\int_gset:Nn	191, 210, 2548, 3062
\int_gset_eq:NN	199, 3000, 3230
\int_if_exist:NTF	2027
\int_if_odd:nTF	2985
\int_max:nn	2142
\int_new:N	182, 183, 429, 467, 1666, 1988, 2902, 2934, 2936, 3204, 3219
\int_set:Nn	3074
\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, 1847, 1886, 1899, 1957, 2017, 2030, 2042, 2044, 2135, 2143,

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, 2870
\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, 1801, 1991, 2029, 2039, 2260, 2275, 2709, 2821, 2825, 2830, 2835, 2872, 3202, 3249
__kernel_backend_literal_page:n	108, 108, 118, 171, 171, 2703, 2705, 2840, 2842
__kernel_backend_literal_pdf:n	88, 88, 98, 168, 168, 170, 282, 339, 1372, 1373, 3345, 3356, 3389
__kernel_backend_literal_- postscript:n	60, 60, 62, 74, 75, 79, 228, 229, 231, 232, 240, 252, 267, 1168, 2445, 2457
__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, 2298, 2349, 2364, 2384, 2418, 2425, 2911, 2917, 2922, 2958, 2990, 2997, 3001, 3015, 3043, 3086, 3093, 3100, 3107, 3303
__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, 3414	
<code>__kernel_backend_scope_begin:</code> ..	
82, 82, 128, 128, 173, 173, 184, 184, 226, 250, 265, 281, 298, 324, 338, 355, 368, 1378, 1563, 1581, 1585, 1760	
<code>__kernel_backend_scope_begin:n</code> ..	
..... 184, 203, 212, 404, 432, 445	
<code>__kernel_backend_scope_end:</code> ..	
..... 82, 84, 128, 137, 173, 175, 184, 193, 243, 261, 275, 291, 318, 332, 348, 364, 376, 427, 441, 460, 1379, 1575, 1582, 1588, 1774	
<code>\g__kernel_backend_scope_int</code> ..	
..... 182, 189, 191, 196, 200, 208, 210, 216	
<code>\l__kernel_backend_scope_int</code> ..	
..... 182, 188, 201, 207	
<code>\g__kernel_clip_path_int</code> ..	
..... 380, 1640, 1643, 1656, 1685, 1688, 1696	
<code>__kernel_color_backend_stack_</code>	
<code>init:Nnn</code> ..	470, 470, 3327
<code>__kernel_color_backend_stack_</code>	
<code>pop:n</code> ..	484, 494, 542, 3360
<code>__kernel_color_backend_stack_</code>	
<code>push:nn</code> ..	484, 484, 539, 984, 996, 3348, 3392
<code>__kernel_dependency_version_</code>	
<code>check:Nn</code> ..	1
<code>__kernel_dependency_version_</code>	
<code>check:nn</code> ..	27, 29
<code>__kernel_file_name_quote:n</code> ..	1917, 1943
<code>__kernel_kern:n</code> ..	2417, 2421, 2424, 2428, 2791, 2799, 2802, 2818, 2916, 2918
	L
lua commands:	
<code>\lua_load_module:n</code> ..	1162
	M
<code>\MessageBreak</code> ..	40
mode commands:	
<code>\mode_if_horizontal:TF</code> ..	3064, 3071
<code>\mode_if_math:TF</code> ..	2962
msg commands:	
<code>\msg_error:nnn</code> ..	556, 2161
<code>\msg_new:nnn</code> ..	558
	O
<code>\oddsidemargin</code> ..	2986
opacity internal commands:	
<code>__opacity_backend:nn</code> ..	3407, 3408, 3410, 3412, 3413
