

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}{2023-04-19}{}
5   {L3 backend support: dvipdfmx}
6 </dvipdfmx>
7 <*dvips>
8   {l3backend-dvips.def}{2023-04-19}{}
9   {L3 backend support: dvips}
10 </dvips>
11 <*dvisvgm>
12   {l3backend-dvisvgm.def}{2023-04-19}{}
13   {L3 backend support: dvisvgm}
14 </dvisvgm>
15 <*luatex>
16   {l3backend-luatex.def}{2023-04-19}{}
17   {L3 backend support: PDF output (LuaTeX)}
18 </luatex>
19 <*pdftex>
20   {l3backend-pdftex.def}{2023-04-19}{}
21   {L3 backend support: PDF output (pdfTeX)}
22 </pdftex>
23 <*xetex>
24   {l3backend-xetex.def}{2023-04-19}{}
25   {L3 backend support: XeTeX}
26 </xetex>
```

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

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

```

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

```

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

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

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

```

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

```

(End definition for `__kernel_backend_literal:e`.)

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

```

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

```

(End definition for `\kernel_backend_first_shipout:n`.)

1.1 dvips backend

```

60 <*dvips>

```

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

```

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

```

(End definition for `_kernel_backend_literal_postscript:n`.)

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

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

(End definition for `_kernel_backend_postscript:n`.)

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

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

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

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

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

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

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

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

```
87 </dvips>
```

1.2 LuaTeX and pdfTeX backends

88 `<*luatex | pdftex>`

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

`_kernel_backend_literal_pdf:n`
`_kernel_backend_literal_pdf:x`

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

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

(End definition for `_kernel_backend_literal_pdf:n`.)

`_kernel_backend_literal_page:n`

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

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

(End definition for `_kernel_backend_literal_page:n`.)

`_kernel_backend_scope_begin:`

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

`_kernel_backend_scope_end:`

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

```

126 </pdftex>
127 }

```

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

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

```

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

```

(End definition for `_kernel_backend_matrix:n.`)

```

139 </luatex | pdftex>

```

1.3 dvipdfmx backend

```

140 <*dvipdfmx | xetex>

```

The `dvipdfmx` shares code with the PDF mode one (using the common section to this file) but also with `XYTeX`. The latter is close to identical to `dvipdfmx` and so all of the code here is extracted for both backends, with some `clean up` for `XYTeX` as required.

`_kernel_backend_literal_pdf:n` 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.

```

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

```

(End definition for `_kernel_backend_literal_pdf:n.`)

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

```

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

```

(End definition for `_kernel_backend_literal_page:n.`)

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

```

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

```

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

```

150 </dvipdfmx | xetex>

```

1.4 dvisvgm backend

151 `*dvisvgm)`

`_kernel_backend_literal_svg:n`
`_kernel_backend_literal_svg:x`

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

152 `\cs_new_protected:Npn _kernel_backend_literal_svg:n #1`
 153 `{ _kernel_backend_literal:n { dvisvgm:raw~ #1 { ?nl } } }`
 154 `\cs_generate_variant:Nn _kernel_backend_literal_svg:n { x }`

(End definition for `_kernel_backend_literal_svg:n`.)

`\g_kernel_backend_scope_int`
`\l_kernel_backend_scope_int`

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

155 `\int_new:N \g_kernel_backend_scope_int`
 156 `\int_new:N \l_kernel_backend_scope_int`

(End definition for `\g_kernel_backend_scope_int` and `\l_kernel_backend_scope_int`.)

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

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

157 `\cs_new_protected:Npn _kernel_backend_scope_begin:`
 158 `{`
 159 `_kernel_backend_literal_svg:n { <g> }`
 160 `\int_set_eq:NN`
 161 `\l_kernel_backend_scope_int`
 162 `\g_kernel_backend_scope_int`
 163 `\group_begin:`
 164 `\int_gset:Nn \g_kernel_backend_scope_int { 1 }`
 165 `}`
 166 `\cs_new_protected:Npn _kernel_backend_scope_end:`
 167 `{`
 168 `\prg_replicate:nn`
 169 `{ \g_kernel_backend_scope_int }`
 170 `{ _kernel_backend_literal_svg:n { </g> } }`
 171 `\group_end:`
 172 `\int_gset_eq:NN`
 173 `\g_kernel_backend_scope_int`
 174 `\l_kernel_backend_scope_int`
 175 `}`
 176 `\cs_new_protected:Npn _kernel_backend_scope_begin:n #1`
 177 `{`
 178 `_kernel_backend_literal_svg:n { <g ~ #1 > }`
 179 `\int_set_eq:NN`
 180 `\l_kernel_backend_scope_int`
 181 `\g_kernel_backend_scope_int`
 182 `\group_begin:`
 183 `\int_gset:Nn \g_kernel_backend_scope_int { 1 }`
 184 `}`
 185 `\cs_generate_variant:Nn _kernel_backend_scope_begin:n { x }`

```

186 \cs_new_protected:Npn \__kernel_backend_scope:n #1
187 {
188   \__kernel_backend_literal_svg:n { <g ~ #1 > }
189   \int_gincr:N \g__kernel_backend_scope_int
190 }
191 \cs_generate_variant:Nn \__kernel_backend_scope:n { x }

```

(End definition for __kernel_backend_scope_begin: and others.)

```

192 </dvisvgm>
193 </package>

```

2 I3backend-box implementation

```

194 <*package>
195 <@@=box>

```

2.1 dvips backend

```

196 <*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.

```

197 \cs_new_protected:Npn \__box_backend_clip:N #1
198 {
199   \__kernel_backend_scope_begin:
200   \__kernel_backend_align_begin:
201   \__kernel_backend_literal_postscript:n { matrix-currentmatrix }
202   \__kernel_backend_literal_postscript:n
203     { Resolution~72~div~VResolution~72~div~scale }
204   \__kernel_backend_literal_postscript:n { DVImag-dup~scale }
205   \__kernel_backend_literal_postscript:x
206     {
207       0 ~
208       \dim_to_decimal_in_bp:n { \box_dp:N #1 } ~
209       \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
210       \dim_to_decimal_in_bp:n { -\box_ht:N #1 - \box_dp:N #1 } ~
211       rectclip
212     }
213   \__kernel_backend_literal_postscript:n { setmatrix }
214   \__kernel_backend_align_end:
215   \hbox_overlap_right:n { \box_use:N #1 }
216   \__kernel_backend_scope_end:
217   \skip_horizontal:n { \box_wd:N #1 }
218 }

```

(End definition for __box_backend_clip:N.)

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

```

219 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
220 { \exp_args:NNf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
221 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
222 {
223   \__kernel_backend_scope_begin:
224   \__kernel_backend_align_begin:
225   \__kernel_backend_literal_postscript:x
226   {
227     \fp_compare:nNnTF {#2} = \c_zero_fp
228     { 0 }
229     { \fp_eval:n { round ( -(#2) , 5 ) } } } ~
230   rotate
231   }
232   \__kernel_backend_align_end:
233   \box_use:N #1
234   \__kernel_backend_scope_end:
235 }

```

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

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

```

236 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
237 {
238   \__kernel_backend_scope_begin:
239   \__kernel_backend_align_begin:
240   \__kernel_backend_literal_postscript:x
241   {
242     \fp_eval:n { round ( #2 , 5 ) } ~
243     \fp_eval:n { round ( #3 , 5 ) } ~
244     scale
245   }
246   \__kernel_backend_align_end:
247   \hbox_overlap_right:n { \box_use:N #1 }
248   \__kernel_backend_scope_end:
249 }

```

(End definition for `__box_backend_scale:Nnn`.)

250 `\</dvips>`

2.2 LuaTeX and pdfTeX backends

251 `<*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.

```

252 \cs_new_protected:Npn \__box_backend_clip:N #1
253 {
254   \__kernel_backend_scope_begin:
255   \__kernel_backend_literal_pdf:x
256   {

```



```

257     0~
258     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
259     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
260     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
261     re~W~n
262   }
263   \hbox_overlap_right:n { \box_use:N #1 }
264   \__kernel_backend_scope_end:
265   \skip_horizontal:n { \box_wd:N #1 }
266 }

```

(End definition for `__box_backend_clip:N`.)

```

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

```

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

```

267 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
268 { \exp_args:Nnf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
269 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
270 {
271   \__kernel_backend_scope_begin:
272   \box_set_wd:Nn #1 { Opt }
273   \fp_set:Nn \l__box_backend_cos_fp { round ( cosd ( #2 ) , 5 ) }
274   \fp_compare:nNnT \l__box_backend_cos_fp = \c_zero_fp
275     { \fp_zero:N \l__box_backend_cos_fp }
276   \fp_set:Nn \l__box_backend_sin_fp { round ( sind ( #2 ) , 5 ) }
277   \__kernel_backend_matrix:x
278   {
279     \fp_use:N \l__box_backend_cos_fp \c_space_tl
280     \fp_compare:nNnTF \l__box_backend_sin_fp = \c_zero_fp
281       { 0~0 }
282     {
283       \fp_use:N \l__box_backend_sin_fp
284       \c_space_tl
285       \fp_eval:n { -\l__box_backend_sin_fp }
286     }
287     \c_space_tl
288     \fp_use:N \l__box_backend_cos_fp
289   }
290   \box_use:N #1
291   \__kernel_backend_scope_end:
292 }
293 \fp_new:N \l__box_backend_cos_fp
294 \fp_new:N \l__box_backend_sin_fp

```

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

```

\__box_backend_scale:Nnn

```

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

```

295 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
296 {
297   \__kernel_backend_scope_begin:
298   \__kernel_backend_matrix:x

```

```

299     {
300       \fp_eval:n { round ( #2 , 5 ) } ~
301       0~0~
302       \fp_eval:n { round ( #3 , 5 ) }
303     }
304     \hbox_overlap_right:n { \box_use:N #1 }
305     \__kernel_backend_scope_end:
306   }

```

(End definition for __box_backend_scale:Nnn.)

```
307 </luatex | pdftex>
```

2.3 dvipdfmx/X_YTeX backend

```
308 <*dvipdfmx | xetex>
```

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

```

309 \cs_new_protected:Npn \__box_backend_clip:N #1
310 {
311   \__kernel_backend_scope_begin:
312   \__kernel_backend_literal_pdf:x
313   {
314     0~
315     \dim_to_decimal_in_bp:n { -\box_dp:N #1 } ~
316     \dim_to_decimal_in_bp:n { \box_wd:N #1 } ~
317     \dim_to_decimal_in_bp:n { \box_ht:N #1 + \box_dp:N #1 } ~
318     re~W~n
319   }
320   \hbox_overlap_right:n { \box_use:N #1 }
321   \__kernel_backend_scope_end:
322   \skip_horizontal:n { \box_wd:N #1 }
323 }

```

(End definition for __box_backend_clip:N.)

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

```

324 \cs_new_protected:Npn \__box_backend_rotate:Nn #1#2
325 { \exp_args:Nmf \__box_backend_rotate_aux:Nn #1 { \fp_eval:n {#2} } }
326 \cs_new_protected:Npn \__box_backend_rotate_aux:Nn #1#2
327 {
328   \__kernel_backend_scope_begin:
329   \__kernel_backend_literal:x
330   {
331     x:rotate~
332     \fp_compare:nNnTF {#2} = \c_zero_fp
333     { 0 }
334     { \fp_eval:n { round ( #2 , 5 ) } }
335   }

```

```

336     \box_use:N #1
337     \__kernel_backend_scope_end:
338 }

```

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

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

```

339 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
340 {
341     \__kernel_backend_scope_begin:
342     \__kernel_backend_literal:x
343     {
344         x:scale~
345         \fp_eval:n { round ( #2 , 5 ) } ~
346         \fp_eval:n { round ( #3 , 5 ) }
347     }
348     \hbox_overlap_right:n { \box_use:N #1 }
349     \__kernel_backend_scope_end:
350 }

```

(End definition for `__box_backend_scale:Nnn`.)

```

351 </dviPDFmx | xetex>

```

2.4 dvisvgm backend

```

352 <*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!

```

353 \cs_new_protected:Npn \__box_backend_clip:N #1
354 {
355     \int_gincr:N \g__kernel_clip_path_int
356     \__kernel_backend_literal_svg:x
357     { < clipPath-id = " l3cp \int_use:N \g__kernel_clip_path_int " > }
358     \__kernel_backend_literal_svg:x
359     {
360         <
361         path ~ d =
362         "
363             M ~ 0 ~
364             \dim_to_decimal:n { -\box_dp:N #1 } ~
365             L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
366             \dim_to_decimal:n { -\box_dp:N #1 } ~
367             L ~ \dim_to_decimal:n { \box_wd:N #1 } ~
368             \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
369             L ~ 0 ~
370             \dim_to_decimal:n { \box_ht:N #1 + \box_dp:N #1 } ~
371             Z

```

```

372         "
373         />
374     }
375     \_kernel_backend_literal_svg:n
376     { < /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.

```

377     \_kernel_backend_scope_begin:n
378     {
379         transform =
380         "
381             translate ( { ?x } , { ?y } ) ~
382             scale ( 1 , -1 )
383         "
384     }
385     \_kernel_backend_scope:x
386     {
387         clip-path =
388         "url ( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int ) "
389     }
390     \_kernel_backend_scope:n
391     {
392         transform =
393         "
394             scale ( -1 , 1 ) ~
395             translate ( { ?x } , { ?y } ) ~
396             scale ( -1 , -1 )
397         "
398     }
399     \box_use:N #1
400     \_kernel_backend_scope_end:
401 }
402 \int_new:N \g__kernel_clip_path_int

```

(End definition for $_box_backend_clip:N$ and $_kernel_clip_path_int$.)

$_box_backend_rotate:Nn$ 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.

```

403 \cs_new_protected:Npn \_box_backend_rotate:Nn #1#2
404 {
405     \_kernel_backend_scope_begin:x
406     {
407         transform =
408         "
409             rotate
410             ( \fp_eval:n { round ( -(#2) , 5 ) } , ~ { ?x } , ~ { ?y } )
411         "
412     }
413     \box_use:N #1

```

```

414   \__kernel_backend_scope_end:
415   }

```

(End definition for __box_backend_rotate:Nn.)

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

```

416 \cs_new_protected:Npn \__box_backend_scale:Nnn #1#2#3
417 {
418   \__kernel_backend_scope_begin:x
419   {
420     transform =
421     "
422       translate ( { ?x } , { ?y } ) ~
423       scale
424       (
425         \fp_eval:n { round ( -#2 , 5 ) } ,
426         \fp_eval:n { round ( -#3 , 5 ) }
427       ) ~
428       translate ( { ?x } , { ?y } ) ~
429       scale ( -1 )
430     "
431   }
432   \hbox_overlap_right:n { \box_use:N #1 }
433   \__kernel_backend_scope_end:
434 }

```

(End definition for __box_backend_scale:Nnn.)

```

435 </dvisvgm>

```

```

436 </package>

```

3 l3backend-color implementation

```

437 <*package>

```

```

438 <@@=color>

```

Color support is split into parts: collecting data from L^AT_EX 2_ε, the color stack, general color, separations, and color for drawings. We have different approaches in each backend, and have some choices to make about dvipdfmx/X_ƎL_AT_EX in particular. Whilst it is in some ways convenient to use the same approach in multiple backends, the fact that dvipdfmx/X_ƎL_AT_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_ƎL_AT_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

439 \langle *luatex | pdftex \rangle

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

440 \backslash int_new:N \backslash l__color_backend_stack_int

(End definition for \backslash l__color_backend_stack_int.)

441 \langle /luatex | pdftex \rangle

3.1.2 LuaTeX and pdfTeX

442 \langle *luatex | pdftex \rangle

\backslash _kernel_color_backend_stack_init:Nnn

```
443 \cs_new_protected:Npn \_kernel_color_backend_stack_init:Nnn #1#2#3
444 {
445   \int_const:Nn #1
446   {
447      $\langle$ *luatex $\rangle$ 
448     \tex_pdffeedback:D colorstackinit ~
449      $\langle$ /luatex $\rangle$ 
450      $\langle$ *pdftex $\rangle$ 
451     \tex_pdfcolorstackinit:D
452      $\langle$ /pdftex $\rangle$ 
453     \tl_if_blank:nF {#2} { #2 ~ }
454     {#3}
455   }
456 }
```

(End definition for \backslash _kernel_color_backend_stack_init:Nnn.)

\backslash _kernel_color_backend_stack_push:nn

\backslash _kernel_color_backend_stack_pop:n

```
457 \cs_new_protected:Npn \_kernel_color_backend_stack_push:nn #1#2
458 {
459    $\langle$ *luatex $\rangle$ 
460   \tex_pdfextension:D colorstack ~
461    $\langle$ /luatex $\rangle$ 
462    $\langle$ *pdftex $\rangle$ 
463   \tex_pdfcolorstack:D
464    $\langle$ /pdftex $\rangle$ 
465   \int_eval:n {#1} ~ push ~ {#2}
466   }
467 \cs_new_protected:Npn \_kernel_color_backend_stack_pop:n #1
468 {
469    $\langle$ *luatex $\rangle$ 
470   \tex_pdfextension:D colorstack ~
471    $\langle$ /luatex $\rangle$ 
472    $\langle$ *pdftex $\rangle$ 
473   \tex_pdfcolorstack:D
474    $\langle$ /pdftex $\rangle$ 
475   \int_eval:n {#1} ~ pop \scan_stop:
476   }
```

(End definition for `_kernel_color_backend_stack_push:n` and `_kernel_color_backend_stack_pop:n`.)

477 `</luatex | pdftex>`

3.2 General color

3.2.1 dvips-style

478 `<*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:
color.sc
479 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
480   { \__color_backend_select:n { cmyk ~ #1 } }
481 \cs_new_protected:Npn \__color_backend_select_gray:n #1
482   { \__color_backend_select:n { gray ~ #1 } }
483 \cs_new_protected:Npn \__color_backend_select_named:n #1
484   { \__color_backend_select:n { ~ #1 } }
485 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
486   { \__color_backend_select:n { rgb ~ #1 } }
487 \cs_new_protected:Npn \__color_backend_select:n #1
488   {
489     \__kernel_backend_literal:n { color~push~ #1 }
490   }
491 \__kernel_backend_postscript:n { /color.sc ~ { } ~ def }
492 </dvips>
493 }
494 \cs_new_protected:Npn \__color_backend_reset:
495   { \__kernel_backend_literal:n { color~pop } }

```

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

496 `</dvips | dvisvgm>`

3.2.2 LuaTeX and pdfTeX

497 `<*luatex | pdftex>`

```

\l__color_backend_fill_tl
\l__color_backend_stroke_tl
498 \tl_new:N \l__color_backend_fill_tl
499 \tl_new:N \l__color_backend_stroke_tl
500 \tl_set:Nn \l__color_backend_fill_tl { 0 ~ g }
501 \tl_set:Nn \l__color_backend_stroke_tl { 0 ~ G }

```

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

```

\__color_backend_select_cmyk:n
\__color_backend_select_gray:n
\__color_backend_select_rgb:n
\__color_backend_select:nn
\__color_backend_reset:
Store the values then pass to the stack.
502 \cs_new_protected:Npn \__color_backend_select_cmyk:n #1
503   { \__color_backend_select:nn { #1 ~ k } { #1 ~ K } }
504 \cs_new_protected:Npn \__color_backend_select_gray:n #1
505   { \__color_backend_select:nn { #1 ~ g } { #1 ~ G } }
506 \cs_new_protected:Npn \__color_backend_select_rgb:n #1
507   { \__color_backend_select:nn { #1 ~ rg } { #1 ~ RG } }
508 \cs_new_protected:Npn \__color_backend_select:nn #1#2
509   {

```

```

510 \tl_set:Nn \l__color_backend_fill_tl {#1}
511 \tl_set:Nn \l__color_backend_stroke_tl {#2}
512 \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int { #1 ~ #2 }
513 }
514 \cs_new_protected:Npn \__color_backend_reset:
515 { \__kernel_color_backend_stack_pop:n \l__color_backend_stack_int }

```

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

```

516 </luatex | pdftex>

```

3.2.3 dvipdfmx/XgT_EX

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 pdfT_EX. 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.

```

517 <*dvipdfmx | xetex>

```

```

\__color_backend_select:n Using the single stack is relatively easy as there is only one route.
  \__color_backend_select_cmyk:n 518 \cs_new_protected:Npn \__color_backend_select:n #1
  \__color_backend_select_gray:n 519 { \__kernel_backend_literal:n { pdf : bc ~ [ #1 ] } }
  \__color_backend_select_rgb:n 520 \cs_new_eq:NN \__color_backend_select_cmyk:n \__color_backend_select:n
\__color_backend_reset: 521 \cs_new_eq:NN \__color_backend_select_gray:n \__color_backend_select:n
522 \cs_new_eq:NN \__color_backend_select_rgb:n \__color_backend_select:n
523 \cs_new_protected:Npn \__color_backend_reset:
524 { \__kernel_backend_literal:n { pdf : ec } }

```

(End 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.
525 \cs_new_protected:Npn \__color_backend_select_named:n #1
526 {
527   \str_if_eq:nnTF {#1} { Black }
528     { \__color_backend_select_gray:n { 0 } }
529     { \msg_error:nnn { color } { unknown-named-color } {#1} }
530 }
531 \msg_new:nnn { color } { unknown-named-color }
532 { Named-color~'#1'~is-not-known. }

```

(End definition for `__color_backend_select_named:n`.)

```

533 </dvipdfmx | xetex>

```

3.3 Separations

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

```

534 <*dvipdfmx | luatex | pdftex | xetex | dvips>

```

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

`\g_color_backend_colorant_prop`

```
535 \prop_new:N \g__color_backend_colorant_prop
```

(End definition for `\g__color_backend_colorant_prop`.)