<code>__opacity_backend:nnn</code> ..	3282, 3284, 3285, 3289, 3296, 3301
<code>__opacity_backend_fill:n</code> ..	3282, 3287, 3363, 3363, 3407, 3409
<code>__opacity_backend_fill_stroke:nn</code> ..	3363, 3365, 3371, 3375, 3398, 3403
<code>\l__opacity_backend_fill_tl</code> ..	3333, 3339, 3372, 3380
<code>__opacity_backend_reset:</code> ..	3337, 3351, 3353, 3395
<code>__opacity_backend_select:n</code> ..	3282, 3282, 3337, 3337, 3378, 3398, 3402, 3407, 3407
<code>\c__opacity_backend_stack_int</code> ..	3322, 3348, 3360, 3392
<code>__opacity_backend_stroke:n</code> ..	3282, 3294, 3363, 3369, 3407, 3411
<code>\l__opacity_backend_stroke_tl</code> ..	3333, 3340, 3367, 3381
	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> ..	2708, 2708, 2710, 2712, 2714, 2728, 2733, 2742, 2762, 2794, 2795, 2805
<code>__pdf_backend_annotation:nnnn</code> ..	3266
<code>__pdf_backend_annotation_last:</code> ..	3267
<code>__pdf_backend_bdc:nn</code> ..	2475, 2475, 2702, 2702, 2839, 2839, 2864, 2864
<code>__pdf_backend_catalog_gput:nn</code> ..	2300, 2300, 2518, 2518, 2711, 2711, 2847, 2847
<code>__pdf_backend_compress_objects:n</code> ..	2441, 2453, 2623, 2634, 2820, 2822, 2858, 2859
<code>__pdf_backend_compresslevel:n</code> ..	2441, 2441, 2623, 2623, 2820, 2820, 2858, 2858
<code>__pdf_backend_destination:nn</code> ..	2382, 2382, 2481, 2481, 2760, 2760, 2845, 2845

_pdf_backend_destination:nnnn .	_pdf_backend_pageobject_ref:n .
. 2382 , 2408 , 2380 , 2380 ,
2481 , 2504 , 2760 , 2782 , 2845 , 2846	2612 , 2612 , 2758 , 2758 , 2849 , 2857
_pdf_backend_destination_-	_pdf_backend_pagesize_gset:nn .
aux:nnnn 2868 , 2868 , 2887 , 2887 , 2894 , 2894
. 2382 , 2410 , 2413 , 2760 , 2784 , 2787	_pdf_backend_pdfmark:n
_pdf_backend_emc:	2297 , 2297 , 2299 , 2301 , 2303 , 2317 ,
2702 , 2704 , 2839 , 2841 , 2864 , 2865	2334 , 2339 , 2385 , 2429 , 2476 , 2478
_pdf_backend_info_gput:nn	_pdf_backend_version_major:
. 2300 , 2302 , 2467 , 2473 , 2473 , 2679 , 2679 ,
2518 , 2528 , 2711 , 2713 , 2847 , 2848	2829 , 2830 , 2837 , 2837 , 2862 , 2862
_pdf_backend_objcompresslevel:n	_pdf_backend_version_major_-
. 2623 , 2637 , 2638 , 2640	gset:n 2465 , 2465 ,
_pdf_backend_object_id:n	2651 , 2651 , 2827 , 2827 , 2860 , 2860
. 2304 , 2307 ,	_pdf_backend_version_minor:
2539 , 2557 , 2716 , 2719 , 2849 , 2851 2471 , 2473 , 2474 , 2679 , 2692 ,
\g_pdf_backend_object_int	2834 , 2835 , 2837 , 2838 , 2862 , 2863
. 2305 , 2372 , 2374 ,	_pdf_backend_version_minor_-
2379 , 2548 , 2717 , 2750 , 2752 , 2757	gset:n 2465 , 2469 ,
_pdf_backend_object_last:	2651 , 2668 , 2827 , 2832 , 2860 , 2861
. 2378 , 2378 ,	_pdf_exp_not_i:nn
2601 , 2601 , 2756 , 2756 , 2849 , 2856 2558 , 2577 , 2582 , 2588
_pdf_backend_object_new:	_pdf_exp_not_ii:nn
. 2304 , 2304 , 2558 , 2578 , 2583 , 2589