`__color_backend_devicen_colorants:n`

`__color_backend_devicen_colorants:w`

```
536 \cs_new:Npx \__color_backend_devicen_colorants:n #1
```

```
537 {
```

```
538   \exp_not:N \tl_if_blank:nF {#1}
```

```
539   {
```

```
540     \c_space_tl
```

```
541     << ~
```

```
542       /Colorants ~
```

```
543       << ~
```

```
544         \exp_not:N \__color_backend_devicen_colorants:w #1 ~
```

```
545         \exp_not:N \q_recursion_tail \c_space_tl
```

```
546         \exp_not:N \q_recursion_stop
```

```
547       >> ~
```

```
548     >>
```

```
549   }
```

```
550 }
```

```
551 \cs_new:Npn \__color_backend_devicen_colorants:w #1 ~
```

```
552 {
```

```
553   \quark_if_recursion_tail_stop:n {#1}
```

```
554   \prop_if_in:NnT \g__color_backend_colorant_prop {#1}
```

```
555   {
```

```
556     #1 ~
```

```
557     \prop_item:Nn \g__color_backend_colorant_prop {#1} ~
```

```
558   }
```

```
559   \__color_backend_devicen_colorants:w
```

```
560 }
```

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

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

```
562 <*dvips>
```

`__color_backend_select_separation:nn`

`__color_backend_select_devicen:nn`

```
563 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2
```

```
564 { \__color_backend_select:n { separation ~ #1 ~ #2 } }
```

```
565 \cs_new_eq:NN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn
```

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

`__color_backend_select_iccbased:nn`

No support.

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

(End definition for `__color_backend_select_iccbased:nn`.)

`__color_backend_separation_init:nmnn`

`__color_backend_separation_init:nxxnn`

`__color_backend_separation_init_aux:nmnnnn`

`__color_backend_separation_init_DeviceCMYK:nnn`

`__color_backend_separation_init_DeviceGray:nnn`

`__color_backend_separation_init_DeviceRGB:nnn`

```
567 \cs_new_protected:Npx \__color_backend_separation_init:nmnnn #1#2#3#4#5
```

```
568 {
```

`__color_backend_separation_init_Device:Mn`

`__color_backend_separation_init:nnm`

`__color_backend_separation_init_count:n`

`__color_backend_separation_init_count:w`

`__color_backend_separation_init:nmnn`

`__color_backend_separation_init:w`

`__color_backend_separation_init:n`

`__color_backend_separation_init:nw`

`__color_backend_separation_init_CIELAB:nnm`

```

569 \bool_if:NT \g__kernel_backend_header_bool
570 {
571   \exp_args:Nx \__kernel_backend_first_shipout:n
572   {
573     \exp_not:N \__color_backend_separation_init_aux:nmnnnn
574     { \exp_not:N \int_use:N \g__color_model_int }
575     {#1} {#2} {#3} {#4} {#5}
576   }
577   \prop_gput:Nxx \exp_not:N \g__color_backend_colorant_prop
578   { / \exp_not:N \str_convert_pdfname:n {#1} }
579   {
580     << ~
581     /setcolorspace ~ {} ~
582     >> ~ begin ~
583     color \exp_not:N \int_use:N \g__color_model_int \c_space_tl
584     end
585   }
586 }
587 }
588 \cs_generate_variant:Nn \__color_backend_separation_init:nmnnnn { nxx }
589 \cs_new_protected:Npn \__color_backend_separation_init_aux:nmnnnn #1#2#3#4#5#6
590 {
591   \__kernel_backend_literal:e
592   {
593     !
594     TeXDict ~ begin ~
595     /color #1
596     {
597       [ ~
598       /Separation ~ ( \str_convert_pdfname:n {#2} ) ~
599       [ ~ #3 ~ ] ~
600       {
601         \cs_if_exist_use:cF { __color_backend_separation_init_ #3 :nnn }
602         { \__color_backend_separation_init:nmnnnn
603           {#4} {#5} {#6}
604         }
605       ] ~ setcolorspace
606     } ~ def ~
607     end
608   }
609 }
610 \cs_new:cpn { __color_backend_separation_init_ /DeviceCMYK :nnn } #1#2#3
611 { \__color_backend_separation_init_Device:Nn 4 {#3} }
612 \cs_new:cpn { __color_backend_separation_init_ /DeviceGray :nnn } #1#2#3
613 { \__color_backend_separation_init_Device:Nn 1 {#3} }
614 \cs_new:cpn { __color_backend_separation_init_ /DeviceRGB :nnn } #1#2#3
615 { \__color_backend_separation_init_Device:Nn 2 {#3} }
616 \cs_new:Npn \__color_backend_separation_init_Device:Nn #1#2
617 {
618   #2 ~
619   \prg_replicate:nn {#1}
620   { #1 ~ index ~ mul ~ #1 ~ 1 ~ roll ~ }
621   \int_eval:n { #1 + 1 } ~ -1 ~ roll ~ pop
622 }

```

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.

```

623 \cs_new:Npn \__color_backend_separation_init:nnn #1#2#3
624 {
625   \exp_args:Ne \__color_backend_separation_init:nnnn
626   { \__color_backend_separation_init_count:n {#2} }
627   {#1} {#2} {#3}
628 }
629 \cs_new:Npn \__color_backend_separation_init_count:n #1
630 { \int_eval:n { 0 \__color_backend_separation_init_count:w #1 ~ \s__color_stop } }
631 \cs_new:Npn \__color_backend_separation_init_count:w #1 ~ #2 \s__color_stop
632 {
633   +1
634   \tl_if_blank:nF {#2}
635   { \__color_backend_separation_init_count:w #2 \s__color_stop }
636 }

```

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

```

637 \cs_new:Npn \__color_backend_separation_init:nnnn #1#2#3#4
638 {
639   \__color_backend_separation_init:w #3 ~ \s__color_stop #4 ~ \s__color_stop
640   \prg_replicate:nn {#1}
641   {
642     pop ~ 1 ~ index ~ neg ~ 1 ~ index ~ add ~
643     \int_eval:n { 3 * #1 } ~ index ~ mul ~
644     2 ~ index ~ add ~
645     \int_eval:n { 3 * #1 } ~ #1 ~ roll ~
646   }
647   \int_step_function:nnnN {#1} { -1 } { 1 }
648   \__color_backend_separation_init:n
649   \int_eval:n { 4 * #1 + 1 } ~ #1 ~ roll ~
650   \prg_replicate:nn { 3 * #1 + 1 } { pop ~ }
651   \tl_if_blank:nF {#2}
652   { \__color_backend_separation_init:nw {#1} #2 ~ \s__color_stop }
653 }
654 \cs_new:Npn \__color_backend_separation_init:w
655 #1 ~ #2 \s__color_stop #3 ~ #4 \s__color_stop
656 {
657   #1 ~ #3 ~ 0 ~
658   \tl_if_blank:nF {#2}
659   { \__color_backend_separation_init:w #2 \s__color_stop #4 \s__color_stop }

```

```

660 }
661 \cs_new:Npn \__color_backend_separation_init:n #1
662 { \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.

```

663 \cs_new:Npn \__color_backend_separation_init:nw #1#2 ~ #3 ~ #4 \s__color_stop
664 {
665   #2 ~ #3 ~
666   2 ~ index ~ 2 ~ index ~ lt ~
667   { ~ pop ~ exch ~ pop ~ } ~
668   { ~
669     2 ~ index ~ 1 ~ index ~ gt ~
670     { ~ exch ~ pop ~ exch ~ pop ~ } ~
671     { ~ pop ~ pop ~ } ~
672     ifelse ~
673   }
674   ifelse ~
675   #1 ~ 1 ~ roll ~
676   \tl_if_blank:nF {#4}
677   { \__color_backend_separation_init:nw {#1} #4 \s__color_stop }
678 }

```

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.

```

679 \cs_new_protected:Npn \__color_backend_separation_init_CIELAB:nmn #1#2#3
680 {
681   \__color_backend_separation_init:nxxxxn
682   {#2}
683   {
684     /CIEBasedABC ~
685     << ~
686     /RangeABC ~ [ ~ \c__color_model_range_CIELAB_tl \c_space_tl ] ~
687     /DecodeABC ~
688     [ ~
689     { ~ 16 ~ add ~ 116 ~ div ~ } ~ bind ~
690     { ~ 500 ~ div ~ } ~ bind ~
691     { ~ 200 ~ div ~ } ~ bind ~
692     ] ~
693     /MatrixABC ~ [ ~ 1 ~ 1 ~ 1 ~ 1 ~ 0 ~ 0 ~ 0 ~ 0 ~ -1 ~ ] ~
694     /DecodeLMN ~
695     [ ~
696     { ~
697     dup ~ 6 ~ 29 ~ div ~ ge ~
698     { ~ dup ~ dup ~ mul ~ mul ~ ~ } ~
699     { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
700     ifelse ~
701     0.9505 ~ mul ~
702     } ~ bind ~
703     { ~
704     dup ~ 6 ~ 29 ~ div ~ ge ~
705     { ~ dup ~ dup ~ mul ~ mul ~ } ~
706     { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
707     ifelse ~

```

```

708         } ~ bind ~
709         { ~
710         dup ~ 6 ~ 29 ~ div ~ ge ~
711         { ~ dup ~ dup ~ mul ~ mul ~ } ~
712         { ~ 4 ~ 29 ~ div ~ sub ~ 108 ~ 841 ~ div ~ mul ~ } ~
713         ifelse ~
714         1.0890 ~ mul ~
715         } ~ bind
716     ] ~
717     /WhitePoint ~
718     [ ~ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _tl } ~ ] ~
719     >>
720 }
721 { \c__color_model_range_CIELAB_tl }
722 { 100 ~ 0 ~ 0 }
723 {#3}
724 }

```

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

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

```

725 \cs_new_protected:Npn \__color_backend_devicen_init:nmn #1#2#3
726 {
727     \__kernel_backend_literal:e
728     {
729         !
730         TeXDict ~ begin ~
731         /color \int_use:N \g__color_model_int
732         {
733             [ ~
734             /DeviceN ~
735             [ ~ #1 ~ ] ~
736             #2 ~
737             { ~ #3 ~ } ~
738             \__color_backend_devicen_colorants:n {#1}
739             ] ~ setcolorspace
740         } ~ def ~
741     end
742 }
743 }

```

(End definition for `__color_backend_devicen_init:nmn`.)

`__color_backend_iccbased_init:nmn` No support at present.

```

744 \cs_new_protected:Npn \__color_backend_iccbased_init:nmn #1#2#3 { }

```

(End definition for `__color_backend_iccbased_init:nmn`.)

```

745 </dvips>
746 <*dvisvgm>

```

`__color_backend_select_separation:nn` No support at present.

```

\__color_backend_select_devicen:nn
747 \cs_new_protected:Npn \__color_backend_select_separation:nn #1#2 { }
748 \cs_new_eq:NN \__color_backend_select_devicen:nn \__color_backend_select_separation:nn

```

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

No support at present.

`_color_backend_separation_init:nmmn`
`_color_backend_separation_init_CIELAB:nnn`

```
749 \cs_new_protected:Npn \_color_backend_separation_init:nmmn #1#2#3#4#5 { }
750 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnn #1#2#3 { }
```

(End definition for `_color_backend_separation_init:nmmn` 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.

```
751 \cs_new_protected:Npn \_color_backend_select_iccbased:nn #1#2
752 {
753   \_kernel_backend_literal_svg:x
754   {
755     <style>
756       @color-profile ~
757       \str_if_eq:nnTF {#2} { cmyk }
758       { device-cmyk }
759       { --color \int_use:N \g_color_model_int }
760       \c_space_tl
761       {
762         src:("#1")
763       }
764     </style>
765   }
766 }
```

(End definition for `_color_backend_select_iccbased:nn`.)

```
767 </dvisvgm>
768 <*dviPDFmx | luatex | pdftex | xetex>
```

`_color_backend_select_separation:nn`
`_color_backend_select_devicen:nn`
`_color_backend_select_iccbased:nn`

```
769 <*dviPDFmx | xetex>
770 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
771 { \_kernel_backend_literal:x { pdf : bc ~ \pdf_object_ref:n {#1} ~ [ #2 ] } }
772 </dviPDFmx | xetex>
773 <*luatex | pdftex>
774 \cs_new_protected:Npn \_color_backend_select_separation:nn #1#2
775 { \_color_backend_select:nn { /#1 ~ cs ~ #2 ~ scn } { /#1 ~ CS ~ #2 ~ SCN } }
776 </luatex | pdftex>
777 \cs_new_eq:NN \_color_backend_select_devicen:nn \_color_backend_select_separation:nn
778 \cs_new_eq:NN \_color_backend_select_iccbased:nn \_color_backend_select_separation:nn
```

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

`_color_backend_init_resource:n`

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

```
779 \cs_new_protected:Npn \_color_backend_init_resource:n #1
780 {
781 <*luatex | pdftex>
782   \bool_lazy_and:nnT
783   { \cs_if_exist_p:N \pdfmanagement_if_active_p: }

```

```

784     { \pdfmanagement_if_active_p: }
785     {
786       \use:x
787       {
788         \pdfmanagement_add:nnn
789         { Page / Resources / ColorSpace }
790         { #1 }
791         { \pdf_object_ref_last: }
792       }
793     }
794 </luatex | pdftex>
795   }

```

(End definition for `_color_backend_init_resource:n`.)

```

\_color_backend_separation_init:nmnm
\_color_backend_separation_init:nn
\_color_backend_separation_init_CIELAB:nnm

```

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/X4TeX`.

```

796 \cs_new_protected:Npn \_color_backend_separation_init:nmnm #1#2#3#4#5
797 {
798   \pdf_object_unnamed_write:nx { dict }
799   {
800     /FunctionType ~ 2
801     /Domain ~ [0 ~ 1]
802     \t1_if_blank:nF {#3} { /Range ~ [#3] }
803     /C0 ~ [#4] ~
804     /C1 ~ [#5] /N ~ 1
805   }
806   \exp_args:Nx \_color_backend_separation_init:nn
807   { \str_convert_pdfname:n {#1} } {#2}
808   \_color_backend_init_resource:n { color \int_use:N \g_color_model_int }
809 }
810 \cs_new_protected:Npn \_color_backend_separation_init:nn #1#2
811 {
812   \use:x
813   {
814     \pdf_object_new:n { color \int_use:N \g_color_model_int }
815     \pdf_object_write:nm { color \int_use:N \g_color_model_int } { array }
816     { /Separation /#1 ~ #2 ~ \pdf_object_ref_last: }
817   }
818   \prop_gput:Nnx \g_color_backend_colorant_prop { /#1 }
819   { \pdf_object_ref_last: }
820 }

```

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

```

821 \cs_new_protected:Npn \_color_backend_separation_init_CIELAB:nnm #1#2#3
822 {
823   \pdf_object_if_exist:nF { \_color_illuminant_CIELAB_ #1 }
824   {
825     \pdf_object_new:n { \_color_illuminant_CIELAB_ #1 }
826     \pdf_object_write:nmx { \_color_illuminant_CIELAB_ #1 } { array }
827     {

```

```

828         /Lab ~
829         <<
830         /WhitePoint ~
831         [ \tl_use:c { c__color_model_whitepoint_CIELAB_ #1 _t1 } ]
832         /Range ~ [ \c__color_model_range_CIELAB_t1 ]
833         >>
834     }
835 }
836 \__color_backend_separation_init:nnnnn
837 {#2}
838 { \pdf_object_ref:n { __color_illuminant_CIELAB_ #1 } }
839 { \c__color_model_range_CIELAB_t1 }
840 { 100 ~ 0 ~ 0 }
841 {#3}
842 }

```

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

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

```

843 \cs_new_protected:Npn \__color_backend_devicen_init:nnn #1#2#3
844 {
845   \pdf_object_unnamed_write:nx { stream }
846   {
847     {
848       /FunctionType ~ 4 ~
849       /Domain ~
850       [ ~
851         \prg_replicate:nn
852         { 0 \__color_backend_devicen_init:w #1 ~ \s__color_stop }
853         { 0 ~ 1 ~ }
854       ] ~
855       /Range ~
856       [ ~
857         \str_case:nn {#2}
858         {
859           { /DeviceCMYK } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
860           { /DeviceGray } { 0 ~ 1 }
861           { /DeviceRGB } { 0 ~ 1 ~ 0 ~ 1 ~ 0 ~ 1 }
862         } ~
863       ]
864     }
865     { {#3} }
866   }
867   \use:x
868   {
869     \pdf_object_new:n { color \int_use:N \g__color_model_int }
870     \pdf_object_write:nnn { color \int_use:N \g__color_model_int } { array }
871     {
872       /DeviceN ~
873       [ ~ #1 ~ ] ~
874       #2 ~
875       \pdf_object_ref_last:

```



```

876         \_color_backend_devicen_colorants:n {#1}
877     }
878 }
879 \_color_backend_init_resource:n { color \int_use:N \g_color_model_int }
880 }
881 \cs_new:Npn \_color_backend_devicen_init:w #1 ~ #2 \s_color_stop
882 {
883     + 1
884     \tl_if_blank:nF {#2}
885     { \_color_backend_devicen_init:w #2 \s_color_stop }
886 }

```

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

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

```

887 \cs_new_protected:Npn \_color_backend_iccbased_init:nnn #1#2#3
888 {
889     \pdf_object_if_exist:nF { __color_icc_ #1 }
890     {
891         \pdf_object_new:n { __color_icc_ #1 }
892         \pdf_object_write:nxx { __color_icc_ #1 } { fstream }
893         {
894             {
895                 /N ~ \exp_not:n { #2 } ~
896                 \tl_if_empty:nF { #3 } { /Range~[ #3 ] }
897             }
898             {#1}
899         }
900     }
901     \pdf_object_unnamed_write:nx { array }
902     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
903     \_color_backend_init_resource:n { color \int_use:N \g_color_model_int }
904 }

```

(End definition for _color_backend_iccbased_init:nnn.)

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

```

905 \cs_new_protected:Npn \_color_backend_iccbased_device:nnn #1#2#3
906 {
907     \pdf_object_if_exist:nF { __color_icc_ #1 }
908     {
909         \pdf_object_new:n { __color_icc_ #1 }
910         \pdf_object_write:nnn { __color_icc_ #1 } { fstream }
911         {
912             { /N ~ #3 }
913             {#1}
914         }
915     }
916     \pdf_object_unnamed_write:nx { array }
917     { /ICCBased ~ \pdf_object_ref:n { __color_icc_ #1 } }
918     \_color_backend_init_resource:n { Default #2 }
919 }

```

(End definition for _color_backend_iccbased_device:nnn.)

```

920 </dviPDFmx | luatex | pdftex | xetex>

```

3.4 Fill and stroke color

Here, dvipdfmx/X_YTeX we write direct PDF specials for the fill, and only use the stack for the stroke color (see above for comments on why we cannot use multiple stacks with these backends). LuaTeX and pdfTeX have mutple 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.

921 \langle *dvipdfmx | xetex \rangle

```

  \_color_backend_fill:n
\_color_backend_fill_cmyk:n 922 \cs_new_protected:Npn \_color_backend_fill:n #1
\_color_backend_fill_gray:n 923 { \_kernel_backend_literal:n { pdf : bc ~ fill ~ [ #1 ] } }
\_color_backend_fill_rgb:n 924 \cs_new_eq:NN \_color_backend_fill_cmyk:n \_color_backend_fill:n
  \_color_backend_stroke:n 925 \cs_new_eq:NN \_color_backend_fill_gray:n \_color_backend_fill:n
    \_color_backend_stroke_cmyk:n 926 \cs_new_eq:NN \_color_backend_fill_rgb:n \_color_backend_fill:n
    \_color_backend_stroke_gray:n 927 \cs_new_protected:Npn \_color_backend_stroke:n #1
    \_color_backend_stroke_rgb:n 928 { \_kernel_backend_literal:n { pdf : bc ~ stroke ~ [ #1 ] } }
929 \cs_new_eq:NN \_color_backend_stroke_cmyk:n \_color_backend_stroke:n
930 \cs_new_eq:NN \_color_backend_stroke_gray:n \_color_backend_stroke:n
931 \cs_new_eq:NN \_color_backend_stroke_rgb:n \_color_backend_stroke:n

```

(End definition for _color_backend_fill:n and others.)

```

  \_color_backend_fill_separation:nn
\_color_backend_stroke_separation:nn 932 \cs_new_protected:Npn \_color_backend_fill_separation:nn #1#2
  \_color_backend_fill_devicen:nn 933 {
  \_color_backend_stroke_devicen:nn 934 \_kernel_backend_literal:x
935 { pdf : bc ~ fill ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
936 }
937 \cs_new_protected:Npn \_color_backend_stroke_separation:nn #1#2
938 {
939 \_kernel_backend_literal:x
940 { pdf : bc ~ stroke ~ \pdf_object_ref:n {#1} ~ [ #2 ] }
941 }
942 \cs_new_eq:NN \_color_backend_fill_devicen:nn \_color_backend_fill_separation:nn
943 \cs_new_eq:NN \_color_backend_stroke_devicen:nn \_color_backend_stroke_separation:nn

```

(End definition for _color_backend_fill_separation:nn and others.)

```

\_color_backend_fill_reset:
  \_color_backend_stroke_reset: 944 \cs_new_eq:NN \_color_backend_fill_reset: \_color_backend_reset:
945 \cs_new_eq:NN \_color_backend_stroke_reset: \_color_backend_reset:

```

(End definition for _color_backend_fill_reset: and _color_backend_stroke_reset:.)

946 \langle /dvipdfmx | xetex \rangle

947 \langle *luatex | pdftex \rangle

```

\_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.

```

948 \cs_new_protected:Npn \_color_backend_fill_cmyk:n #1
949 { \_color_backend_fill:n { #1 ~ k } }

```

```

950 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
951   { \__color_backend_fill:n { #1 ~ g } }
952 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
953   { \__color_backend_fill:n { #1 ~ rg } }
954 \cs_new_protected:Npn \__color_backend_fill:n #1
955   {
956     \tl_set:Nn \l__color_backend_fill_tl {#1}
957     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
958       { #1 ~ \l__color_backend_stroke_tl }
959   }
960 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
961   { \__color_backend_stroke:n { #1 ~ K } }
962 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
963   { \__color_backend_stroke:n { #1 ~ G } }
964 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
965   { \__color_backend_stroke:n { #1 ~ RG } }
966 \cs_new_protected:Npn \__color_backend_stroke:n #1
967   {
968     \tl_set:Nn \l__color_backend_stroke_tl {#1}
969     \__kernel_color_backend_stack_push:nn \l__color_backend_stack_int
970       { \l__color_backend_fill_tl \c_space_tl #1 }
971   }

```

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

```

\__color_backend_fill_separation:nn
\__color_backend_stroke_separation:nn
\__color_backend_fill_devicen:nn
\__color_backend_stroke_devicen:nn
972 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
973   { \__color_backend_fill:n { /#1 ~ cs ~ #2 ~ scn } }
974 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
975   { \__color_backend_stroke:n { /#1 ~ CS ~ #2 ~ SCN } }
976 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
977 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

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

```

\__color_backend_fill_reset:
\__color_backend_stroke_reset:
978 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
979 \cs_new_eq:NN \__color_backend_stroke_reset: \__color_backend_reset:

```

(End definition for `__color_backend_fill_reset:` and `__color_backend_stroke_reset:.`)

```

980 </luatex | pdftex>

```

```

981 <*dvips>

```

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

```

\__color_backend_fill_gray:n
\__color_backend_fill_rgb:n
\__color_backend_fill:n
\__color_backend_stroke_cmyk:n
\__color_backend_stroke_gray:n
\__color_backend_stroke_rgb:n
982 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
983   { \__color_backend_fill:n { cmyk ~ #1 } }
984 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
985   { \__color_backend_fill:n { gray ~ #1 } }
986 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
987   { \__color_backend_fill:n { rgb ~ #1 } }
988 \cs_new_protected:Npn \__color_backend_fill:n #1
989   {
990     \__kernel_backend_literal:n { color~push~ #1 }
991   }

```

```

992 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
993   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setcmykcolor } def } }
994 \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
995   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setgray } def } }
996 \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
997   { \__kernel_backend_postscript:n { /color.sc { #1 ~ setrgbcolor } def } }

```

(End definition for __color_backend_fill_cmyk:n and others.)

```

\__color_backend_fill_separation:nn
\__color_backend_stroke_separation:nn
\__color_backend_fill_devicen:nn
\__color_backend_stroke_devicen:nn
998 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2
999   { \__color_backend_fill:n { separation ~ #1 ~ #2 } }
1000 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2
1001   { \__kernel_backend_postscript:n { /color.sc { separation ~ #1 ~ #2 } def } }
1002 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1003 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

\__color_backend_fill_reset:
\__color_backend_stroke_reset:

```

```

1004 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1005 \cs_new_protected:Npn \__color_backend_stroke_reset: { }

```

(End definition for __color_backend_fill_reset: and __color_backend_stroke_reset:.)

```

1006 </dvips>
1007 <*dvisvgm>

```

```

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

```

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

```

1008 \cs_new_protected:Npn \__color_backend_fill_cmyk:n #1
1009   { \__color_backend_fill:n { cmyk ~ #1 } }
1010 \cs_new_protected:Npn \__color_backend_fill_gray:n #1
1011   { \__color_backend_fill:n { gray ~ #1 } }
1012 \cs_new_protected:Npn \__color_backend_fill_rgb:n #1
1013   { \__color_backend_fill:n { rgb ~ #1 } }
1014 \cs_new_protected:Npn \__color_backend_fill:n #1
1015   {
1016     \__kernel_backend_literal:n { color~push~ #1 }
1017   }

```

(End definition for __color_backend_fill_cmyk:n and others.)

```

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

```

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

```

1018 \cs_new_protected:Npn \__color_backend_stroke_cmyk:n #1
1019   { \__color_backend_cmyk:w #1 \s__color_stop }
1020 \cs_new_protected:Npn \__color_backend_stroke_cmyk:w
1021   #1 ~ #2 ~ #3 ~ #4 \s__color_stop
1022   {
1023     \use:x
1024     {
1025       \__color_backend:nnn
1026       { \fp_eval:n { -100 * ( 1 - min ( 1 , #1 + #4 ) ) } }
1027       { \fp_eval:n { -100 * ( 1 - min ( 1 , #2 + #4 ) ) } }
1028       { \fp_eval:n { -100 * ( 1 - min ( 1 , #3 + #4 ) ) } }

```

```

1029     }
1030   }
1031   \cs_new_protected:Npn \__color_backend_stroke_gray:n #1
1032   {
1033     \use:x
1034     {
1035       \__color_backend_stroke_gray_aux:n
1036       { \fp_eval:n { 100 * (#1) } }
1037     }
1038   }
1039   \cs_new_protected:Npn \__color_backend_stroke_gray_aux:n #1
1040   { \__color_backend:nnn {#1} {#1} {#1} }
1041   \cs_new_protected:Npn \__color_backend_stroke_rgb:n #1
1042   { \__color_backend_rgb:w #1 \s__color_stop }
1043   \cs_new_protected:Npn \__color_backend_stroke_rgb:w
1044   #1 ~ #2 ~ #3 \s__color_stop
1045   {
1046     \use:x
1047     {
1048       \__color_backend:nnn
1049       { \fp_eval:n { 100 * (#1) } }
1050       { \fp_eval:n { 100 * (#2) } }
1051       { \fp_eval:n { 100 * (#3) } }
1052     }
1053   }
1054   \cs_new_protected:Npx \__color_backend:nnn #1#2#3
1055   {
1056     \__kernel_backend_scope:n
1057     {
1058       stroke =
1059       "
1060         rgb
1061         (
1062           #1 \c_percent_str ,
1063           #2 \c_percent_str ,
1064           #3 \c_percent_str
1065         )
1066       "
1067     }
1068   }

```

(End 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.

```

1069 \cs_new_protected:Npn \__color_backend_fill_separation:nn #1#2 { }
1070 \cs_new_protected:Npn \__color_backend_stroke_separation:nn #1#2 { }
1071 \cs_new_eq:NN \__color_backend_fill_devicen:nn \__color_backend_fill_separation:nn
1072 \cs_new_eq:NN \__color_backend_stroke_devicen:nn \__color_backend_stroke_separation:nn

```

(End definition for __color_backend_fill_separation:nn and others.)

```

\__color_backend_fill_reset:
\__color_backend_stroke_reset:

```

```

1073 \cs_new_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
1074 \cs_new_protected:Npn \__color_backend_stroke_reset: { }

```

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

`_color_backend_devicen_init:nnn` No support at present.

`_color_backend_iccbased_init:nnn` 1075 `\cs_new_protected:Npn _color_backend_devicen_init:nnn #1#2#3 { }`
1076 `\cs_new_protected:Npn _color_backend_iccbased_init:nnn #1#2#3 { }`

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

1077 `</divisvgm>`

1078 `</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.

```
1079 <*lua>
1080 local l = lpeg
1081 local spaces = 1.P' ^0
1082 local digit16 = 1.R('09', 'af', 'AF')
1083
1084 local octet = digit16 * digit16 / function(s)
1085   return string.format('%.3g ', tonumber(s, 16) / 255)
1086 end
1087
1088 if luaotfload and luaotfload.set_transparent_colorstack then
1089   local htmlcolor = 1.Cs(octet * octet * octet * -1 * 1.Cc'rg')
1090   local color_export = {
1091     token.create'tex_endlocalcontrol:D',
1092     token.create'tex_hpack:D',
1093     token.new(0, 1),
1094     token.create'color_export:nnN',
1095     token.new(0, 1),
1096     '',
1097     token.new(0, 2),
1098     token.new(0, 1),
1099     'backend',
1100     token.new(0, 2),
1101     token.create'l_tmpa_tl',
1102     token.create'exp_after:wN',
1103     token.create'__color_select:nn',
1104     token.create'l_tmpa_tl',
1105     token.new(0, 2),
1106   }
1107   local group_end = token.create'group_end:'
1108   local value = (1 - 1.P' }')^0
1109   luatexbase.add_to_callback('luaotfload.parse_color', function (value)
1110     % Also allow HTML colors to preserve compatibility
1111     local html = htmlcolor:match(value)
1112     if html then return html end
1113
1114     tex.runtoks(function()
1115       token.get_next()
1116       color_export[6] = value
```

```

1117     tex.sprint(-2, color_export)
1118   end)
1119   local list = token.scan_list()
1120   if not list.head or list.head.next
1121     or list.head.subtype ~= node.subtype'pdf_colorstack' then
1122     error'Unexpected backend behavior'
1123   end
1124   local cmd = list.head.data
1125   node.free(list)
1126   return cmd
1127 end, 'l3color')
1128 end
1129 </lua>
1130 <*luatex>
1131 <*package>
1132 \lua_load_module:n {l3backend-luatex}
1133 </package>
1134 </luatex>

```

4 l3backend-draw implementation

```

1135 <*package>
1136 <@@=draw>

```

4.1 dvips backend

```

1137 <*dvips>

```

```

\__draw_backend_literal:n The same as literal PostScript: same arguments about positioning apply her.
\__draw_backend_literal:x 1138 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_postscript:n
1139 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for `__draw_backend_literal:n`.)

```

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

```

```

1140 \cs_new_protected:Npn \__draw_backend_begin:
1141   {
1142     \__kernel_backend_literal:n { ps::[begin] }
1143     \__draw_backend_literal:n { @beginspecial }
1144   }
1145 \cs_new_protected:Npn \__draw_backend_end:
1146   {
1147     \__draw_backend_literal:n { @endspecial }
1148     \__kernel_backend_literal:n { ps::[end] }
1149   }

```

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

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

```

1150 \cs_new_protected:Npn \_draw_backend_scope_begin:
1151   { \_draw_backend_literal:n { save } }
1152 \cs_new_protected:Npn \_draw_backend_scope_end:
1153   { \_draw_backend_literal:n { restore } }

```

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

`_draw_backend_moveto:nn` Path creation operations mainly resolve directly to PostScript primitive steps, with only
`_draw_backend_lineto:nn` the need to convert to bp. Notice that x-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:nnmmnn` rectangular path command (without also clipping, filling or stroking), so that task is
done using a small amount of PostScript.

```

1154 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1155   {
1156     \_draw_backend_literal:x
1157     {
1158       \dim_to_decimal_in_bp:n {#1} ~
1159       \dim_to_decimal_in_bp:n {#2} ~ moveto
1160     }
1161   }
1162 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1163   {
1164     \_draw_backend_literal:x
1165     {
1166       \dim_to_decimal_in_bp:n {#1} ~
1167       \dim_to_decimal_in_bp:n {#2} ~ lineto
1168     }
1169   }
1170 \cs_new_protected:Npn \_draw_backend_rectangle:nmmn #1#2#3#4
1171   {
1172     \_draw_backend_literal:x
1173     {
1174       \dim_to_decimal_in_bp:n {#4} ~ \dim_to_decimal_in_bp:n {#3} ~
1175       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1176       moveto~dup~0~rlineto~exch~0~exch~rlineto~neg~0~rlineto~closepath
1177     }
1178   }
1179 \cs_new_protected:Npn \_draw_backend_curveto:nnmmnn #1#2#3#4#5#6
1180   {
1181     \_draw_backend_literal:x
1182     {
1183       \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1184       \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1185       \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1186       curveto
1187     }
1188   }

```

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

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

```

\_draw_backend_nonzero_rule: 1189 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
\g__draw_draw_eor_bool      1190 { \bool_gset_true:N \g__draw_draw_eor_bool }
                             1191 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
                             1192 { \bool_gset_false:N \g__draw_draw_eor_bool }
                             1193 \bool_new:N \g__draw_draw_eor_bool

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

```

`_draw_backend_closepath:` Unlike PDF, PostScript doesn't track separate colors for strokes and other elements. It is 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 TeX switch.

`_draw_backend_fillstroke:` All of the operations end with a new path instruction as they do not terminate (again in contrast to PDF).

```

\_draw_backend_clip: 1194 \cs_new_protected:Npn \_draw_backend_closepath:
\_draw_backend_stroke: 1195 { \_draw_backend_literal:n { closepath } }
\_draw_backend_fill: 1196 \cs_new_protected:Npn \_draw_backend_stroke:
\_draw_backend_clip: 1197 {
\_draw_backend_fillstroke: 1198 \_draw_backend_literal:n { gsave }
\_draw_backend_clip: 1199 \_draw_backend_literal:n { color.sc }
\_draw_backend_fillstroke: 1200 \_draw_backend_literal:n { stroke }
\_draw_backend_clip: 1201 \_draw_backend_literal:n { grestore }
\_draw_backend_fillstroke: 1202 \bool_if:NT \g__draw_draw_clip_bool
\_draw_backend_clip: 1203 {
\_draw_backend_fillstroke: 1204 \_draw_backend_literal:x
\_draw_backend_clip: 1205 {
\_draw_backend_fillstroke: 1206 \bool_if:NT \g__draw_draw_eor_bool { eo }
\_draw_backend_fillstroke: 1207 clip
\_draw_backend_fillstroke: 1208 }
\_draw_backend_fillstroke: 1209 }
\_draw_backend_fillstroke: 1210 \_draw_backend_literal:n { newpath }
\_draw_backend_fillstroke: 1211 \bool_gset_false:N \g__draw_draw_clip_bool
\_draw_backend_fillstroke: 1212 }
\_draw_backend_fillstroke: 1213 \cs_new_protected:Npn \_draw_backend_closestroke:
\_draw_backend_fillstroke: 1214 {
\_draw_backend_fillstroke: 1215 \_draw_backend_closepath:
\_draw_backend_fillstroke: 1216 \_draw_backend_stroke:
\_draw_backend_fillstroke: 1217 }
\_draw_backend_fillstroke: 1218 \cs_new_protected:Npn \_draw_backend_fill:
\_draw_backend_fillstroke: 1219 {
\_draw_backend_fillstroke: 1220 \_draw_backend_literal:x
\_draw_backend_fillstroke: 1221 {
\_draw_backend_fillstroke: 1222 \bool_if:NT \g__draw_draw_eor_bool { eo }
\_draw_backend_fillstroke: 1223 fill
\_draw_backend_fillstroke: 1224 }
\_draw_backend_fillstroke: 1225 \bool_if:NT \g__draw_draw_clip_bool
\_draw_backend_fillstroke: 1226 {
\_draw_backend_fillstroke: 1227 \_draw_backend_literal:x
\_draw_backend_fillstroke: 1228 {
\_draw_backend_fillstroke: 1229 \bool_if:NT \g__draw_draw_eor_bool { eo }
\_draw_backend_fillstroke: 1230 clip
\_draw_backend_fillstroke: 1231 }

```

```

1232     }
1233     \_draw_backend_literal:n { newpath }
1234     \bool_gset_false:N \g__draw_draw_clip_bool
1235   }
1236 \cs_new_protected:Npn \_draw_backend_fillstroke:
1237 {
1238   \_draw_backend_literal:x
1239   {
1240     \bool_if:NT \g__draw_draw_eor_bool { eo }
1241     fill
1242   }
1243   \_draw_backend_literal:n { gsave }
1244   \_draw_backend_literal:n { color.sc }
1245   \_draw_backend_literal:n { stroke }
1246   \_draw_backend_literal:n { grestore }
1247   \bool_if:NT \g__draw_draw_clip_bool
1248   {
1249     \_draw_backend_literal:x
1250     {
1251       \bool_if:NT \g__draw_draw_eor_bool { eo }
1252       clip
1253     }
1254   }
1255   \_draw_backend_literal:n { newpath }
1256   \bool_gset_false:N \g__draw_draw_clip_bool
1257 }
1258 \cs_new_protected:Npn \_draw_backend_clip:
1259 { \bool_gset_true:N \g__draw_draw_clip_bool }
1260 \bool_new:N \g__draw_draw_clip_bool
1261 \cs_new_protected:Npn \_draw_backend_discardpath:
1262 {
1263   \bool_if:NT \g__draw_draw_clip_bool
1264   {
1265     \_draw_backend_literal:x
1266     {
1267       \bool_if:NT \g__draw_draw_eor_bool { eo }
1268       clip
1269     }
1270   }
1271   \_draw_backend_literal:n { newpath }
1272   \bool_gset_false:N \g__draw_draw_clip_bool
1273 }

```

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

`_draw_backend_dash_pattern:mn` Converting paths to output is again a case of mapping directly to PostScript operations.

```

\_draw_backend_dash:n 1274 \cs_new_protected:Npn \_draw_backend_dash_pattern:mn #1#2
\_draw_backend_linewidth:n 1275 {
\_draw_backend_miterlimit:n 1276   \_draw_backend_literal:x
\_draw_backend_cap_butt: 1277   {
\_draw_backend_cap_round: 1278     [
  \_draw_backend_cap_rectangle: 1279     \exp_args:Nf \use:n
\_draw_backend_join_miter: 1280     { \clist_map_function:nN {#1} \_draw_backend_dash:n }
\_draw_backend_join_round: 1281     ] ~
\_draw_backend_join_bevel:

```

```

1282     \dim_to_decimal_in_bp:n {#2} ~ setdash
1283   }
1284 }
1285 \cs_new:Npn \__draw_backend_dash:n #1
1286 { ~ \dim_to_decimal_in_bp:n {#1} }
1287 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1288 {
1289   \__draw_backend_literal:x
1290   { \dim_to_decimal_in_bp:n {#1} ~ setlinewidth }
1291 }
1292 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1293 { \__draw_backend_literal:n { #1 ~ setmiterlimit } }
1294 \cs_new_protected:Npn \__draw_backend_cap_but:
1295 { \__draw_backend_literal:n { 0 ~ setlinecap } }
1296 \cs_new_protected:Npn \__draw_backend_cap_round:
1297 { \__draw_backend_literal:n { 1 ~ setlinecap } }
1298 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1299 { \__draw_backend_literal:n { 2 ~ setlinecap } }
1300 \cs_new_protected:Npn \__draw_backend_join_miter:
1301 { \__draw_backend_literal:n { 0 ~ setlinejoin } }
1302 \cs_new_protected:Npn \__draw_backend_join_round:
1303 { \__draw_backend_literal:n { 1 ~ setlinejoin } }
1304 \cs_new_protected:Npn \__draw_backend_join_bevel:
1305 { \__draw_backend_literal:n { 2 ~ setlinejoin } }

```

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

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

```

1306 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1307 {
1308   \__draw_backend_literal:n
1309   { [ #1 ~ #2 ~ #3 ~ #4 ~ 0 ~ 0 ] ~ concat }
1310 }

```

(End definition for `__draw_backend_cm:nnnn`.)

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

```

1311 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1312 {
1313   \__draw_backend_literal:n { @endspecial }

```

```

1314     \_draw_backend_literal:n { [end] }
1315     \_draw_backend_literal:n { [begin] }
1316     \_draw_backend_literal:n { save }
1317     \_draw_backend_literal:n { currentpoint }
1318     \_draw_backend_literal:n { currentpoint~translate }
1319     \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1320     \_draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1321     \_draw_backend_cm:nnnn { 1 } { 0 } { 0 } { -1 }
1322     \_draw_backend_literal:n { neg~exch~neg~exch~translate }
1323     \_draw_backend_literal:n { [end] }
1324     \hbox_overlap_right:n { \box_use:N #1 }
1325     \_draw_backend_literal:n { [begin] }
1326     \_draw_backend_literal:n { restore }
1327     \_draw_backend_literal:n { [end] }
1328     \_draw_backend_literal:n { [begin] }
1329     \_draw_backend_literal:n { @beginspecial }
1330 }

```

(End definition for `_draw_backend_box_use:Nnnnn`.)

```
1331 </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.

```
1332 <{*dvipdfmx | luatex | pdftex | xetex}>
```

4.2.1 Drawing

`_draw_backend_literal:n` Pass data through using a dedicated interface.

```

\_draw_backend_literal:x 1333 \cs_new_eq:NN \_draw_backend_literal:n \_kernel_backend_literal_pdf:n
1334 \cs_generate_variant:Nn \_draw_backend_literal:n { x }

```

(End definition for `_draw_backend_literal:n`.)

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

```

\_draw_backend_end: 1335 \cs_new_protected:Npn \_draw_backend_begin:
1336 { \_draw_backend_scope_begin: }
1337 \cs_new_protected:Npn \_draw_backend_end:
1338 { \_draw_backend_scope_end: }

```

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

`_draw_backend_scope_begin:` Use the backend-level scope mechanisms.