2539 , 2539 , 2716 , 2716 , 2849 , 2849	pdf.baselineskip 3792
_pdf_backend_object_now:nn	pdf.bordertracking 3550
2370 , 2370 , 2377 , 2590 , 2590 , 2600 ,	pdf.bordertracking.begin 3550
2748 , 2748 , 2755 , 2849 , 2854 , 2855	pdf.bordertracking.continue 3550
\g_pdf_backend_object_prop	pdf.bordertracking.end 3550
. 2538 , 2715	pdf.bordertracking.endpage 3550
_pdf_backend_object_ref:n	pdf.breaklink 3688
2304 , 2306 , 2307 , 2311 , 2539 , 2556 ,	pdf.breaklink.write 3688
2716 , 2718 , 2719 , 2723 , 2849 , 2850	pdf.brokenlink.dict 3550
_pdf_backend_object_write:nn	pdf.brokenlink.rect 3550
. 2558 , 2567 , 2569 , 2598 , 2849	pdf.brokenlink.skip 3550
_pdf_backend_object_write:nnn	pdf.count 3688
2308 , 2308 , 2314 , 2558 , 2558 , 2587 ,	pdf.currentrect 3688
2720 , 2720 , 2725 , 2849 , 2852 , 2853	pdf.cvs 3472
_pdf_backend_object_write_-	pdf.dest.anchor 3515
array:nn 2308 , 2332 , 2720 , 2726	pdf.dest.point 3515
_pdf_backend_object_write_-	pdf.dest.x 3515
aux:nnn 2308 , 2310 , 2315 , 2373	pdf.dest.y 3515
_pdf_backend_object_write_-	pdf.dest2device 3515
dict:nn 2308 , 2337 , 2720 , 2731	pdf.dev.x 3515
_pdf_backend_object_write_-	pdf.dev.y 3515
fstream:nn 2308 , 2342 , 2720 , 2736	pdf.dvi.pt 3472
_pdf_backend_object_write_-	pdf.globaldict 3469
fstream:nnn 2345 , 2347	pdf.leftboundary 3550
_pdf_backend_object_write_-	pdf.linkdp.pad 3476
stream:nn 2308 , 2357 , 2720 , 2738	pdf.linkht.pad 3476
_pdf_backend_object_write_-	pdf.linkmargin 3476
stream:nnn 2308 , 2360 , 2362	pdf.llx 3479
_pdf_backend_object_write_-	pdf.lly 3479
stream:nnnn 2720 , 2737 , 2739 , 2740	pdf.originx 3550

pdf.originy	3550
pdf.outerbox	3792
pdf.pdfmark	3792
pdf.pdfmark.dict	3792
pdf.pdfmark.good	3792
pdf.pt.dvi	3472
pdf.rect	3479
pdf.rect.ht	3472
pdf.rightboundary	3550
pdf.save.linkll	3479
pdf.save.linkur	3479
pdf.save.ll	3479
pdf.save.ur	3479
pdf.tmpa	3515
pdf.tmpb	3515
pdf.tmpc	3515
pdf.tmpd	3515
pdf.urx	3479
pdf.ury	3479
pdfannot internal commands:	
_pdfannot_backend:n	3201, 3201, 3203, 3208, 3232, 3245, 3250, 3251
_l_pdfannot_backend_breaklink_pdfmark_tl	2939, 3007, 3098
_pdfannot_backend_breaklink_postscript:n	2941, 2941, 2991, 2993, 3099
_pdfannot_backend_breaklink_usebox:N	2942, 2942, 2992, 3101
_l_pdfannot_backend_content_box	2900, 2965, 2989, 2992, 2994, 3023, 3034
_pdfannot_backend_generic:nnnn	2903, 2903, 3116, 3116, 3205, 3205, 3254, 3254, 3266
_pdfannot_backend_generic_aux:nnnn	2903, 2905, 2908
_g_pdfannot_backend_int	2902, 2921, 2925, 2933, 2999, 3000, 3204, 3207, 3210, 3218, 3229, 3231
_pdfannot_backend_last:	2932, 2932, 3129, 3129, 3217, 3217, 3255, 3255, 3267
_pdfannot_backend_link:nw	2943
_pdfannot_backend_link_aux:nw	2943
_pdfannot_backend_link_begin:n	3220, 3222, 3226, 3227