```

\_draw_backend_scope_end: 1339 \cs_new_eq:NN \_draw_backend_scope_begin: \_kernel_backend_scope_begin:
1340 \cs_new_eq:NN \_draw_backend_scope_end: \_kernel_backend_scope_end:

```

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

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

```

\__draw_backend_lineto:nn
  \_draw_backend_curveto:nnnnnn
  \_draw_backend_rectangle:nnnn
1341 \cs_new_protected:Npn \_draw_backend_moveto:nn #1#2
1342 {
1343   \_draw_backend_literal:x
1344   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ m }
1345 }
1346 \cs_new_protected:Npn \_draw_backend_lineto:nn #1#2
1347 {
1348   \_draw_backend_literal:x
1349   { \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~ l }
1350 }
1351 \cs_new_protected:Npn \_draw_backend_curveto:nnnnnn #1#2#3#4#5#6
1352 {
1353   \_draw_backend_literal:x
1354   {
1355     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1356     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1357     \dim_to_decimal_in_bp:n {#5} ~ \dim_to_decimal_in_bp:n {#6} ~
1358     c
1359   }
1360 }
1361 \cs_new_protected:Npn \_draw_backend_rectangle:nnnn #1#2#3#4
1362 {
1363   \_draw_backend_literal:x
1364   {
1365     \dim_to_decimal_in_bp:n {#1} ~ \dim_to_decimal_in_bp:n {#2} ~
1366     \dim_to_decimal_in_bp:n {#3} ~ \dim_to_decimal_in_bp:n {#4} ~
1367     re
1368   }
1369 }

```

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

`_draw_backend_evenodd_rule:` The even-odd rule here can be implemented as a simply switch.
`_draw_backend_nonzero_rule:`
`\g__draw_draw_eor_bool`

```

1370 \cs_new_protected:Npn \_draw_backend_evenodd_rule:
1371 { \bool_gset_true:N \g__draw_draw_eor_bool }
1372 \cs_new_protected:Npn \_draw_backend_nonzero_rule:
1373 { \bool_gset_false:N \g__draw_draw_eor_bool }
1374 \bool_new:N \g__draw_draw_eor_bool

```

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

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

```

\_draw_backend_stroke:
\_draw_backend_closestroke:
  \_draw_backend_fill:
\_draw_backend_fillstroke:
  \_draw_backend_clip:
\_draw_backend_discardpath:
1375 \cs_new_protected:Npn \_draw_backend_closepath:
1376 { \_draw_backend_literal:n { h } }
1377 \cs_new_protected:Npn \_draw_backend_stroke:
1378 { \_draw_backend_literal:n { S } }
1379 \cs_new_protected:Npn \_draw_backend_closestroke:
1380 { \_draw_backend_literal:n { s } }
1381 \cs_new_protected:Npn \_draw_backend_fill:
1382 {
1383   \_draw_backend_literal:x

```

```

1384     { f \bool_if:NT \g__draw_draw_eor_bool * }
1385   }
1386 \cs_new_protected:Npn \__draw_backend_fillstroke:
1387   {
1388     \__draw_backend_literal:x
1389     { B \bool_if:NT \g__draw_draw_eor_bool * }
1390   }
1391 \cs_new_protected:Npn \__draw_backend_clip:
1392   {
1393     \__draw_backend_literal:x
1394     { W \bool_if:NT \g__draw_draw_eor_bool * }
1395   }
1396 \cs_new_protected:Npn \__draw_backend_discardpath:
1397   { \__draw_backend_literal:n { n } }

```

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

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1398 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1399   {
1400     \__draw_backend_literal:x
1401     {
1402       [
1403         \exp_args:Nf \use:n
1404         { \clist_map_function:nN {#1} \__draw_backend_dash:n }
1405       ] ~
1406       \dim_to_decimal_in_bp:n {#2} ~ d
1407     }
1408   }
1409 \cs_new:Npn \__draw_backend_dash:n #1
1410   { ~ \dim_to_decimal_in_bp:n {#1} }
1411 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1412   {
1413     \__draw_backend_literal:x
1414     { \dim_to_decimal_in_bp:n {#1} ~ w }
1415   }
1416 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1417   { \__draw_backend_literal:x { #1 ~ M } }
1418 \cs_new_protected:Npn \__draw_backend_cap_butt:
1419   { \__draw_backend_literal:n { 0 ~ J } }
1420 \cs_new_protected:Npn \__draw_backend_cap_round:
1421   { \__draw_backend_literal:n { 1 ~ J } }
1422 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1423   { \__draw_backend_literal:n { 2 ~ J } }
1424 \cs_new_protected:Npn \__draw_backend_join_miter:
1425   { \__draw_backend_literal:n { 0 ~ j } }
1426 \cs_new_protected:Npn \__draw_backend_join_round:
1427   { \__draw_backend_literal:n { 1 ~ j } }
1428 \cs_new_protected:Npn \__draw_backend_join_bevel:
1429   { \__draw_backend_literal:n { 2 ~ j } }

```

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

`__draw_backend_cm:nnnn` Another split here between LuaTeX/pdfTeX and dvipdfmx/X_YTeX. In the former, we have a direct method to maintain alignment: the backend can use a matrix itself. For

`dvipdfmx/XYTeX`, 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/XYTeX`, 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!

```

1430 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1431 {
1432 <*luatex | pdftex>
1433   \__kernel_backend_matrix:n { #1 ~ #2 ~ #3 ~ #4 }
1434 </luatex | pdftex>
1435 <*dvipdfmx | xetex>
1436   \__draw_backend_cm_decompose:nnnnN {#1} {#2} {#3} {#4}
1437   \__draw_backend_cm_aux:nnnn
1438 </dvipdfmx | xetex>
1439 }
1440 <*dvipdfmx | xetex>
1441 \cs_new_protected:Npn \__draw_backend_cm_aux:nnnn #1#2#3#4
1442 {
1443   \__kernel_backend_literal:x
1444   {
1445     x:rotate~
1446     \fp_compare:nNnTF {#1} = \c_zero_fp
1447     { 0 }
1448     { \fp_eval:n { round ( -#1 , 5 ) } } }
1449   }
1450   \__kernel_backend_literal:x
1451   {
1452     x:scale~
1453     \fp_eval:n { round ( #2 , 5 ) } ~
1454     \fp_eval:n { round ( #3 , 5 ) }
1455   }
1456   \__kernel_backend_literal:x
1457   {
1458     x:rotate~
1459     \fp_compare:nNnTF {#4} = \c_zero_fp
1460     { 0 }
1461     { \fp_eval:n { round ( -#4 , 5 ) } } }
1462   }
1463 }
1464 </dvipdfmx | xetex>

```

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

```

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

```

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

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

The parent matrix can be converted to

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

From these, we can find that

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

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

```

1465 <*dviptdpmx | xetex>
1466 \cs_new_protected:Npn \__draw_backend_cm_decompose:nnnnN #1#2#3#4#5
1467 {
1468   \use:x
1469   {
1470     \__draw_backend_cm_decompose_auxi:nnnnN
1471     { \fp_eval:n { (#1 + #4) / 2 } }
1472     { \fp_eval:n { (#1 - #4) / 2 } }
1473     { \fp_eval:n { (#3 + #2) / 2 } }
1474     { \fp_eval:n { (#3 - #2) / 2 } }
1475   }
1476   #5
1477 }
1478 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxi:nnnnN #1#2#3#4#5
1479 {
1480   \use:x
1481   {
1482     \__draw_backend_cm_decompose_auxii:nnnnN
1483     { \fp_eval:n { 2 * sqrt ( #1 * #1 + #4 * #4 ) } }
1484     { \fp_eval:n { 2 * sqrt ( #2 * #2 + #3 * #3 ) } }
1485     { \fp_eval:n { atand ( #3 , #2 ) } }
1486     { \fp_eval:n { atand ( #4 , #1 ) } }
1487   }
1488   #5
1489 }
1490 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxii:nnnnN #1#2#3#4#5
1491 {
1492   \use:x
1493   {
1494     \__draw_backend_cm_decompose_auxiii:nnnnN
1495     { \fp_eval:n { ( #4 - #3 ) / 2 } }
1496     { \fp_eval:n { ( #1 + #2 ) / 2 } }
1497     { \fp_eval:n { ( #1 - #2 ) / 2 } }
1498     { \fp_eval:n { ( #4 + #3 ) / 2 } }
1499   }

```



```

1500         #5
1501     }
1502 \cs_new_protected:Npn \__draw_backend_cm_decompose_auxiii:nnnnN #1#2#3#4#5
1503 {
1504     \fp_compare:nNnTF { abs( #2 ) } > { abs ( #3 ) }
1505         { #5 {#1} {#2} {#3} {#4} }
1506         { #5 {#1} {#3} {#2} {#4} }
1507     }
1508 </dvipdfmx | xetex>

```

(End definition for `__draw_backend_cm_decompose:nnnnN` and others.)

`__draw_backend_box_use:Nnnnn`

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

```

1509 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1510 {
1511     \__kernel_backend_scope_begin:
1512 <*luatex | pdftex>
1513     \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1514 </luatex | pdftex>
1515 <*dvipdfmx | xetex>
1516     \__kernel_backend_literal:n
1517         { pdf:btrans~matrix~ #2 ~ #3 ~ #4 ~ #5 ~ 0 ~ 0 }
1518 </dvipdfmx | xetex>
1519     \hbox_overlap_right:n { \box_use:N #1 }
1520 <*dvipdfmx | xetex>
1521     \__kernel_backend_literal:n { pdf:etrans }
1522 </dvipdfmx | xetex>
1523     \__kernel_backend_scope_end:
1524 }

```

(End definition for `__draw_backend_box_use:Nnnnn`.)

```
1525 </dvipdfmx | luatex | pdftex | xetex>
```

4.3 dvisvgm backend

```
1526 <*dvisvgm>
```

The same as the more general literal call.

`__draw_backend_literal:n`
`__draw_backend_literal:x`

```

1527 \cs_new_eq:NN \__draw_backend_literal:n \__kernel_backend_literal_svg:n
1528 \cs_generate_variant:Nn \__draw_backend_literal:n { x }

```

(End definition for `__draw_backend_literal:n`.)

Use the backend-level scope mechanisms.

`__draw_backend_scope_begin:`
`__draw_backend_scope_end:`

```

1529 \cs_new_eq:NN \__draw_backend_scope_begin: \__kernel_backend_scope_begin:
1530 \cs_new_eq:NN \__draw_backend_scope_end: \__kernel_backend_scope_end:

```

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

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

```

1531 \cs_new_protected:Npn \__draw_backend_begin:
1532 {
1533   \__kernel_backend_scope_begin:
1534   \__kernel_backend_scope:n { transform="translate({?x},{?y})~scale(1,-1)" }
1535 }
1536 \cs_new_eq:NN \__draw_backend_end: \__kernel_backend_scope_end:

```

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

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

```

1537 \cs_new_protected:Npn \__draw_backend_moveto:nn #1#2
1538 {
1539   \__draw_backend_add_to_path:n
1540   { M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1541 }
1542 \cs_new_protected:Npn \__draw_backend_lineto:nn #1#2
1543 {
1544   \__draw_backend_add_to_path:n
1545   { L ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} }
1546 }
1547 \cs_new_protected:Npn \__draw_backend_rectangle:nmmn #1#2#3#4
1548 {
1549   \__draw_backend_add_to_path:n
1550   {
1551     M ~ \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2}
1552     h ~ \dim_to_decimal:n {#3} ~
1553     v ~ \dim_to_decimal:n {#4} ~
1554     h ~ \dim_to_decimal:n { -#3 } ~
1555     Z
1556   }
1557 }
1558 \cs_new_protected:Npn \__draw_backend_curveto:nmmmn #1#2#3#4#5#6
1559 {
1560   \__draw_backend_add_to_path:n
1561   {
1562     C ~
1563     \dim_to_decimal:n {#1} ~ \dim_to_decimal:n {#2} ~
1564     \dim_to_decimal:n {#3} ~ \dim_to_decimal:n {#4} ~
1565     \dim_to_decimal:n {#5} ~ \dim_to_decimal:n {#6}
1566   }
1567 }
1568 \cs_new_protected:Npn \__draw_backend_add_to_path:n #1
1569 {
1570   \tl_gset:Nx \g__draw_backend_path_tl
1571   {
1572     \g__draw_backend_path_tl
1573     \tl_if_empty:NF \g__draw_backend_path_tl { \c_space_tl }
1574     #1

```

```

1575     }
1576   }
1577   \tl_new:N \g__draw_backend_path_tl

```

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

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

```

1578   \cs_new_protected:Npn \__draw_backend_evenodd_rule:
1579     { \__kernel_backend_scope:n { fill-rule="evenodd" } }
1580   \cs_new_protected:Npn \__draw_backend_nonzero_rule:
1581     { \__kernel_backend_scope:n { fill-rule="nonzero" } }

```

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

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

```

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

```

```

1582   \cs_new_protected:Npn \__draw_backend_closepath:
1583     { \__draw_backend_add_to_path:n { Z } }
1584   \cs_new_protected:Npn \__draw_backend_path:n #1
1585     {
1586       \bool_if:NTF \g__draw_draw_clip_bool
1587         {
1588           \int_gincr:N \g__kernel_clip_path_int
1589           \__draw_backend_literal:x
1590             {
1591               < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1592                 { ?nl }
1593               <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1594               < /clipPath > { ? nl }
1595               <
1596                 use~xlink:href =
1597                   "\c_hash_str l3path \int_use:N \g__draw_backend_path_int " ~
1598                   #1
1599               />
1600             }
1601           \__kernel_backend_scope:x
1602             {
1603               clip-path =
1604                 "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1605             }
1606         }
1607       {
1608         \__draw_backend_literal:x
1609           { <path ~ d=" \g__draw_backend_path_tl " ~ #1 /> }
1610       }
1611       \tl_gclear:N \g__draw_backend_path_tl
1612       \bool_gset_false:N \g__draw_draw_clip_bool
1613     }
1614   \int_new:N \g__draw_backend_path_int
1615   \cs_new_protected:Npn \__draw_backend_stroke:
1616     { \__draw_backend_path:n { style="fill:none" } }

```

```

1617 \cs_new_protected:Npn \__draw_backend_closestroke:
1618 {
1619   \__draw_backend_closepath:
1620   \__draw_backend_stroke:
1621 }
1622 \cs_new_protected:Npn \__draw_backend_fill:
1623 { \__draw_backend_path:n { style="stroke:none" } }
1624 \cs_new_protected:Npn \__draw_backend_fillstroke:
1625 { \__draw_backend_path:n { } }
1626 \cs_new_protected:Npn \__draw_backend_clip:
1627 { \bool_gset_true:N \g__draw_draw_clip_bool }
1628 \bool_new:N \g__draw_draw_clip_bool
1629 \cs_new_protected:Npn \__draw_backend_discardpath:
1630 {
1631   \bool_if:NT \g__draw_draw_clip_bool
1632   {
1633     \int_gincr:N \g__kernel_clip_path_int
1634     \__draw_backend_literal:x
1635     {
1636       < clipPath~id = " l3cp \int_use:N \g__kernel_clip_path_int " >
1637       { ?nl }
1638       <path~d=" \g__draw_backend_path_tl "/> { ?nl }
1639       < /clipPath >
1640     }
1641     \__kernel_backend_scope:x
1642     {
1643       clip-path =
1644       "url( \c_hash_str l3cp \int_use:N \g__kernel_clip_path_int)"
1645     }
1646   }
1647   \tl_gclear:N \g__draw_path_tl
1648   \bool_gset_false:N \g__draw_draw_clip_bool
1649 }

```

(End definition for __draw_backend_path:n and others.)

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

```

\__draw_backend_dash_pattern:nn
\__draw_backend_dash:n
\__draw_backend_dash_aux:nn
\__draw_backend_linewidth:n
\__draw_backend_miterlimit:n
\__draw_backend_cap_butt:
\__draw_backend_cap_round:
\__draw_backend_cap_rectangle:
\__draw_backend_join_miter:
\__draw_backend_join_round:
\__draw_backend_join_bevel:
1650 \cs_new_protected:Npn \__draw_backend_dash_pattern:nn #1#2
1651 {
1652   \use:x
1653   {
1654     \__draw_backend_dash_aux:nn
1655     { \clist_map_function:nn {#1} \__draw_backend_dash:n }
1656     { \dim_to_decimal:n {#2} }
1657   }
1658 }
1659 \cs_new:Npn \__draw_backend_dash:n #1
1660 { , \dim_to_decimal_in_bp:n {#1} }
1661 \cs_new_protected:Npn \__draw_backend_dash_aux:nn #1#2
1662 {
1663   \__kernel_backend_scope:x
1664   {
1665     stroke-dasharray =

```

```

1666     "
1667     \tl_if_empty:nTF {#1}
1668     { none }
1669     { \use_none:n #1 }
1670     " ~
1671     stroke-offset=" #2 "
1672   }
1673 }
1674 \cs_new_protected:Npn \__draw_backend_linewidth:n #1
1675 { \__kernel_backend_scope:x { stroke-width=" \dim_to_decimal:n {#1} " } }
1676 \cs_new_protected:Npn \__draw_backend_miterlimit:n #1
1677 { \__kernel_backend_scope:x { stroke-miterlimit=" #1 " } }
1678 \cs_new_protected:Npn \__draw_backend_cap_but:
1679 { \__kernel_backend_scope:n { stroke-linecap="butt" } }
1680 \cs_new_protected:Npn \__draw_backend_cap_round:
1681 { \__kernel_backend_scope:n { stroke-linecap="round" } }
1682 \cs_new_protected:Npn \__draw_backend_cap_rectangle:
1683 { \__kernel_backend_scope:n { stroke-linecap="square" } }
1684 \cs_new_protected:Npn \__draw_backend_join_miter:
1685 { \__kernel_backend_scope:n { stroke-linejoin="miter" } }
1686 \cs_new_protected:Npn \__draw_backend_join_round:
1687 { \__kernel_backend_scope:n { stroke-linejoin="round" } }
1688 \cs_new_protected:Npn \__draw_backend_join_bevel:
1689 { \__kernel_backend_scope:n { stroke-linejoin="bevel" } }

```

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

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

```

1690 \cs_new_protected:Npn \__draw_backend_cm:nnnn #1#2#3#4
1691 {
1692   \__kernel_backend_scope:n
1693   {
1694     transform =
1695     " matrix ( #1 , #2 , #3 , #4 , Opt , Opt ) "
1696   }
1697 }

```

(End definition for `__draw_backend_cm:nnnn`.)

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

```

1698 \cs_new_protected:Npn \__draw_backend_box_use:Nnnnn #1#2#3#4#5
1699 {
1700   \__kernel_backend_scope_begin:
1701   \__draw_backend_cm:nnnn {#2} {#3} {#4} {#5}
1702   \__kernel_backend_literal_svg:n
1703   {
1704     < g~
1705     stroke="none"~
1706     transform="scale(-1,1)~translate({?x},{?y})~scale(-1,-1)"
1707     >
1708   }
1709   \box_set_wd:Nn #1 { Opt }

```

```

1710     \box_set_ht:Nn #1 { Opt }
1711     \box_set_dp:Nn #1 { Opt }
1712     \box_use:N #1
1713     \__kernel_backend_literal_svg:n { </g> }
1714     \__kernel_backend_scope_end:
1715 }

```

(End definition for __draw_backend_box_use:Nnnnn.)

```
1716 </dvisvgm>
```

```
1717 </package>
```

5 l3backend-graphics implementation

```

1718 <*package>
1719 <@=graphics>

```

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

```

1720 \cs_new_protected:Npn \__graphics_backend_loaded:n #1
1721 {
1722     \cs_if_exist:NTF \hook_gput_code:nnn
1723     {
1724         \hook_gput_code:nnn
1725         { package / l3graphics / after }
1726         { backend }
1727         {#1}
1728     }
1729     {#1}
1730 }

```

(End definition for __graphics_backend_loaded:n.)

5.1 dvips backend

```
1731 <*dvips>
```

\l_graphics_search_ext_seq

```

1732 \__graphics_backend_loaded:n
1733 { \seq_set_from_clist:Nn \l_graphics_search_ext_seq { .eps , .ps } }

```

(End definition for \l_graphics_search_ext_seq. This variable is documented on page ??.)

_graphics_backend_getbb_eps:n Simply use the generic function.

```

\__graphics_backend_getbb_ps:n
1734 \__graphics_backend_loaded:n
1735 {
1736     \cs_new_eq:NN \__graphics_backend_getbb_eps:n \__graphics_read_bb:n
1737     \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_read_bb:n
1738 }

```

(End 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
1739 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1740 {
1741   \__kernel_backend_literal:x
1742   {
1743     PSfile = #1 \c_space_tl
1744     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1745     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1746     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1747     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
1748   }
1749 }
1750 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n

```

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

```

\__graphics_backend_get_pagecount:n
1751 \__graphics_backend_loaded:n
1752 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }

```

(End definition for `__graphics_backend_get_pagecount:n`.)

```

1753 </dvips>

```

5.2 LuaTeX and pdfTeX backends

```

1754 < *luatex | pdftex >

```

```

\l__graphics_search_ext_seq

```

```

1755 \__graphics_backend_loaded:n
1756 {
1757   \seq_set_from_clist:Nn
1758   \l__graphics_search_ext_seq
1759   { .pdf , .eps , .ps , .png , .jpg , .jpeg }
1760 }

```

(End definition for `\l__graphics_search_ext_seq`. This variable is documented on page ??.)

```

\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.

```

1761 \tl_new:N \l__graphics_attr_tl

```

(End 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.

```

1762 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
1763 {
1764   \int_zero:N \l__graphics_page_int

```

```

1765 \tl_clear:N \l__graphics_pagebox_tl
1766 \tl_set:Nx \l__graphics_attr_tl
1767 {
1768   \tl_if_empty:NF \l__graphics_decodearray_str
1769   { :D \l__graphics_decodearray_str }
1770   \bool_if:NT \l__graphics_interpolate_bool
1771   { :I }
1772   \str_if_empty:NF \l__graphics_pdf_str
1773   { :X \l__graphics_pdf_str }
1774 }
1775 \__graphics_backend_getbb_auxi:n {#1}
1776 }
1777 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
1778 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
1779 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
1780 {
1781   \tl_clear:N \l__graphics_decodearray_str
1782   \bool_set_false:N \l__graphics_interpolate_bool
1783   \tl_set:Nx \l__graphics_attr_tl
1784   {
1785     : \l__graphics_pagebox_tl
1786     \int_compare:nNnT \l__graphics_page_int > 1
1787     { :P \int_use:N \l__graphics_page_int }
1788     \str_if_empty:NF \l__graphics_pdf_str
1789     { :X \l__graphics_pdf_str }
1790   }
1791   \__graphics_backend_getbb_auxi:n {#1}
1792 }
1793 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:n #1
1794 {
1795   \__graphics_bb_restore:xF { #1 \l__graphics_attr_tl }
1796   { \__graphics_backend_getbb_auxii:n {#1} }
1797 }

```

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.

```

1798 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:n #1
1799 {
1800   \exp_args:Ne \__graphics_backend_getbb_auxiii:n
1801   { \__graphics_backend_dequote:w #1 " #1 " \s__graphics_stop }
1802   \int_const:cn { c__graphics_ #1 \l__graphics_attr_tl_int }
1803   { \tex_the:D \tex_pdflastximage:D }
1804   \__graphics_bb_save:x { #1 \l__graphics_attr_tl }
1805 }
1806 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:n #1
1807 {
1808   \tex_immediate:D \tex_pdfximage:D
1809   \bool_lazy_any:nT
1810   {
1811     { \l__graphics_interpolate_bool }
1812     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1813     { ! \str_if_empty_p:N \l__graphics_pdf_str }

```



```

1814     }
1815     {
1816         attr ~
1817         {
1818             \tl_if_empty:NF \l__graphics_decodearray_str
1819             { /Decode~[ \l__graphics_decodearray_str ] }
1820             \bool_if:NT \l__graphics_interpolate_bool
1821             { /Interpolate~true }
1822             \l__graphics_pdf_str
1823         }
1824     }
1825     \int_compare:nNnT \l__graphics_page_int > 0
1826     { page ~ \int_use:N \l__graphics_page_int }
1827     \tl_if_empty:NF \l__graphics_pagebox_tl
1828     { \l__graphics_pagebox_tl }
1829     {#1}
1830     \hbox_set:Nn \l__graphics_internal_box
1831     { \tex_pdfrefximage:D \tex_pdflastximage:D }
1832     \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
1833     \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
1834 }
1835 \cs_new:Npn \__graphics_backend_dequote:w #1 " #2 " #3 \s__graphics_stop {#2}

```

(End 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.

```

1836 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1837 {
1838     \tex_pdfrefximage:D
1839     \int_use:c { c__graphics_ #1 \l__graphics_attr_tl _int }
1840 }
1841 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1842 \cs_new_eq:NN \__graphics_backend_include_pdf:n \__graphics_backend_include_jpg:n
1843 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n

```

(End 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_EX 2_ε package, but simplified, conversion takes place here if we have shell access.

```

1844 \sys_if_shell:T
1845 {
1846     \str_new:N \l__graphics_backend_dir_str
1847     \str_new:N \l__graphics_backend_name_str
1848     \str_new:N \l__graphics_backend_ext_str
1849     \cs_new_protected:Npn \__graphics_backend_getbb_eps:n #1
1850     {
1851         \file_parse_full_name:nNNN {#1}
1852         \l__graphics_backend_dir_str
1853         \l__graphics_backend_name_str
1854         \l__graphics_backend_ext_str
1855         \exp_args:Nx \__graphics_backend_getbb_eps:nm

```

```

1856     {
1857         \exp_args:Ne \__kernel_file_name_quote:n
1858         {
1859             \l__graphics_backend_name_str
1860             - \str_tail:N \l__graphics_backend_ext_str
1861             -converted-to.pdf
1862         }
1863     }
1864     {#1}
1865 }
1866 \cs_new_eq:NN \__graphics_backend_getbb_ps:n \__graphics_backend_getbb_eps:n
1867 \cs_new_protected:Npn \__graphics_backend_getbb_eps:nn #1#2
1868 {
1869     \file_compare_timestamp:nNnT {#2} > {#1}
1870     {
1871         \sys_shell_now:n
1872         { repstopdf ~ #2 ~ #1 }
1873     }
1874     \tl_set:Nn \l__graphics_final_name_str {#1}
1875     \__graphics_backend_getbb_pdf:n {#1}
1876 }
1877 \cs_new_protected:Npn \__graphics_backend_include_eps:n #1
1878 {
1879     \file_parse_full_name:nNNN {#1}
1880     \l__graphics_backend_dir_str \l__graphics_backend_name_str \l__graphics_backend_ext_str
1881     \exp_args:Nx \__graphics_backend_include_pdf:n
1882     {
1883         \exp_args:Ne \__kernel_file_name_quote:n
1884         {
1885             \l__graphics_backend_name_str
1886             - \str_tail:N \l__graphics_backend_ext_str
1887             -converted-to.pdf
1888         }
1889     }
1890 }
1891 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1892 }

```

(End definition for __graphics_backend_getbb_eps:n and others.)

__graphics_backend_get_pagecount:n Simply load and store.

```

1893 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
1894 {
1895     \tex_pdfximage:D {#1}
1896     \int_const:cn { c__graphics_ #1 _pages_int }
1897     { \int_use:N \tex_pdflastximagepages:D }
1898 }

```

(End definition for __graphics_backend_get_pagecount:n.)

1899 </luatex | pdftex>

5.3 dvipdfmx backend

1900 <*dvipdfmx | xetex>

`\l_graphics_search_ext_seq`

```
1901 \l_graphics_backend_loaded:n
1902 {
1903   \seq_set_from_clist:Nn \l_graphics_search_ext_seq
1904   { .pdf , .eps , .ps , .png , .jpg , .jpeg , .bmp }
1905 }
```

(End definition for `\l_graphics_search_ext_seq`. This variable is documented on page ??.)

`_graphics_backend_getbb_eps:n`

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

`_graphics_backend_getbb_ps:n`

```
1906 \l_graphics_backend_loaded:n
```

`_graphics_backend_getbb_jpg:n`

```
1907 {
```

`_graphics_backend_getbb_jpeg:n`

```
1908   \cs_new_eq:NN \_graphics_backend_getbb_eps:n \_graphics_read_bb:n
```

`_graphics_backend_getbb_pdf:n`

```
1909   \cs_new_eq:NN \_graphics_backend_getbb_ps:n \_graphics_read_bb:n
```

`_graphics_backend_getbb_png:n`

```
1910 }
```

`_graphics_backend_getbb_bmp:n`

```
1911 <*dvipdfmx>
```

```
1912 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
```

```
1913 {
```

```
1914   \int_zero:N \l__graphics_page_int
```

```
1915   \tl_clear:N \l__graphics_pagebox_tl
```

```
1916   \_graphics_extract_bb:n {#1}
```

```
1917 }
```

```
1918 \cs_new_eq:NN \_graphics_backend_getbb_jpeg:n \_graphics_backend_getbb_jpg:n
```

```
1919 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n
```

```
1920 \cs_new_eq:NN \_graphics_backend_getbb_bmp:n \_graphics_backend_getbb_jpg:n
```

```
1921 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
```

```
1922 {
```

```
1923   \tl_clear:N \l__graphics_decodearray_str
```

```
1924   \bool_set_false:N \l__graphics_interpolate_bool
```

```
1925   \_graphics_extract_bb:n {#1}
```

```
1926 }
```

```
1927 </dvipdfmx>
```

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

`\g__graphics_track_int`

Used to track the object number associated with each graphic.

```
1928 \int_new:N \g__graphics_track_int
```

(End definition for `\g__graphics_track_int`.)

`_graphics_backend_include_eps:n`

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

`_graphics_backend_include_ps:n`

`_graphics_backend_include_jpg:n`

`_graphics_backend_include_jpeg:n`

`_graphics_backend_include_pdf:n`

```
1929 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
```

`_graphics_backend_include_png:n`

```
1930 {
```

`_graphics_backend_include_bmp:n`

```
1931   \__kernel_backend_literal:x
```

```
1932   {
```

`_graphics_backend_include_auxi:nn`

```
1933     PSfile = #1 \c_space_tl
```

`_graphics_backend_include_auxii:nnn`

```
1934     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
```

`_graphics_backend_include_auxiii:nnn`

```
1935     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
```

`_graphics_backend_include_auxiiii:nnn`

```
1936     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
```

```
1937     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
```

```
1938   }
```

```
1939 }
```

```

1940 \cs_new_eq:NN \__graphics_backend_include_ps:n \__graphics_backend_include_eps:n
1941 \cs_new_protected:Npn \__graphics_backend_include_jpg:n #1
1942   { \__graphics_backend_include_auxi:nn {#1} { image } }
1943 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_jpg:n
1944 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_jpg:n
1945 \cs_new_eq:NN \__graphics_backend_include_bmp:n \__graphics_backend_include_jpg:n
1946 <*dvipdfmx>
1947 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1
1948   { \__graphics_backend_include_auxi:nn {#1} { epdf } }
1949 </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.

```

1950 \cs_new_protected:Npn \__graphics_backend_include_auxi:nn #1#2
1951   {
1952     \__graphics_backend_include_auxii:xnn
1953     {
1954       \tl_if_empty:NF \l__graphics_pagebox_tl
1955       { : \l__graphics_pagebox_tl }
1956       \int_compare:nNnT \l__graphics_page_int > 1
1957       { :P \int_use:N \l__graphics_page_int }
1958       \tl_if_empty:NF \l__graphics_decodearray_str
1959       { :D \l__graphics_decodearray_str }
1960       \bool_if:NT \l__graphics_interpolate_bool
1961       { :I }
1962     }
1963     {#1} {#2}
1964   }
1965 \cs_new_protected:Npn \__graphics_backend_include_auxii:nnn #1#2#3
1966   {
1967     \int_if_exist:cTF { c__graphics_ #2#1 _int }
1968     {
1969       \__kernel_backend_literal:x
1970       { pdf:usexobj~@graphic \int_use:c { c__graphics_ #2#1 _int } }
1971     }
1972     { \__graphics_backend_include_auxiii:nnn {#2} {#1} {#3} }
1973   }
1974 \cs_generate_variant:Nn \__graphics_backend_include_auxii:nnn { x }

```

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

```

1975 \cs_new_protected:Npn \__graphics_backend_include_auxiii:nnn #1#2#3
1976   {
1977     \int_gincr:N \g__graphics_track_int
1978     \int_const:cn { c__graphics_ #1#2 _int } { \g__graphics_track_int }
1979     \__kernel_backend_literal:x
1980     {
1981       pdf:#3~
1982       @graphic \int_use:c { c__graphics_ #1#2 _int } ~
1983       \int_compare:nNnT \l__graphics_page_int > 1
1984       { page ~ \int_use:N \l__graphics_page_int \c_space_tl }

```

```

1985     \tl_if_empty:NF \l__graphics_pagebox_tl
1986     {
1987         pagebox ~ \l__graphics_pagebox_tl \c_space_tl
1988         bbox ~
1989             \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
1990             \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
1991             \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
1992             \dim_to_decimal_in_bp:n \l__graphics_ury_dim \c_space_tl
1993     }
1994     (#1)
1995     \bool_lazy_or:nnT
1996     { \l__graphics_interpolate_bool }
1997     { ! \tl_if_empty_p:N \l__graphics_decodearray_str }
1998     {
1999         <<
2000         \tl_if_empty:NF \l__graphics_decodearray_str
2001         { /Decode~[ \l__graphics_decodearray_str ] }
2002         \bool_if:NT \l__graphics_interpolate_bool
2003         { /Interpolate~true }
2004         >>
2005     }
2006 }
2007 }

```

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

`__graphics_backend_get_pagecount:n`

```

2008 < *dvipdfmx >
2009 \__graphics_backend_loaded:n
2010 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }
2011 < /dvipdfmx >

```

(End definition for `__graphics_backend_get_pagecount:n`.)

```
2012 < /dvipdfmx | xetex >
```

5.4 X_YTeX backend

```
2013 < *xetex >
```

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

```

\__graphics_backend_getbb_jpg:n
\__graphics_backend_getbb_jpeg:n
\__graphics_backend_getbb_pdf:n
\__graphics_backend_getbb_png:n
\__graphics_backend_getbb_bmp:n
\__graphics_backend_getbb_auxi:nN
\__graphics_backend_getbb_auxii:nnN
\__graphics_backend_getbb_auxiii:VnN
\__graphics_backend_getbb_auxiiii:nnNn
\__graphics_backend_getbb_auxiv:nnNn
\__graphics_backend_getbb_auxiv:VnNn
\__graphics_backend_getbb_auxv:nnNn
\__graphics_backend_getbb_auxv:nnNn
\__graphics_backend_getbb_pagebox:w

```

```

2014 \cs_new_protected:Npn \__graphics_backend_getbb_jpg:n #1
2015 {
2016     \int_zero:N \l__graphics_page_int
2017     \tl_clear:N \l__graphics_pagebox_tl
2018     \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpicfile:D
2019 }
2020 \cs_new_eq:NN \__graphics_backend_getbb_jpeg:n \__graphics_backend_getbb_jpg:n
2021 \cs_new_eq:NN \__graphics_backend_getbb_png:n \__graphics_backend_getbb_jpg:n
2022 \cs_new_eq:NN \__graphics_backend_getbb_bmp:n \__graphics_backend_getbb_jpg:n
2023 \cs_new_protected:Npn \__graphics_backend_getbb_pdf:n #1
2024 {

```

```

2025 \tl_clear:N \l__graphics_decodearray_str
2026 \bool_set_false:N \l__graphics_interpolate_bool
2027 \__graphics_backend_getbb_auxi:nN {#1} \tex_XeTeXpdffile:D
2028 }
2029 \cs_new_protected:Npn \__graphics_backend_getbb_auxi:nN #1#2
2030 {
2031   \int_compare:nNnTF \l__graphics_page_int > 1
2032     { \__graphics_backend_getbb_auxii:VnN \l__graphics_page_int {#1} #2 }
2033     { \__graphics_backend_getbb_auxiii:nNnn {#1} #2 { :P 1 } { page 1 } }
2034 }
2035 \cs_new_protected:Npn \__graphics_backend_getbb_auxii:nnN #1#2#3
2036 { \__graphics_backend_getbb_auxiii:nNnn {#2} #3 { :P #1 } { page #1 } }
2037 \cs_generate_variant:Nn \__graphics_backend_getbb_auxii:nnN { V }
2038 \cs_new_protected:Npn \__graphics_backend_getbb_auxiii:nNnn #1#2#3#4
2039 {
2040   \tl_if_empty:NTF \l__graphics_pagebox_tl
2041     { \__graphics_backend_getbb_auxiv:VnNnn \l__graphics_pagebox_tl }
2042     { \__graphics_backend_getbb_auxv:nNnn }
2043     {#1} #2 {#3} {#4}
2044 }
2045 \cs_new_protected:Npn \__graphics_backend_getbb_auxiv:nnNnn #1#2#3#4#5
2046 {
2047   \use:x
2048   {
2049     \__graphics_backend_getbb_auxv:nNnn {#2} #3 { : #1 #4 }
2050     {
2051       #5
2052       \tl_if_blank:nF {#1}
2053         { \c_space_tl \__graphics_backend_getbb_pagebox:w #1 }
2054     }
2055   }
2056 }
2057 \cs_generate_variant:Nn \__graphics_backend_getbb_auxiv:nnNnn { V }
2058 \cs_new_protected:Npn \__graphics_backend_getbb_auxv:nNnn #1#2#3#4
2059 {
2060   \__graphics_bb_restore:nF {#1#3}
2061   { \__graphics_backend_getbb_auxvi:nNnn {#1} #2 {#3} {#4} }
2062 }
2063 \cs_new_protected:Npn \__graphics_backend_getbb_auxvi:nNnn #1#2#3#4
2064 {
2065   \hbox_set:Nn \l__graphics_internal_box { #2 #1 ~ #4 }
2066   \dim_set:Nn \l__graphics_urx_dim { \box_wd:N \l__graphics_internal_box }
2067   \dim_set:Nn \l__graphics_ury_dim { \box_ht:N \l__graphics_internal_box }
2068   \__graphics_bb_save:n {#1#3}
2069 }
2070 \cs_new:Npn \__graphics_backend_getbb_pagebox:w #1 box {#1}

```

(End definition for __graphics_backend_getbb_jpg:n and others.)

__graphics_backend_include_pdf:n 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.

```

2071 \cs_new_protected:Npn \__graphics_backend_include_pdf:n #1

```

```

2072 {
2073   \tex_XeTeXpdffile:D #1 ~
2074   \int_compare:nNnT \l__graphics_page_int > 0
2075     { page ~ \int_use:N \l__graphics_page_int \c_space_tl }
2076     \exp_after:wN \__graphics_backend_getbb_pagebox:w \l__graphics_pagebox_tl
2077 }

```

(End 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.

```

2078 \cs_new_protected:Npn \__graphics_backend_get_pagecount:n #1
2079 {
2080   \int_const:cn { c__graphics_#1_pages_int }
2081   {
2082     \int_max:nn
2083       { \int_use:N \tex_XeTeXpdfpagecount:D #1 ~ }
2084       { 1 }
2085   }
2086 }

```

(End definition for `__graphics_backend_get_pagecount:n`.)

```
2087 </xetex>
```

5.5 dvisvgm backend

```
2088 <*dvisvgm>
```

`\l_graphics_search_ext_seq`

```

2089 \__graphics_backend_loaded:n
2090 {
2091   \seq_set_from_clist:Nn
2092     \l_graphics_search_ext_seq
2093     { .svg , .pdf , .eps , .ps , .png , .jpg , .jpeg }
2094 }

```

(End definition for `\l_graphics_search_ext_seq`. This variable is documented on page ??.)

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

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

```

2095 \cs_new_protected:Npn \__graphics_backend_getbb_svg:n #1
2096 {
2097   \__graphics_bb_restore:nF {#1}
2098   {
2099     \ior_open:Nn \l__graphics_internal_ior {#1}
2100     \ior_if_eof:NTF \l__graphics_internal_ior
2101       { \msg_error:nnn { graphics } { graphic-not-found } {#1} }
2102       {
2103         \dim_zero:N \l__graphics_llx_dim
2104         \dim_zero:N \l__graphics_lly_dim
2105         \dim_set:Nn \l__graphics_urx_dim { -\c_max_dim }
2106         \dim_set:Nn \l__graphics_ury_dim { -\c_max_dim }
2107         \ior_str_map_inline:Nn \l__graphics_internal_ior

```

```

2108     {
2109         \dim_compare:nNnT \l__graphics_urx_dim = { -\c_max_dim }
2110         {
2111             \__graphics_backend_getbb_svg_auxi:nNn
2112             { width } \l__graphics_urx_dim {##1}
2113         }
2114         \dim_compare:nNnT \l__graphics_ury_dim = { -\c_max_dim }
2115         {
2116             \__graphics_backend_getbb_svg_auxi:nNn
2117             { height } \l__graphics_ury_dim {##1}
2118         }
2119         \bool_lazy_and:nnF
2120         { \dim_compare_p:nNn \l__graphics_urx_dim = { -\c_max_dim } }
2121         { \dim_compare_p:nNn \l__graphics_ury_dim = { -\c_max_dim } }
2122         { \ior_map_break: }
2123     }
2124     \__graphics_bb_save:n {##1}
2125 }
2126 \ior_close:N \l__graphics_internal_ior
2127 }
2128 }
2129 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxi:nNn #1#2#3
2130 {
2131     \use:x
2132     {
2133         \cs_set_protected:Npn \__graphics_backend_getbb_svg_auxii:w
2134         ###1 \tl_to_str:n {#1} = ###2 \tl_to_str:n {#1} = ###3
2135         \s__graphics_stop
2136     }
2137     {
2138         \tl_if_blank:nF {##2}
2139         {
2140             \peek_remove_spaces:n
2141             {
2142                 \peek_meaning:NTF ' % '
2143                 { \__graphics_backend_getbb_svg_auxiii:Nw #2 }
2144                 {
2145                     \peek_meaning:NTF " % "
2146                     { \__graphics_backend_getbb_svg_auxiv:Nw #2 }
2147                     { \__graphics_backend_getbb_svg_auxv:Nw #2 }
2148                 }
2149             }
2150             ##2 \s__graphics_stop
2151         }
2152     }
2153     \use:x
2154     {
2155         \__graphics_backend_getbb_svg_auxii:w #3
2156         \tl_to_str:n {#1} = \tl_to_str:n {#1} =
2157         \s__graphics_stop
2158     }
2159 }
2160 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxii:w { }
2161 \cs_new_protected:Npn \__graphics_backend_getbb_svg_auxiii:Nw #1 ' #2 ' #3 \s__graphics_stop

```



```

2162 { \_graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2163 \cs_new_protected:Npn \_graphics_backend_getbb_svg_auxiv:Nw #1 " #2 " #3 \s__graphics_stop
2164 { \_graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2165 \cs_new_protected:Npn \_graphics_backend_getbb_svg_auxv:Nw #1 #2 ~ #3 \s__graphics_stop
2166 { \_graphics_backend_getbb_svg_auxvi:Nn #1 {#2} }
2167 \cs_new_protected:Npn \_graphics_backend_getbb_svg_auxvi:Nn #1#2
2168 {
2169   \tex_afterassignment:D \_graphics_backend_getbb_svg_auxvii:w
2170   \l__graphics_internal_dim #2 bp \scan_stop:
2171   \dim_set_eq:NN #1 \l__graphics_internal_dim
2172 }
2173 \cs_new_protected:Npn \_graphics_backend_getbb_svg_auxvii:w #1 \scan_stop: { }

```

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

`_graphics_backend_getbb_eps:n` Simply use the generic function.