_pdfannot_backend_link_begin:nnnw	3140, 3141, 3143, 3144, 3256, 3258
_pdfannot_backend_link_begin:nw	2945, 2949, 2950
_pdfannot_backend_link_begin_aux:nw	2953, 2955
_pdfannot_backend_link_begin_goto:nnw	2943, 2943, 3140, 3140, 3220, 3220, 3256, 3256
_pdfannot_backend_link_begin_user:nnw	2943, 2948, 3140, 3142, 3220, 3225, 3256, 3257
_g_pdfannot_backend_link_bool	2938, 2952, 2957, 2972, 3010
_g_pdfannot_backend_link_dict_tl	2935, 2960, 3005
_pdfannot_backend_link_end:	2943, 2970, 3140, 3155, 3220, 3244, 3256, 3259
_pdfannot_backend_link_end_aux:	2943, 2973, 2975
_g_pdfannot_backend_link_int	2934, 3000, 3004, 3104, 3219, 3230, 3236, 3247
_pdfannot_backend_link_last:	3103, 3103, 3164, 3164, 3246, 3246, 3260, 3260
_pdfannot_backend_link_margin:n	3105, 3105, 3175, 3175, 3248, 3248, 3261, 3261
_g_pdfannot_backend_link_math_bool	2937, 2963, 2964, 2967, 2977
_pdfannot_backend_link_minima:	2943, 2981, 3012
_pdfannot_backend_link_off:	3112, 3113, 3185, 3192, 3250, 3251, 3262, 3263
_pdfannot_backend_link_on:	3112, 3112, 3185, 3185, 3250, 3250, 3262, 3262
_pdfannot_backend_link_outerbox:n	2943, 2983, 3041
_g_pdfannot_backend_link_sf_int	2936, 3062, 3073, 3074
_pdfannot_backend_link_sf_restore:	2943, 2966, 3009, 3069
_pdfannot_backend_link_sf_save:	2943, 2961, 2979, 3060
_l_pdfannot_backend_model_box	2901, 2982, 3014, 3022, 3033, 3048, 3050
pdfmanagement commands:	
_pdfmanagement_add:nnn	815, 3330, 3341, 3382, 3385
_pdfmanagement_if_active_p:	810, 811, 3323, 3324, 3399, 3400
peek commands:	
_peek_meaning:NTF	2202, 2205
_peek_remove_spaces:n	2200

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, 2538, 2715	
\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, 2230, 2233, 2502, 2516, 2632, 2649, 2657, 2664, 2677, 3158, 3183	
scan internal commands:		
\s__color_stop	657, 658, 662, 666, 679, 682, 686, 690, 704, 879, 908, 912, 1062, 1064	
\s__graphics_stop	1861, 1895, 2195, 2210, 2217, 2221, 2223, 2225, 2280, 2288	
separation	3466	
seq commands:		
\seq_set_from_clist:Nn	1793, 1817, 1963, 2151	
shipout commands:		
\l_shipout_box	3082, 3084, 3092	
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, 2321, 2571	
\str_case:nnTF	2389, 2490, 2767	
\str_convert_pdfname:n	605, 625, 834	
\str_if_empty:NnTF	1832, 1848	
\str_if_empty_p:N	1873	
\str_if_eq:nnTF	554, 784, 1475, 3377	
\str_new:N	1906, 1907, 1908	
\str_tail:N	1920, 1946	
sys commands:		
\sys_if_shell:TF	1904	
\sys_shell_now:n	1931	
T		
T _E X and L ^A T _E X 2 _ε commands:		
\@ifl@t@r	49, 51	
\special		2
tex commands:		
\tex_afterassignment:D	2229	
\tex_baselineskip:D	3054	
\tex_endinput:D	44	
\tex_global:D	2625, 2642, 2656, 2663, 2670	
\tex_immediate:D	1868, 2561, 2564, 2593, 2596	
\tex_luaTeXversion:D	2654, 2682	
\tex_pageheight:D	2890	
\tex_pagewidth:D	2889	