```

\_graphics_backend_getbb_ps:n 2174 \_graphics_backend_loaded:n
2175 {
2176   \cs_new_eq:NN \_graphics_backend_getbb_eps:n \_graphics_read_bb:n
2177   \cs_new_eq:NN \_graphics_backend_getbb_ps:n \_graphics_read_bb:n
2178 }

```

(End 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 2179 \cs_new_protected:Npn \_graphics_backend_getbb_jpg:n #1
\_graphics_backend_getbb_jpeg:n 2180 {
2181   \int_zero:N \l__graphics_page_int
2182   \tl_clear:N \l__graphics_pagebox_tl
2183   \_graphics_extract_bb:n {#1}
2184 }
2185 \cs_new_eq:NN \_graphics_backend_getbb_jpeg:n \_graphics_backend_getbb_jpg:n
2186 \cs_new_eq:NN \_graphics_backend_getbb_png:n \_graphics_backend_getbb_jpg:n

```

(End 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

```

2187 \cs_new_protected:Npn \_graphics_backend_getbb_pdf:n #1
2188 {
2189   \tl_clear:N \l__graphics_decodearray_str
2190   \bool_set_false:N \l__graphics_interpolate_bool
2191   \_graphics_extract_bb:n {#1}
2192 }

```

(End definition for `_graphics_backend_getbb_pdf:n`.)

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

```

\_graphics_backend_include_ps:n 2193 \cs_new_protected:Npn \_graphics_backend_include_eps:n #1
\_graphics_backend_include_pdf:n 2194 { \_graphics_backend_include:nn { PSfile } {#1} }
\_graphics_backend_include:nn 2195 \cs_new_eq:NN \_graphics_backend_include_ps:n \_graphics_backend_include_eps:n
2196 \cs_new_protected:Npn \_graphics_backend_include_pdf:n #1
2197 { \_graphics_backend_include:nn { pdffile } {#1} }
2198 \cs_new_protected:Npn \_graphics_backend_include:nn #1#2

```

```

2199 {
2200   \__kernel_backend_literal:x
2201   {
2202     #1 = #2 \c_space_tl
2203     llx = \dim_to_decimal_in_bp:n \l__graphics_llx_dim \c_space_tl
2204     lly = \dim_to_decimal_in_bp:n \l__graphics_lly_dim \c_space_tl
2205     urx = \dim_to_decimal_in_bp:n \l__graphics_urx_dim \c_space_tl
2206     ury = \dim_to_decimal_in_bp:n \l__graphics_ury_dim
2207   }
2208 }

```

(End 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.

```

2209 \cs_new_protected:Npn \__graphics_backend_include_svg:n #1
2210 {
2211   \box_move_up:nn { \l__graphics_ury_dim }
2212   {
2213     \hbox:n
2214     {
2215       \__kernel_backend_literal:x
2216       {
2217         dvisvgm:img~
2218         \dim_to_decimal:n { \l__graphics_urx_dim } ~
2219         \dim_to_decimal:n { \l__graphics_ury_dim } ~
2220         \__graphics_backend_include_dequote:w #1 " #1 " \s__graphics_stop
2221       }
2222     }
2223   }
2224 }
2225 \cs_new_eq:NN \__graphics_backend_include_png:n \__graphics_backend_include_svg:n
2226 \cs_new_eq:NN \__graphics_backend_include_jpeg:n \__graphics_backend_include_svg:n
2227 \cs_new_eq:NN \__graphics_backend_include_jpg:n \__graphics_backend_include_svg:n
2228 \cs_new:Npn \__graphics_backend_include_dequote:w #1 " #2 " #3 \s__graphics_stop
2229   {#2}

```

(End definition for __graphics_backend_include_svg:n and others.)

```

\__graphics_backend_get_pagecount:n

```

```

2230 \__graphics_backend_loaded:n
2231 { \cs_new_eq:NN \__graphics_backend_get_pagecount:n \__graphics_get_pagecount:n }

```

(End definition for __graphics_backend_get_pagecount:n.)

```

2232 </dvisvgm>
2233 </package>

```

6 I3backend-pdf implementation

```
2234 ⟨*package⟩
2235 ⟨@@=pdf⟩
```

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

6.1 Shared code

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

```
2236 ⟨*!dvisvgm⟩

\l__pdf_internal_box
2237 \box_new:N \l__pdf_internal_box
(End definition for \l__pdf_internal_box.)
2238 ⟨!/dvisvgm⟩
```

6.2 dvips backend

```
2239 ⟨*dvips⟩

\_pdf_backend_pdfmark:n Used often enough it should be a separate function.
\_pdf_backend_pdfmark:x
2240 \cs_new_protected:Npn \_pdf_backend_pdfmark:n #1
2241 { \_kernel_backend_postscript:n { mark #1 ~ pdfmark } }
2242 \cs_generate_variant:Nn \_pdf_backend_pdfmark:n { x }
(End definition for \_pdf_backend_pdfmark:n.)
```

6.2.1 Catalogue entries

```
\_pdf_backend_catalog_gput:nn
\_pdf_backend_info_gput:nn
2243 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2244 { \_pdf_backend_pdfmark:n { { Catalog } << /#1 ~ #2 >> /PUT } }
2245 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2246 { \_pdf_backend_pdfmark:n { /#1 ~ #2 /DOCINFO } }
(End definition for \_pdf_backend_catalog_gput:nn and \_pdf_backend_info_gput:nn.)
```

6.2.2 Objects

```
\g__pdf_backend_object_int For tracking objects.
2247 \int_new:N \g__pdf_backend_object_int
(End definition for \g__pdf_backend_object_int.)
```

```

\__pdf_backend_object_new:n
\__pdf_backend_object_ref:n
2248 \cs_new_protected:Npn \__pdf_backend_object_new:n #1
2249 {
2250   \int_gincr:N \g__pdf_backend_object_int
2251   \int_const:cn
2252   { c__pdf_object_ \tl_to_str:n {#1} _int }
2253   { \g__pdf_backend_object_int }
2254 }
2255 \cs_new:Npn \__pdf_backend_object_ref:n #1
2256 { { pdf.obj \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } } }

```

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

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

```

\__pdf_backend_object_write:nnn
\__pdf_backend_object_write:nnx
\__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
2257 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2258 {
2259   \__pdf_backend_object_write_aux:nnn
2260   { \__pdf_backend_object_ref:n {#1} }
2261   {#2} {#3}
2262 }
2263 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nnx }
2264 \cs_new_protected:Npn \__pdf_backend_object_write_aux:nnn #1#2#3
2265 {
2266   \__pdf_backend_pdfmark:x
2267   {
2268     /objdef ~ #1
2269     /type
2270     \str_case:nn {#2}
2271     {
2272       { array } { /array }
2273       { dict } { /dict }
2274       { fstream } { /stream }
2275       { stream } { /stream }
2276     }
2277     /OBJ
2278   }
2279   \use:c { __pdf_backend_object_write_ #2 :nn } {#1} {#3}
2280 }
2281 \cs_new_protected:Npn \__pdf_backend_object_write_array:nn #1#2
2282 {
2283   \__pdf_backend_pdfmark:x
2284   { #1 ~0~ [ ~ \exp_not:n {#2} ~ ] ~ /PUTINTERVAL }
2285 }
2286 \cs_new_protected:Npn \__pdf_backend_object_write_dict:nn #1#2
2287 {
2288   \__pdf_backend_pdfmark:x
2289   { #1 << \exp_not:n {#2} >> /PUT }
2290 }
2291 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nn #1#2
2292 {
2293   \exp_args:Nx
2294   \__pdf_backend_object_write_fstream:nnn {#1} #2
2295 }

```

```

2296 \cs_new_protected:Npn \__pdf_backend_object_write_fstream:nnn #1#2#3
2297 {
2298   \__kernel_backend_postscript:n
2299   {
2300     SDict ~ begin ~
2301     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark ~
2302     mark ~ #1 ~ ( #3 )~ ( r )~ file ~ /PUT ~ pdfmark ~
2303     end
2304   }
2305 }
2306 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nn #1#2
2307 {
2308   \exp_args:Nx
2309   \__pdf_backend_object_write_stream:nnn {#1} #2
2310 }
2311 \cs_new_protected:Npn \__pdf_backend_object_write_stream:nnn #1#2#3
2312 {
2313   \__kernel_backend_postscript:n
2314   {
2315     mark ~ #1 ~ ( #3 ) /PUT ~ pdfmark ~
2316     mark ~ #1 ~ << #2 >> /PUT ~ pdfmark
2317   }
2318 }

```

(End 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:nx
2319 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2320 {
2321   \int_gincr:N \g__pdf_backend_object_int
2322   \__pdf_backend_object_write_aux:nnn
2323   { { pdf.obj \int_use:N \g__pdf_backend_object_int } }
2324   {#1} {#2}
2325 }
2326 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last: Much like the annotation version.

```

2327 \cs_new:Npn \__pdf_backend_object_last:
2328 { { pdf.obj \int_use:N \g__pdf_backend_object_int } }

```

(End definition for __pdf_backend_object_last:.)

__pdf_backend_pageobject_ref:n Page references are easy in dvips.

```

2329 \cs_new:Npn \__pdf_backend_pageobject_ref:n #1
2330 { { Page #1 } }

```

(End definition for __pdf_backend_pageobject_ref:n.)

6.2.3 Annotations

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

```

\l__pdf_backend_content_box The content of an annotation.
2331 \box_new:N \l__pdf_backend_content_box
(End definition for \l__pdf_backend_content_box.)

\l__pdf_backend_model_box For creating model sizing for links.
2332 \box_new:N \l__pdf_backend_model_box
(End definition for \l__pdf_backend_model_box.)

\g__pdf_backend_annotation_int Needed as objects which are not annotations could be created.
2333 \int_new:N \g__pdf_backend_annotation_int
(End definition for \g__pdf_backend_annotation_int.)

\_pdf_backend_annotation:nmmn Annotations are objects, but we track them separately. Notably, they are not in the
object data lists. Here, to get the co-ordinates of the annotation, we need to have the
data collected at the PostScript level. That requires a bit of box trickery (effectively a
LATEX 2ε picture of zero size). Once the data is collected, use it to set up the annotation
border.
2334 \cs_new_protected:Npn \_pdf_backend_annotation:nmmn #1#2#3#4
2335 {
2336   \exp_args:Nf \_pdf_backend_annotation_aux:nmmn
2337     { \dim_eval:n {#1} } {#2} {#3} {#4}
2338 }
2339 \cs_new_protected:Npn \_pdf_backend_annotation_aux:nmmn #1#2#3#4
2340 {
2341   \box_move_down:nn {#3}
2342   { \hbox:n { \_kernel_backend_postscript:n { pdf.save.ll } } }
2343   \box_move_up:nn {#2}
2344   {
2345     \hbox:n
2346       {
2347         \_kernel_kern:n {#1}
2348         \_kernel_backend_postscript:n { pdf.save.ur }
2349         \_kernel_kern:n { -#1 }
2350       }
2351   }
2352   \int_gincr:N \g__pdf_backend_object_int
2353   \int_gset_eq:NN \g__pdf_backend_annotation_int \g__pdf_backend_object_int
2354   \_pdf_backend_pdfmark:x
2355   {
2356     /_objdef { pdf.obj \int_use:N \g__pdf_backend_object_int }
2357     pdf.rect
2358     #4 ~
2359     /ANN
2360   }
2361 }
(End definition for \_pdf_backend_annotation:nmmn.)

```

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

```

2362 \cs_new:Npn \_pdf_backend_annotation_last:
2363   { { pdf.obj \int_use:N \g__pdf_backend_annotation_int } }

```

(End definition for `_pdf_backend_annotation_last:.`)

`\g__pdf_backend_link_int` To track annotations which are links.

```

2364 \int_new:N \g__pdf_backend_link_int

```

(End definition for `\g__pdf_backend_link_int.`)

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

```

2365 \tl_new:N \g__pdf_backend_link_dict_tl

```

(End definition for `\g__pdf_backend_link_dict_tl.`)

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

```

2366 \int_new:N \g__pdf_backend_link_sf_int

```

(End definition for `\g__pdf_backend_link_sf_int.`)

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

```

2367 \bool_new:N \g__pdf_backend_link_math_bool

```

(End definition for `\g__pdf_backend_link_math_bool.`)

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

```

2368 \bool_new:N \g__pdf_backend_link_bool

```

(End definition for `\g__pdf_backend_link_bool.`)

`\l__pdf_breaklink_pdfmark_tl` Swappable content for link breaking.

```

2369 \tl_new:N \l__pdf_breaklink_pdfmark_tl
2370 \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdfmark }

```

(End definition for `\l__pdf_breaklink_pdfmark_tl.`)

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

```

2371 \cs_new_protected:Npn \_pdf_breaklink_postscript:n #1 { }

```

(End definition for `_pdf_breaklink_postscript:n.`)

`__pdf_breaklink_usebox:N` Swappable box unpacking or use.

```

2372 \cs_new_eq:NN \__pdf_breaklink_usebox:N \box_use:N

```

(End definition for `__pdf_breaklink_usebox:N.`)

```

\_pdf_backend_link_begin_goto:nnw
\_pdf_backend_link_begin_user:nnw
\_pdf_backend_link:nw
\_pdf_backend_link_aux:nw
\_pdf_backend_link_end:
\_pdf_backend_link_end_aux:
\_pdf_backend_link_minima:
  \_pdf_backend_link_outerbox:n
\_pdf_backend_link_sf_save:
  \_pdf_backend_link_sf_restore:
pdf.linkdp.pad
pdf.linkht.pad
pdf.llx
pdf.lly
pdf.ury
pdf.link.dict
pdf.outerbox
pdf.baselineskip

```

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

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

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

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

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

```

2373 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2374 {
2375   \_pdf_backend_link_begin:nw
2376   { #1 /Subtype /Link /Action << /S /GoTo /D ( #2 ) >> }
2377 }
2378 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
2379 { \_pdf_backend_link_begin:nw {#1#2} }
2380 \cs_new_protected:Npn \_pdf_backend_link_begin:nw #1
2381 {
2382   \bool_if:NF \g__pdf_backend_link_bool
2383   { \_pdf_backend_link_begin_aux:nw {#1} }
2384 }

```

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

```

2385 \cs_new_protected:Npn \_pdf_backend_link_begin_aux:nw #1
2386 {
2387   \bool_gset_true:N \g__pdf_backend_link_bool
2388   \_kernel_backend_postscript:n
2389   { /pdf.link.dict ( #1 ) def }
2390   \tl_gset:Nn \g__pdf_backend_link_dict_tl {#1}
2391   \_pdf_backend_link_sf_save:
2392   \mode_if_math:TF
2393   { \bool_gset_true:N \g__pdf_backend_link_math_bool }
2394   { \bool_gset_false:N \g__pdf_backend_link_math_bool }
2395   \hbox_set:Nw \l__pdf_backend_content_box
2396   \_pdf_backend_link_sf_restore:
2397   \bool_if:NT \g__pdf_backend_link_math_bool
2398   { \c_math_toggle_token }
2399 }
2400 \cs_new_protected:Npn \_pdf_backend_link_end:
2401 {
2402   \bool_if:NT \g__pdf_backend_link_bool
2403   { \_pdf_backend_link_end_aux: }
2404 }
2405 \cs_new_protected:Npn \_pdf_backend_link_end_aux:

```



```

2406 {
2407   \bool_if:NT \g__pdf_backend_link_math_bool
2408     { \c_math_toggle_token }
2409   \__pdf_backend_link_sf_save:
2410   \hbox_set_end:
2411   \__pdf_backend_link_minima:
2412   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2413   \exp_args:Nx \__pdf_backend_link_outerbox:n
2414     {
2415       \int_if_odd:nTF { \value { page } }
2416         { \oddsidemargin }
2417         { \evensidemargin }
2418     }
2419   \box_move_down:nn { \box_dp:N \l__pdf_backend_content_box }
2420     { \hbox:n { \__kernel_backend_postscript:n { pdf.save.linkll } } }
2421   \__pdf_breaklink_postscript:n { pdf.bordertracking.begin }
2422   \__pdf_breaklink_usebox:N \l__pdf_backend_content_box
2423   \__pdf_breaklink_postscript:n { pdf.bordertracking.end }
2424   \box_move_up:nn { \box_ht:N \l__pdf_backend_content_box }
2425     {
2426       \hbox:n
2427         { \__kernel_backend_postscript:n { pdf.save.linkur } }
2428     }
2429   \int_gincr:N \g__pdf_backend_object_int
2430   \int_gset_eq:NN \g__pdf_backend_link_int \g__pdf_backend_object_int
2431   \__kernel_backend_postscript:x
2432     {
2433       mark
2434       /_objdef { pdf.obj \int_use:N \g__pdf_backend_link_int }
2435       \g__pdf_backend_link_dict_tl \c_space_tl
2436       pdf.rect
2437       /ANN ~ \l__pdf_breaklink_pdfmark_tl
2438     }
2439   \__pdf_backend_link_sf_restore:
2440   \bool_gset_false:N \g__pdf_backend_link_bool
2441 }
2442 \cs_new_protected:Npn \__pdf_backend_link_minima:
2443 {
2444   \hbox_set:Nn \l__pdf_backend_model_box { Gg }
2445   \__kernel_backend_postscript:x
2446     {
2447     /pdf.linkdp.pad ~
2448     \dim_to_decimal:n
2449     {
2450       \dim_max:nn
2451       {
2452         \box_dp:N \l__pdf_backend_model_box
2453         - \box_dp:N \l__pdf_backend_content_box
2454       }
2455       { Opt }
2456     } ~
2457     pdf.pt.dvi ~ def
2458     /pdf.linkht.pad ~
2459     \dim_to_decimal:n

```

```

2460         {
2461             \dim_max:nn
2462             {
2463                 \box_ht:N \l__pdf_backend_model_box
2464                 - \box_ht:N \l__pdf_backend_content_box
2465             }
2466             { Opt }
2467         } ~
2468         pdf.pt.dvi ~ def
2469     }
2470 }
2471 \cs_new_protected:Npn \__pdf_backend_link_outerbox:n #1
2472 {
2473     \__kernel_backend_postscript:x
2474     {
2475         /pdf.outerbox
2476         [
2477             \dim_to_decimal:n {#1} ~
2478             \dim_to_decimal:n { -\box_dp:N \l__pdf_backend_model_box } ~
2479             \dim_to_decimal:n { #1 + \textwidth } ~
2480             \dim_to_decimal:n { \box_ht:N \l__pdf_backend_model_box }
2481         ]
2482         [ exch { pdf.pt.dvi } forall ] def
2483         /pdf.baselineskip ~
2484         \dim_to_decimal:n { \tex_baselineskip:D } ~ dup ~ 0 ~ gt
2485         { pdf.pt.dvi ~ def }
2486         { pop ~ pop }
2487         ifelse
2488     }
2489 }
2490 \cs_new_protected:Npn \__pdf_backend_link_sf_save:
2491 {
2492     \int_gset:Nn \g__pdf_backend_link_sf_int
2493     {
2494         \mode_if_horizontal:TF
2495         { \tex_spacefactor:D }
2496         { 0 }
2497     }
2498 }
2499 \cs_new_protected:Npn \__pdf_backend_link_sf_restore:
2500 {
2501     \mode_if_horizontal:T
2502     {
2503         \int_compare:nNnT \g__pdf_backend_link_sf_int > { 0 }
2504         { \int_set_eq:NN \tex_spacefactor:D \g__pdf_backend_link_sf_int }
2505     }
2506 }

```

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

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

```

2507 \use_none:n
2508 {
2509   \cs_if_exist:NT \@makecol@hook
2510   {
2511     \tl_put_right:Nn \@makecol@hook
2512     {
2513       \box_if_empty:NF \@cclv
2514       {
2515         \vbox_set:Nn \@cclv
2516         {
2517           \__kernel_backend_postscript:n
2518           {
2519             pdf.globaldict /pdf.brokenlink.rect ~ known
2520             { pdf.bordertracking.continue }
2521             if
2522             }
2523             \vbox_unpack_drop:N \@cclv
2524             \__kernel_backend_postscript:n
2525             { pdf.bordertracking.endpage }
2526           }
2527         }
2528       }
2529       \tl_set:Nn \l__pdf_breaklink_pdfmark_tl { pdf.pdfmark }
2530       \cs_set_eq:NN \__pdf_breaklink_postscript:n \__kernel_backend_postscript:n
2531       \cs_set_eq:NN \__pdf_breaklink_usebox:N \hbox_unpack:N
2532     }
2533   }

```

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

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

```

2534 \cs_new:Npn \__pdf_backend_link_last:
2535 { { pdf.obj \int_use:N \g__pdf_backend_link_int } }

```

(End definition for __pdf_backend_link_last:.)

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

```

2536 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1
2537 {
2538   \__kernel_backend_postscript:x
2539   {
2540     /pdf.linkmargin { \dim_to_decimal:n {#1} ~ pdf.pt.dvi } def
2541   }
2542 }

```

(End definition for __pdf_backend_link_margin:n.)