\tex_pdfannot:D	3122	
\tex_pdfcatalog:D	2524	
\tex_pdfcolorstack:D	490, 500	
\tex_pdfcolorstackinit:D	478	
\tex_pdfcompresslevel:D	2630	
\tex_pdfdest:D	2487, 2510	
\tex_pdfendlink:D	3161	
\tex_pdfextension:D	91, 101, 111, 121, 131, 140, 149, 159, 487, 497, 2484, 2507, 2521, 2531, 2542, 2561, 2593, 3119, 3147, 3158, 3187, 3194	
\tex_pdffeedback:D	475, 2550, 2605, 2616, 3133, 3168	
\tex_pdfinfo:D	2534	
\tex_pdflastannot:D	3136	
\tex_pdflastlink:D	3171	
\tex_pdflastobj:D	2553, 2608	
\tex_pdflastximage:D	1863, 1891	
\tex_pdflastximagepages:D	1957	
\tex_pdflinkmargin:D	3181	
\tex_pdfliteral:D	94, 104, 114, 124	
\tex_pdfmajorversion:D	2661, 2663, 2687, 2688	
\tex_pdfminorversion:D	2675, 2699	
\tex_pdfobj:D	2545, 2564, 2596	
\tex_pdfobjcompresslevel:D	2647	
\tex_pdfpageref:D	2619	
\tex_pdfrefximage:D	1891, 1898	
\tex_pdfrestore:D	143	
\tex_pdfrunninglinkoff:D	3197	
\tex_pdfrunninglinkon:D	3190	
\tex_pdfsave:D	134	
\tex_pdfsetmatrix:D	152, 162	
\tex_pdfstartlink:D	3150	
\tex_pdfvariable:D	2627, 2644, 2656, 2672, 2683, 2696, 3178	
\tex_pdfximage:D	1868, 1955	
\tex_spacefactor:D	3065, 3074	
\tex_special:D	46	
\tex_the:D	1863, 2683, 2688, 2694	
\tex_vss:D	2419, 2426, 2797, 2816	
\tex_XeTeXpdffile:D	2087, 2133	

<code>\tex_XeTeXpdfpagecount:D</code>	2143	526, 1629, 1821, 2935, 2939, 3333, 3334
<code>\tex_XeTeXpicfile:D</code>	2078	<code>\tl_set:Nn</code> . 527, 528, 537, 538, 983,
<code>TeXcolorseparation</code>	<u>3466</u>	995, 1826, 1843, 1934, 2940, 3098,
<code>\textwidth</code>	3049	3335, 3336, 3339, 3340, 3380, 3381
tl commands:		<code>\tl_to_str:n</code> 2194, 2216
<code>\c_space_tl</code>		<code>\tl_use:N</code> 745, 858
. 306, 311, 314, 567, 572, 610, 713,		token commands:
787, 997, 1625, 1803, 1804, 1805,		<code>\c_math_toggle_token</code> 2968, 2978
1806, 1993, 1994, 1995, 1996, 2044,		
2047, 2049, 2050, 2051, 2052, 2113,		U
2135, 2262, 2263, 2264, 2265, 2610,		use commands:
2621, 3005, 3138, 3173, 3210, 3237		<code>\use:N</code> 43, 2330, 2722, 2751
<code>\tl_clear:N</code> 1825, 1841,		<code>\use:n</code> 58, 813, 839,
1975, 1983, 2077, 2085, 2242, 2249		894, 1053, 1066, 1310, 1442, 1520,
<code>\tl_gclear:N</code> 1663, 1699		1532, 1544, 1704, 2107, 2191, 2213
<code>\tl_gset:Nn</code> 1622, 2960		<code>\use_none:n</code> 1721
<code>\tl_if_blank:nTF</code> 480, 565,		<code>\use_none:nnn</code> 3077
661, 678, 685, 703, 829, 911, 2112, 2198		V
<code>\tl_if_empty:NTF</code> . 1625, 1828, 1878,		<code>\value</code> 2985
1887, 2014, 2018, 2045, 2060, 2100		vbox commands:
<code>\tl_if_empty:nTF</code> 923, 1719		<code>\vbox_set:Nn</code> 3084
<code>\tl_if_empty_p:N</code> 1872, 2057		<code>\vbox_to_zero:n</code> 2415, 2422, 2789, 2800
<code>\tl_new:N</code> 525,		<code>\vbox_unpack_drop:N</code> 3092