__pdf_backend_destination:nn Here, we need to turn the zoom into a scale. We also need to know where the current 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.

```

2543 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2
2544 {
2545   \__kernel_backend_postscript:n { pdf.dest.anchor }

```

```

2546 \__pdf_backend_pdfmark:x
2547 {
2548 /View
2549 [
2550 \str_case:nnF {#2}
2551 {
2552 { xyz } { /XYZ ~ pdf.dest.point ~ null }
2553 { fit } { /Fit }
2554 { fitb } { /FitB }
2555 { fitbh } { /FitBH ~ pdf.dest.y }
2556 { fitbv } { /FitBV ~ pdf.dest.x }
2557 { fith } { /FitH ~ pdf.dest.y }
2558 { fitv } { /FitV ~ pdf.dest.x }
2559 { fitr } { /Fit }
2560 }
2561 {
2562 /XYZ ~ pdf.dest.point ~ \fp_eval:n { (#2) / 100 }
2563 }
2564 ]
2565 /Dest ( \exp_not:n {#1} ) cvn
2566 /DEST
2567 }
2568 }
2569 \cs_new_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
2570 {
2571 \exp_args:Ne \__pdf_backend_destination_aux:nnnn
2572 { \dim_eval:n {#2} } {#1} {#3} {#4}
2573 }
2574 \cs_new_protected:Npn \__pdf_backend_destination_aux:nnnn #1#2#3#4
2575 {
2576 \vbox_to_zero:n
2577 {
2578 \__kernel_kern:n {#4}
2579 \hbox:n { \__kernel_backend_postscript:n { pdf.save.ll } }
2580 \tex_vss:D
2581 }
2582 \__kernel_kern:n {#1}
2583 \vbox_to_zero:n
2584 {
2585 \__kernel_kern:n { -#3 }
2586 \hbox:n { \__kernel_backend_postscript:n { pdf.save.ur } }
2587 \tex_vss:D
2588 }
2589 \__kernel_kern:n { -#1 }
2590 \__pdf_backend_pdfmark:n
2591 {
2592 /View
2593 [
2594 /FitR ~
2595 pdf.llx ~ pdf.lly ~ pdf.dest2device ~
2596 pdf.urx ~ pdf.ury ~ pdf.dest2device
2597 ]
2598 /Dest ( #2 ) cvn
2599 /DEST

```

```

2600     }
2601 }

```

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

6.2.4 Structure

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

```

\_pdf_backend_compress_objects:n
2602 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2603 {
2604   \int_compare:nNnT {#1} = 0
2605   {
2606     \__kernel_backend_literal_postscript:n
2607     {
2608       /setdistillerparams ~ where
2609       { pop << /CompressPages ~ false >> setdistillerparams }
2610       if
2611     }
2612   }
2613 }
2614 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2615 {
2616   \bool_if:nF {#1}
2617   {
2618     \__kernel_backend_literal_postscript:n
2619     {
2620       /setdistillerparams ~ where
2621       { pop << /CompressStreams ~ false >> setdistillerparams }
2622       if
2623     }
2624   }
2625 }

```

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

`_pdf_backend_version_major_gset:n`

`_pdf_backend_version_minor_gset:n`

```

2626 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1
2627 {
2628   \cs_gset:Npx \_pdf_backend_version_major: { \int_eval:n {#1} }
2629 }
2630 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2631 {
2632   \cs_gset:Npx \_pdf_backend_version_minor: { \int_eval:n {#1} }
2633 }

```

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

`_pdf_backend_version_major:` Data not available!

`_pdf_backend_version_minor:`

```

2634 \cs_new:Npn \_pdf_backend_version_major: { -1 }
2635 \cs_new:Npn \_pdf_backend_version_minor: { -1 }

```

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

6.2.5 Marked content

`_pdf_backend_bdc:nn` Simple wrappers.
`_pdf_backend_emc:`

```
2636 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2637   { \_pdf_backend_pdfmark:n { /#1 ~ #2 /BDC } }
2638 \cs_new_protected:Npn \_pdf_backend_emc:
2639   { \_pdf_backend_pdfmark:n { /EMC } }

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

2640 </dvips>
```

6.3 LuaTeX and pdfTeX backend

```
2641 <*luatex | pdftex>
```

6.3.1 Annotations

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

```
2642 \cs_new_protected:Npn \_pdf_backend_annotation:nnnn #1#2#3#4
2643   {
2644     <*luatex>
2645     \tex_pdfextension:D annot ~
2646     </luatex>
2647     <*pdftex>
2648     \tex_pdfannot:D
2649     </pdftex>
2650     width ~ \dim_eval:n {#1} ~
2651     height ~ \dim_eval:n {#2} ~
2652     depth ~ \dim_eval:n {#3} ~
2653     {#4}
2654   }
```

(End definition for `_pdf_backend_annotation:nnnn`.)

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

```
2655 \cs_new:Npx \_pdf_backend_annotation_last:
2656   {
2657     \exp_not:N \int_value:w
2658     <*luatex>
2659     \exp_not:N \tex_pdffeedback:D lastannot ~
2660     </luatex>
2661     <*pdftex>
2662     \exp_not:N \tex_pdflastannot:D
2663     </pdftex>
2664     \c_space_tl 0 ~ R
2665   }
```

(End definition for `_pdf_backend_annotation_last:`.)

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

```
\_pdf_backend_link_begin_user:nnw
\_pdf_backend_link_begin:nnnw
\_pdf_backend_link_end:
2666 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nnw #1#2
2667   { \_pdf_backend_link_begin:nnw {#1} { goto~name } {#2} }
2668 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nnw #1#2
```

```

2669 { \_pdf_backend_link_begin:nnw {#1} { user } {#2} }
2670 \cs_new_protected:Npn \_pdf_backend_link_begin:nnw #1#2#3
2671 {
2672 <*luatex>
2673   \tex_pdfextension:D startlink ~
2674 </luatex>
2675 <*pdftex>
2676   \tex_pdfstartlink:D
2677 </pdftex>
2678   attr {#1}
2679   #2 {#3}
2680 }
2681 \cs_new_protected:Npn \_pdf_backend_link_end:
2682 {
2683 <*luatex>
2684   \tex_pdfextension:D endlink \scan_stop:
2685 </luatex>
2686 <*pdftex>
2687   \tex_pdfendlink:D
2688 </pdftex>
2689 }

```

(End definition for _pdf_backend_link_begin_goto:nnw and others.)

_pdf_backend_link_last: Formatted for direct use.

```

2690 \cs_new:Npx \_pdf_backend_link_last:
2691 {
2692   \exp_not:N \int_value:w
2693 <*luatex>
2694   \exp_not:N \tex_pdffeedback:D lastlink ~
2695 </luatex>
2696 <*pdftex>
2697   \exp_not:N \tex_pdflastlink:D
2698 </pdftex>
2699   \c_space_tl 0 ~ R
2700 }

```

(End definition for _pdf_backend_link_last:.)

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

```

2701 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
2702 {
2703 <*luatex>
2704   \tex_pdfvariable:D linkmargin
2705 </luatex>
2706 <*pdftex>
2707   \tex_pdflinkmargin:D
2708 </pdftex>
2709   \dim_eval:n {#1} \scan_stop:
2710 }

```

(End definition for _pdf_backend_link_margin:n.)

`_pdf_backend_destination:nn`
`_pdf_backend_destination:nnnn`

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

```

2711 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
2712 {
2713 <*luatex>
2714   \tex_pdfextension:D dest ~
2715 </luatex>
2716 <*pdftex>
2717   \tex_pdfdest:D
2718 </pdftex>
2719   name {#1}
2720   \str_case:nnF {#2}
2721   {
2722     { xyz } { xyz }
2723     { fit } { fit }
2724     { fitb } { fitb }
2725     { fitbh } { fitbh }
2726     { fitbv } { fitbv }
2727     { fith } { fith }
2728     { fitv } { fitv }
2729     { fitr } { fitr }
2730   }
2731   { xyz ~ zoom \fp_eval:n { #2 * 10 } }
2732   \scan_stop:
2733 }
2734 \cs_new_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
2735 {
2736 <*luatex>
2737   \tex_pdfextension:D dest ~
2738 </luatex>
2739 <*pdftex>
2740   \tex_pdfdest:D
2741 </pdftex>
2742   name {#1}
2743   fitr ~
2744   width \dim_eval:n {#2} ~
2745   height \dim_eval:n {#3} ~
2746   depth \dim_eval:n {#4} \scan_stop:
2747 }

```

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

6.3.2 Catalogue entries

`_pdf_backend_catalog_gput:nn`
`_pdf_backend_info_gput:nn`

```

2748 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2749 {
2750 <*luatex>
2751   \tex_pdfextension:D catalog
2752 </luatex>
2753 <*pdftex>
2754   \tex_pdfcatalog:D
2755 </pdftex>

```



```

2756     { / #1 ~ #2 }
2757   }
2758   \cs_new_protected:Npn \__pdf_backend_info_gput:nn #1#2
2759     {
2760     \*luatex
2761       \tex_pdfextension:D info
2762     \</luatex
2763     \*pdftex
2764       \tex_pdfinfo:D
2765     \</pdftex
2766     { / #1 ~ #2 }
2767   }

```

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

6.3.3 Objects

\g__pdf_backend_object_prop For tracking objects to allow finalisation.

```

2768 \prop_new:N \g__pdf_backend_object_prop

```

(End definition for \g__pdf_backend_object_prop.)

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

__pdf_backend_object_ref:n

```

2769 \cs_new_protected:Npn \__pdf_backend_object_new:n #1
2770   {
2771   \*luatex
2772     \tex_pdfextension:D obj ~
2773   \</luatex
2774   \*pdftex
2775     \tex_pdfobj:D
2776   \</pdftex
2777     reserveobjnum ~
2778     \int_const:cn
2779     { c__pdf_object_ \tl_to_str:n {#1} _int }
2780   \*luatex
2781     { \tex_pdffeedback:D lastobj }
2782   \</luatex
2783   \*pdftex
2784     { \tex_pdflastobj:D }
2785   \</pdftex
2786   }

```

```

2787 \cs_new:Npn \__pdf_backend_object_ref:n #1
2788   { \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

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

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

__pdf_backend_object_write:nnx

__pdf_backend_object_write:nn

__pdf_exp_not_i:nn

__pdf_exp_not_ii:nn

```

2789 \cs_new_protected:Npn \__pdf_backend_object_write:nnn #1#2#3
2790   {
2791   \*luatex
2792     \tex_immediate:D \tex_pdfextension:D obj ~
2793   \</luatex
2794   \*pdftex
2795     \tex_immediate:D \tex_pdfobj:D
2796   \</pdftex

```

```

2797     useobjnum ~
2798     \int_use:c
2799     { c__pdf_object_ \tl_to_str:n {#1} _int }
2800     \__pdf_backend_object_write:nn {#2} {#3}
2801   }
2802 \cs_new:Npn \__pdf_backend_object_write:nn #1#2
2803 {
2804   \str_case:nn {#1}
2805   {
2806     { array } { { [ ~ \exp_not:n {#2} ~ ] } }
2807     { dict } { { << ~ \exp_not:n {#2} ~ >> } }
2808     { fstream }
2809     {
2810       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2811       file ~ { \__pdf_exp_not_ii:nn #2 }
2812     }
2813     { stream }
2814     {
2815       stream ~ attr ~ { \__pdf_exp_not_i:nn #2 } ~
2816       { \__pdf_exp_not_ii:nn #2 }
2817     }
2818   }
2819 }
2820 \cs_generate_variant:Nn \__pdf_backend_object_write:nnn { nnx }
2821 \cs_new:Npn \__pdf_exp_not_i:nn #1#2 { \exp_not:n {#1} }
2822 \cs_new:Npn \__pdf_exp_not_ii:nn #1#2 { \exp_not:n {#2} }

```

(End definition for __pdf_backend_object_write:nnn and others.)

__pdf_backend_object_now:nn Much like writing, but direct creation.

```

\__pdf_backend_object_now:nx
2823 \cs_new_protected:Npn \__pdf_backend_object_now:nn #1#2
2824 {
2825   <*luatex>
2826   \tex_immediate:D \tex_pdfextension:D obj ~
2827   </luatex>
2828   <*pdftex>
2829   \tex_immediate:D \tex_pdfobj:D
2830   </pdftex>
2831   \__pdf_backend_object_write:nn {#1} {#2}
2832 }
2833 \cs_generate_variant:Nn \__pdf_backend_object_now:nn { nx }

```

(End definition for __pdf_backend_object_now:nn.)

__pdf_backend_object_last: Much like annotation.

```

2834 \cs_new:Npx \__pdf_backend_object_last:
2835 {
2836   \exp_not:N \int_value:w
2837   <*luatex>
2838   \exp_not:N \tex_pdffeedback:D lastobj ~
2839   </luatex>
2840   <*pdftex>
2841   \exp_not:N \tex_pdflastobj:D
2842   </pdftex>
2843   \c_space_tl 0 ~ R

```

```
2844 }
(End definition for \_pdf_backend_object_last:.)
```

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

```
2845 \cs_new:Npx \_pdf_backend_pageobject_ref:n #1
2846 {
2847   \exp_not:N \int_value:w
2848   \*luatex
2849   \exp_not:N \tex_pdffeedback:D pageref
2850   \*luatex
2851   \*pdftex
2852   \exp_not:N \tex_pdfpageref:D
2853   \*pdftex
2854   \c_space_tl #1 \c_space_tl \c_space_tl \c_space_tl 0 ~ R
2855 }
(End definition for \_pdf_backend_pageobject_ref:n.)
```

6.3.4 Structure

_pdf_backend_compresslevel:n Simply pass data to the engine.

```
\_pdf_backend_compress_objects:n
\_pdf_backend_objcompresslevel:n
2856 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1
2857 {
2858   \tex_global:D
2859   \*luatex
2860   \tex_pdfvariable:D compresslevel
2861   \*luatex
2862   \*pdftex
2863   \tex_pdfcompresslevel:D
2864   \*pdftex
2865   \int_value:w \int_eval:n {#1} \scan_stop:
2866 }
2867 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1
2868 {
2869   \bool_if:nTF {#1}
2870     { \_pdf_backend_objcompresslevel:n { 2 } }
2871     { \_pdf_backend_objcompresslevel:n { 0 } }
2872 }
2873 \cs_new_protected:Npn \_pdf_backend_objcompresslevel:n #1
2874 {
2875   \tex_global:D
2876   \*luatex
2877   \tex_pdfvariable:D objcompresslevel
2878   \*luatex
2879   \*pdftex
2880   \tex_pdfobjcompresslevel:D
2881   \*pdftex
2882   #1 \scan_stop:
2883 }
(End definition for \_pdf_backend_compresslevel:n, \_pdf_backend_compress_objects:n, and \_pdf_backend_objcompresslevel:n.)
```

`_pdf_backend_version_major_gset:n`
`_pdf_backend_version_minor_gset:n`

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

```
2884 \cs_new_protected:Npx \_pdf_backend_version_major_gset:n #1
2885 {
2886 <*luatex>
2887   \int_compare:nNnT \tex luatexversion:D > { 106 }
2888   {
2889     \exp_not:N \tex_global:D \tex_pdfvariable:D majorversion
2890     \exp_not:N \int_eval:n {#1} \scan_stop:
2891   }
2892 </luatex>
2893 <*pdftex>
2894   \cs_if_exist:NT \tex_pdfmajorversion:D
2895   {
2896     \exp_not:N \tex_global:D \tex_pdfmajorversion:D
2897     \exp_not:N \int_eval:n {#1} \scan_stop:
2898   }
2899 </pdftex>
2900 }
2901 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1
2902 {
2903   \tex_global:D
2904 <*luatex>
2905   \tex_pdfvariable:D minorversion
2906 </luatex>
2907 <*pdftex>
2908   \tex_pdfminorversion:D
2909 </pdftex>
2910   \int_eval:n {#1} \scan_stop:
2911 }
```

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

`_pdf_backend_version_major:`
`_pdf_backend_version_minor:`

As above.

```
2912 \cs_new:Npx \_pdf_backend_version_major:
2913 {
2914 <*luatex>
2915   \int_compare:nNnTF \tex luatexversion:D > { 106 }
2916   { \exp_not:N \tex_the:D \tex_pdfvariable:D majorversion }
2917   { 1 }
2918 </luatex>
2919 <*pdftex>
2920   \cs_if_exist:NTF \tex_pdfmajorversion:D
2921   { \exp_not:N \tex_the:D \tex_pdfmajorversion:D }
2922   { 1 }
2923 </pdftex>
2924 }
2925 \cs_new:Npn \_pdf_backend_version_minor:
2926 {
2927   \tex_the:D
2928 <*luatex>
2929   \tex_pdfvariable:D minorversion
2930 </luatex>
2931 <*pdftex>
2932   \tex_pdfminorversion:D
```

```

2933 </pdftex>
2934 }

```

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

6.3.5 Marked content

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

```

2935 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2
2936 { \_kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
2937 \cs_new_protected:Npn \_pdf_backend_emc:
2938 { \_kernel_backend_literal_page:n { EMC } }

```

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

```

2939 </luatex | pdftex>

```

6.4 dviPDF backend

```

2940 <*dviPDF | xetex>

```

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

```

\_pdf_backend:n
\_pdf_backend:x
2941 \cs_new_protected:Npx \_pdf_backend:n #1
2942 { \_kernel_backend_literal:n { pdf: #1 } }
2943 \cs_generate_variant:Nn \_pdf_backend:n { x }

```

(End definition for `_pdf_backend:n`.)

6.4.1 Catalogue entries

```

\_pdf_backend_catalog_gput:nn
\_pdf_backend_info_gput:nn
2944 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2
2945 { \_pdf_backend:n { put ~ @catalog << /#1 ~ #2 >> } }
2946 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2
2947 { \_pdf_backend:n { docinfo << /#1 ~ #2 >> } }

```

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

6.4.2 Objects

`\g__pdf_backend_object_int` For tracking objects to allow finalisation.

```

\g__pdf_backend_object_int
\g__pdf_backend_object_prop
2948 \int_new:N \g__pdf_backend_object_int
2949 \prop_new:N \g__pdf_backend_object_prop

```

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

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

```

\_pdf_backend_object_ref:n
2950 \cs_new_protected:Npn \_pdf_backend_object_new:n #1
2951 {
2952   \int_gincr:N \g__pdf_backend_object_int
2953   \int_const:cn
2954   { c__pdf_object_ \tl_to_str:n {#1} _int }
2955   { \g__pdf_backend_object_int }
2956 }
2957 \cs_new:Npn \_pdf_backend_object_ref:n #1
2958 { @pdf.obj \int_use:c { c__pdf_object_ \tl_to_str:n {#1} _int } }

```

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

```
\_pdf_backend_object_write:nnn
\_pdf_backend_object_write:nnx
\_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.
2959 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3
2960 {
2961   \use:c { \_pdf_backend_object_write_ #2 :nn }
2962   { \_pdf_backend_object_ref:n {#1} } {#3}
2963 }
2964 \cs_generate_variant:Nn \_pdf_backend_object_write:nnn { nnx }
2965 \cs_new_protected:Npn \_pdf_backend_object_write_array:nn #1#2
2966 {
2967   \_pdf_backend:x
2968   { obj ~ #1 ~ [ ~ \exp_not:n {#2} ~ ] }
2969 }
2970 \cs_new_protected:Npn \_pdf_backend_object_write_dict:nn #1#2
2971 {
2972   \_pdf_backend:x
2973   { obj ~ #1 ~ << ~ \exp_not:n {#2} ~ >> }
2974 }
2975 \cs_new_protected:Npn \_pdf_backend_object_write_fstream:nn #1#2
2976 { \_pdf_backend_object_write_stream:nnnn { f } {#1} #2 }
2977 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nn #1#2
2978 { \_pdf_backend_object_write_stream:nnnn { } {#1} #2 }
2979 \cs_new_protected:Npn \_pdf_backend_object_write_stream:nnnn #1#2#3#4
2980 {
2981   \_pdf_backend:x
2982   {
2983     #1 stream ~ #2 ~
2984     ( \exp_not:n {#4} ) ~ << \exp_not:n {#3} >>
2985   }
2986 }
```

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

```
\_pdf_backend_object_now:nn
\_pdf_backend_object_now:nx

No anonymous objects with dvipdfmx so we have to give an object name.
2987 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2
2988 {
2989   \int_gincr:N \g__pdf_backend_object_int
2990   \exp_args:Nnx \use:c { \_pdf_backend_object_write_ #1 :nn }
2991   { @pdf.obj \int_use:N \g__pdf_backend_object_int }
2992   {#2}
2993 }
2994 \cs_generate_variant:Nn \_pdf_backend_object_now:nn { nx }
```

(End definition for `_pdf_backend_object_now:nn`.)

`_pdf_backend_object_last:`

```
2995 \cs_new:Npn \_pdf_backend_object_last:
2996 { @pdf.obj \int_use:N \g__pdf_backend_object_int }
```

(End definition for `_pdf_backend_object_last:.`)

`_pdf_backend_pageobject_ref:n`

Page references are easy in dvipdfmx/X_qTeX.

```
2997 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1
2998 { @page #1 }
```

(End definition for `_pdf_backend_pageobject_ref:n`.)

6.4.3 Annotations

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

```
2999 \int_new:N \g_pdf_backend_annotation_int
```

(End definition for `\g_pdf_backend_annotation_int`.)

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

```
3000 \cs_new_protected:Npn \_pdf_backend_annotation:nmmn #1#2#3#4
3001 {
3002   \int_gincr:N \g_pdf_backend_object_int
3003   \int_gset_eq:NN \g_pdf_backend_annotation_int \g_pdf_backend_object_int
3004   \_pdf_backend:x
3005   {
3006     ann ~ @pdf.obj \int_use:N \g_pdf_backend_object_int \c_space_tl
3007     width ~ \dim_eval:n {#1} ~
3008     height ~ \dim_eval:n {#2} ~
3009     depth ~ \dim_eval:n {#3} ~
3010     << /Type /Annot #4 >>
3011   }
3012 }
```

(End definition for `_pdf_backend_annotation:nmmn`.)

`_pdf_backend_annotation_last:`

```
3013 \cs_new:Npn \_pdf_backend_annotation_last:
3014 { @pdf.obj \int_use:N \g_pdf_backend_annotation_int }
```

(End definition for `_pdf_backend_annotation_last:`.)

`\g_pdf_backend_link_int` To track annotations which are links.

```
3015 \int_new:N \g_pdf_backend_link_int
```

(End definition for `\g_pdf_backend_link_int`.)

`_pdf_backend_link_begin_goto:nmw` All created using the same internals.

`_pdf_backend_link_begin_user:nmw`

```
3016 \cs_new_protected:Npn \_pdf_backend_link_begin_goto:nmw #1#2
```

```
3017 { \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) >> } }
```

`_pdf_backend_link_begin:n`

```
3018 \cs_new_protected:Npn \_pdf_backend_link_begin_user:nmw #1#2
```

```
3019 { \_pdf_backend_link_begin:n { #1#2 } }
```

`_pdf_backend_link_end:`

```
3020 \cs_new_protected:Npx \_pdf_backend_link_begin:n #1
```

```
3021 {
3022   \exp_not:N \int_gincr:N \exp_not:N \g_pdf_backend_link_int
3023   \_pdf_backend:x
```

```
3024   {
```

```
3025     bann ~
```

```
3026     @pdf.lnk
```

```
3027     \exp_not:N \int_use:N \exp_not:N \g_pdf_backend_link_int
```

```
3028     \c_space_tl
```

```
3029     <<
```

```
3030     /Type /Annot
```

```
3031     #1
```

```
3032     >>
```

```
3033   }
```

```
3034 }
```

```
3035 \cs_new_protected:Npn \_pdf_backend_link_end:
```

```
3036 { \_pdf_backend:n { eann } }
```

(End definition for `_pdf_backend_link_begin_goto:nw` and others.)

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

```
3037 \cs_new:Npn \_pdf_backend_link_last:
3038 { @pdf.lnk \int_use:N \g_pdf_backend_link_int }
```

(End definition for `_pdf_backend_link_last:.`)

`_pdf_backend_link_margin:n` Pass to `dvipdfmx`.

```
3039 \cs_new_protected:Npn \_pdf_backend_link_margin:n #1
3040 { \_kernel_backend_literal:x { dvipdfmx.config-g~ \dim_eval:n {#1} } }
```

(End definition for `_pdf_backend_link_margin:n`.)

`_pdf_backend_destination:nn`
`_pdf_backend_destination:nmnn`
`_pdf_backend_destination_aux:nmnn`

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.

```
3041 \cs_new_protected:Npn \_pdf_backend_destination:nn #1#2
3042 {
3043   \_pdf_backend:x
3044   {
3045     dest ~ ( \exp_not:n {#1} )
3046     [
3047       @thispage
3048       \str_case:nmF {#2}
3049       {
3050         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
3051         { fit } { /Fit }
3052         { fitb } { /FitB }
3053         { fitbh } { /FitBH }
3054         { fitbv } { /FitBV ~ @xpos }
3055         { fith } { /FitH ~ @ypos }
3056         { fitv } { /FitV ~ @xpos }
3057         { fitr } { /Fit }
3058       }
3059       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
3060     ]
3061   }
3062 }
3063 \cs_new_protected:Npn \_pdf_backend_destination:nmnn #1#2#3#4
3064 {
3065   \exp_args:Ne \_pdf_backend_destination_aux:nmnn
3066   { \dim_eval:n {#2} } {#1} {#3} {#4}
3067 }
3068 \cs_new_protected:Npn \_pdf_backend_destination_aux:nmnn #1#2#3#4
3069 {
3070   \vbox_to_zero:n
3071   {
3072     \_kernel_kern:n {#4}
3073     \hbox:n
3074     {
3075       \_pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
3076       \_pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
```



```

3077     }
3078     \tex_vss:D
3079   }
3080   \__kernel_kern:n {#1}
3081   \vbox_to_zero:n
3082   {
3083     \__kernel_kern:n { -#3 }
3084     \hbox:n
3085     {
3086       \__pdf_backend:n
3087       {
3088         dest ~ (#2)
3089         [
3090           @thispage
3091           /FitR ~
3092           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
3093           @xpos ~ @ypos
3094         ]
3095       }
3096     }
3097     \tex_vss:D
3098   }
3099   \__kernel_kern:n { -#1 }
3100 }

```

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

6.4.4 Structure

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

```

3101 \cs_new_protected:Npn \__pdf_backend_compresslevel:n #1
3102 { \__kernel_backend_literal:x { dvipdfmx:config~z~ \int_eval:n {#1} } }
3103 \cs_new_protected:Npn \__pdf_backend_compress_objects:n #1
3104 {
3105   \bool_if:nF {#1}
3106   { \__kernel_backend_literal:n { dvipdfmx:config~C~0x40 } }
3107 }

```

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

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

```

3108 \cs_new_protected:Npn \__pdf_backend_version_major_gset:n #1
3109 {
3110   \cs_gset:Npx \__pdf_backend_version_major: { \int_eval:n {#1} }
3111   \__kernel_backend_literal:x { pdf:majorversion~ \__pdf_backend_version_major: }
3112 }
3113 \cs_new_protected:Npn \__pdf_backend_version_minor_gset:n #1
3114 {
3115   \cs_gset:Npx \__pdf_backend_version_minor: { \int_eval:n {#1} }
3116   \__kernel_backend_literal:x { pdf:minorversion~ \__pdf_backend_version_minor: }
3117 }

```

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

```

\__pdf_backend_version_major: We start with the assumption that the default is active.
\__pdf_backend_version_minor: 3118 \cs_new:Npn \__pdf_backend_version_major: { 1 }
                               3119 \cs_new:Npn \__pdf_backend_version_minor: { 5 }

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

```

6.4.5 Marked content

```

\__pdf_backend_bdc:nn Simple wrappers. May need refinement: see https://chat.stackexchange.com/transcript/message/49970158#49970158.
\__pdf_backend_emc:
3120 \cs_new_protected:Npn \__pdf_backend_bdc:nn #1#2
3121 { \__kernel_backend_literal_page:n { /#1 ~ #2 ~ BDC } }
3122 \cs_new_protected:Npn \__pdf_backend_emc:
3123 { \__kernel_backend_literal_page:n { EMC } }

(End definition for \__pdf_backend_bdc:nn and \__pdf_backend_emc:.)
3124 </dviPDFmx | xetex>

```

6.5 dvisvgm backend

```
3125 <*dvisvgm>
```

6.5.1 Annotations

```

\__pdf_backend_annotation:nmmn
3126 \cs_new_protected:Npn \__pdf_backend_annotation:nmmn #1#2#3#4 { }

(End definition for \__pdf_backend_annotation:nmmn.)

```

```

\__pdf_backend_annotation_last:
3127 \cs_new:Npn \__pdf_backend_annotation_last: { }

(End definition for \__pdf_backend_annotation_last:.)

```

```

\__pdf_backend_link_begin_goto:nnw
\__pdf_backend_link_begin_user:nnw 3128 \cs_new_protected:Npn \__pdf_backend_link_begin_goto:nnw #1#2 { }
\__pdf_backend_link_begin:nmmw 3129 \cs_new_protected:Npn \__pdf_backend_link_begin_user:nnw #1#2 { }
\__pdf_backend_link_end: 3130 \cs_new_protected:Npn \__pdf_backend_link_begin:nmmw #1#2#3 { }
3131 \cs_new_protected:Npn \__pdf_backend_link_end: { }

(End definition for \__pdf_backend_link_begin_goto:nnw and others.)

```

```

\__pdf_backend_link_last:
3132 \cs_new:Npx \__pdf_backend_link_last: { }

(End definition for \__pdf_backend_link_last:.)

```

```

\__pdf_backend_link_margin:n A simple task: pass the data to the primitive.
3133 \cs_new_protected:Npn \__pdf_backend_link_margin:n #1 { }

(End definition for \__pdf_backend_link_margin:n.)

```

```

\__pdf_backend_destination:nn
\__pdf_backend_destination:nmmn 3134 \cs_new_protected:Npn \__pdf_backend_destination:nn #1#2 { }
3135 \cs_new_protected:Npn \__pdf_backend_destination:nmmn #1#2#3#4 { }

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

```

6.5.2 Catalogue entries

No-op.

```
\_pdf_backend_catalog_gput:nn
\__pdf_backend_info_gput:nn 3136 \cs_new_protected:Npn \_pdf_backend_catalog_gput:nn #1#2 { }
3137 \cs_new_protected:Npn \_pdf_backend_info_gput:nn #1#2 { }

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

6.5.3 Objects

All no-ops here.

```
\_pdf_backend_object_new:n
\__pdf_backend_object_ref:n 3138 \cs_new_protected:Npn \_pdf_backend_object_new:nn #1 { }
\_pdf_backend_object_write:nn 3139 \cs_new:Npn \_pdf_backend_object_ref:n #1 { }
\_pdf_backend_object_write:nx 3140 \cs_new_protected:Npn \_pdf_backend_object_write:nnn #1#2#3 { }
\_pdf_backend_object_now:nn 3141 \cs_new_protected:Npn \_pdf_backend_object_write:nnx #1#2#3 { }
\_pdf_backend_object_now:nx 3142 \cs_new_protected:Npn \_pdf_backend_object_now:nn #1#2 { }
\_pdf_backend_object_last: 3143 \cs_new_protected:Npn \_pdf_backend_object_now:nx #1#2 { }
\_pdf_backend_pageobject_ref:n 3144 \cs_new:Npn \_pdf_backend_object_last: { }
3145 \cs_new:Npn \_pdf_backend_pageobject_ref:n #1 { }

(End definition for \_pdf_backend_object_new:n and others.)
```

6.5.4 Structure

These are all no-ops.

```
\_pdf_backend_compresslevel:n
\_pdf_backend_compress_objects:n 3146 \cs_new_protected:Npn \_pdf_backend_compresslevel:n #1 { }
3147 \cs_new_protected:Npn \_pdf_backend_compress_objects:n #1 { }

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

Data not available!

```
\_pdf_backend_version_major_gset:n
\_pdf_backend_version_minor_gset:n 3148 \cs_new_protected:Npn \_pdf_backend_version_major_gset:n #1 { }
3149 \cs_new_protected:Npn \_pdf_backend_version_minor_gset:n #1 { }

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

Data not available!

```
\_pdf_backend_version_major:
\_pdf_backend_version_minor: 3150 \cs_new:Npn \_pdf_backend_version_major: { -1 }
3151 \cs_new:Npn \_pdf_backend_version_minor: { -1 }

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

More no-ops.

```
\__pdf_backend_bdc:nn
\_pdf_backend_emc: 3152 \cs_new_protected:Npn \_pdf_backend_bdc:nn #1#2 { }
3153 \cs_new_protected:Npn \_pdf_backend_emc: { }

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

3154 </dvisvgm>
```

6.6 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.

```
3155 <*dvipdfmx | dvips>
```

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

```
3156 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
3157 {
3158   \__kernel_backend_first_shipout:n
3159   {
3160     \__kernel_backend_literal:e
3161     {
3162       <*dvipdfmx>
3163         pdf:pagesize ~
3164         width ~ \dim_eval:n {#1} ~
3165         height ~ \dim_eval:n {#2}
3166       </dvipdfmx>
3167       <*dvips>
3168         papersize = \dim_eval:n {#1} , \dim_eval:n {#2}
3169       </dvips>
3170     }
3171   }
3172 }
```

(End definition for _pdf_backend_pagesize_gset:nn.)

```
3173 </dvipdfmx | dvips>
```

```
3174 <*luatex | pdftex | xetex>
```

`_pdf_backend_pagesize_gset:nn` Pass to the primitives.

```
3175 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2
3176 {
3177   \dim_gset:Nn \tex_pagewidth:D {#1}
3178   \dim_gset:Nn \tex_pageheight:D {#2}
3179 }
```

(End definition for _pdf_backend_pagesize_gset:nn.)

```
3180 </luatex | pdftex | xetex>
```

```
3181 <*dvisvgm>
```

`_pdf_backend_pagesize_gset:nn` A no-op.

```
3182 \cs_new_protected:Npn \_pdf_backend_pagesize_gset:nn #1#2 { }
```

(End definition for _pdf_backend_pagesize_gset:nn.)

```
3183 </dvisvgm>
```

```
3184 </package>
```

7 I3backend-opacity implementation

```
3185 (*package)
3186 (@@=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.

```
3187 (*dvips)
```

`_opacity_backend_select:n` 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.

```
3188 \\cs_new_protected:Npn \\_opacity_backend_select:n #1
3189 {
3190   \\exp_args:Nx \\_opacity_backend_select_aux:n
3191   { \\fp_eval:n { min(max(0,#1),1) } }
3192 }
3193 \\cs_new_protected:Npn \\_opacity_backend_select_aux:n #1
3194 {
3195   \\_opacity_backend:nnn {#1} { fill } { ca }
3196   \\_opacity_backend:nnn {#1} { stroke } { CA }
3197 }
3198 \\cs_new_protected:Npn \\_opacity_backend_fill:n #1
3199 {
3200   \\_opacity_backend:xnn
3201   { \\fp_eval:n { min(max(0,#1),1) } }
3202   { fill }
3203   { ca }
3204 }
3205 \\cs_new_protected:Npn \\_opacity_backend_stroke:n #1
3206 {
3207   \\_opacity_backend:xnn
3208   { \\fp_eval:n { min(max(0,#1),1) } }
3209   { stroke }
3210   { CA }
3211 }
3212 \\cs_new_protected:Npn \\_opacity_backend:nnn #1#2#3
3213 {
3214   \\_kernel_backend_postscript:n
3215   {
3216     product ~ (Ghostscript) ~ search
3217     {
3218       pop ~ pop ~ pop ~
3219       #1 ~ .set #2 constantalpha
3220     }
3221     {
3222       pop ~
3223       mark ~
3224       /#3 ~ #1
```

```

3225         /SetTransparency ~
3226         pdfmark
3227     }
3228     ifelse
3229 }
3230 }
3231 \cs_generate_variant:Nn \_opacity_backend:nnn { x }

```

(End definition for _opacity_backend_select:n and others.)

```

3232 </dvips>
3233 <*dvipdfmx | luatex | pdftex | xetex>

```

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

```

3234 \bool_lazy_and:nnT
3235 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3236 { \pdfmanagement_if_active_p:}
3237 {
3238 <*luatex | pdftex>
3239     \__kernel_color_backend_stack_init:Nnn \c__opacity_backend_stack_int
3240     { page ~ direct } { /opacity 1 ~ gs }
3241 </luatex | pdftex>
3242     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3243     { opacity 1 } { << /ca ~ 1 /CA ~ 1 >> }
3244 }

```

(End definition for \c__opacity_backend_stack_int.)

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

```

\l__opacity_backend_stroke_tl
3245 \tl_new:N \l__opacity_backend_fill_tl
3246 \tl_new:N \l__opacity_backend_stroke_tl

```

(End definition for \l__opacity_backend_fill_tl and \l__opacity_backend_stroke_tl.)

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

```

\__opacity_backend_select_aux:n
\__opacity_backend_reset:
3247 \cs_new_protected:Npn \__opacity_backend_select:n #1
3248 {
3249     \exp_args:Nx \__opacity_backend_select_aux:n
3250     { \fp_eval:n { min(max(0,#1),1) } }
3251 }
3252 \cs_new_protected:Npn \__opacity_backend_select_aux:n #1
3253 {
3254     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3255     \tl_set:Nn \l__opacity_backend_stroke_tl {#1}
3256     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3257     { opacity #1 }
3258     { << /ca ~ #1 /CA ~ #1 >> }
3259 <*dvipdfmx | xetex>
3260     \__kernel_backend_literal_pdf:n
3261 </dvipdfmx | xetex>
3262 <*luatex | pdftex>
3263     \__kernel_color_backend_stack_push:nm \c__opacity_backend_stack_int
3264 </luatex | pdftex>
3265     { /opacity #1 ~ gs }
3266     \group_insert_after:N \__opacity_backend_reset:

```

```

3267 }
3268 \bool_lazy_and:nnF
3269 { \cs_if_exist_p:N \pdfmanagement_if_active_p: }
3270 { \pdfmanagement_if_active_p:}
3271 {
3272   \cs_gset_protected:Npn \__opacity_backend_select_aux:n #1 { }
3273 }
3274 \cs_new_protected:Npn \__opacity_backend_reset:
3275 {
3276   <*dvipdfmx | xetex>
3277   \__kernel_backend_literal_pdf:n
3278   { /opacity1 ~ gs }
3279   </dvipdfmx | xetex>
3280   <*luatex | pdftex>
3281   \__kernel_color_backend_stack_pop:n \c__opacity_backend_stack_int
3282   </luatex | pdftex>
3283 }

```

(End definition for __opacity_backend_select:n, __opacity_backend_select_aux: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_fillstroke:mn 3284 \cs_new_protected:Npn \__opacity_backend_fill:n #1
\__opacity_backend_fillstroke:xx 3285 {
3286   \__opacity_backend_fill_stroke:xx
3287   { \fp_eval:n { min(max(0,#1),1) } }
3288   \l__opacity_backend_stroke_tl
3289 }
3290 \cs_new_protected:Npn \__opacity_backend_stroke:n #1
3291 {
3292   \__opacity_backend_fill_stroke:xx
3293   \l__opacity_backend_fill_tl
3294   { \fp_eval:n { min(max(0,#1),1) } }
3295 }
3296 \cs_new_protected:Npn \__opacity_backend_fill_stroke:nn #1#2
3297 {
3298   \str_if_eq:nnTF {#1} {#2}
3299   { \__opacity_backend_select_aux:n {#1} }
3300   {
3301     \tl_set:Nn \l__opacity_backend_fill_tl {#1}
3302     \tl_set:Nn \l__opacity_backend_stroke_tl {#2}
3303     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3304     { opacity.fill #1 }
3305     { << /ca ~ #1 >> }
3306     \pdfmanagement_add:nnn { Page / Resources / ExtGState }
3307     { opacity.stroke #1 }
3308     { << /CA ~ #2 >> }
3309   <*dvipdfmx | xetex>
3310   \__kernel_backend_literal_pdf:n
3311   </dvipdfmx | xetex>
3312   <*luatex | pdftex>
3313   \__kernel_color_backend_stack_push:nn \c__opacity_backend_stack_int
3314   </luatex | pdftex>

```

```

3315         { /opacity.fill #1 ~ gs /opacity.stroke #2 ~ gs }
3316         \group_insert_after:N \_opacity_backend_reset:
3317     }
3318 }
3319 \cs_generate_variant:Nn \_opacity_backend_fill_stroke:nn { xx }

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

3320 </dviptfm | luatex | pdftex | xetex>
3321 <*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
3322 \cs_new_protected:Npn \_opacity_backend_select:n #1
3323   { \_opacity_backend:nn {#1} { } }
3324 \cs_new_protected:Npn \_opacity_backend_fill:n #1
3325   { \_opacity_backend:nn {#1} { fill- } }
3326 \cs_new_protected:Npn \_opacity_backend_stroke:n #1
3327   { \_opacity_backend:nn {#1} { stroke- } }
3328 \cs_new_protected:Npn \_opacity_backend:nn #1#2
3329   { \_kernel_backend_scope:x { #2 opacity = " \fp_eval:n { min(max(0,#1),1) } " } }

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

3330 </dvisvgm>
3331 </package>

```

7.1 Font handling integration

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

```

3332 <*lua>

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

3333 local pdfmanagement_active do
3334   local pdfmanagement_if_active_p = token.create'pdfmanagement_if_active_p:'
3335   local cmd = pdfmanagement_if_active_p.cmdname
3336   if cmd == 'undefined_cs' then
3337     pdfmanagement_active = false
3338   else
3339     token.put_next(pdfmanagement_if_active_p)
3340     pdfmanagement_active = token.scan_int() ~= 0
3341   end
3342 end
3343
3344 if pdfmanagement_active and luaotfload and luaotfload.set_transparent_colorstack then
3345   luaotfload.set_transparent_colorstack(token.create'c__opacity_backend_stack_int'.index)
3346
3347   local transparent_register = {
3348     token.create'pdfmanagement_add:nnn',
3349     token.new(0, 1),
3350     'Page/Resources/ExtGState',
3351     token.new(0, 2),

```



```

3352     token.new(0, 1),
3353     '',
3354     token.new(0, 2),
3355     token.new(0, 1),
3356     '<</ca ',
3357     '',
3358     '/CA ',
3359     '',
3360     '>>',
3361     token.new(0, 2),
3362 }
3363 luatexbase.add_to_callback('luaotfload.parse_transparent', function(value)
3364     value = (octet * -1):match(value)
3365     if not value then
3366         tex.error'Invalid transparency value'
3367         return
3368     end
3369     value = value:sub(1, -2)
3370     local result = 'opacity' .. value
3371     tex.runtoks(function()
3372         transparent_register[6], transparent_register[10], transparent_register[12] = result,
3373         tex.sprint(-2, transparent_register)
3374     end)
3375     return '/' .. result .. ' gs'
3376 end, 'l3opacity')
3377 end
3378 </lua>

```

8 l3backend-header implementation

```

3379 <*dvips & header>

```

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

```

3380 /color.sc { } def

```

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

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

```

3381 TeXDict begin
3382 /TeXcolorseparation { setcolor } def
3383 end

```

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

`pdf.globaldict` A small global dictionary for backend use.

```

3384 true setglobal
3385 /pdf.globaldict 4 dict def
3386 false setglobal

```

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

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

```

pdf.dvi.pt
pdf.pt.dvi
pdf.rect.ht
3387 /pdf.cvs { 65534 string cvs } def
3388 /pdf.dvi.pt { 72.27 mul Resolution div } def
3389 /pdf.pt.dvi { 72.27 div Resolution mul } def
3390 /pdf.rect.ht { dup 1 get neg exch 3 get add } def

```

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

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

```

pdf.linkdp.pad 3391 /pdf.linkmargin { 1 pdf.pt.dvi } def
pdf.linkht.pad 3392 /pdf.linkdp.pad { 0 } def
3393 /pdf.linkht.pad { 0 } def

```

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

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

```

pdf.save.ll
pdf.save.ur
pdf.save.linkll 3394 /pdf.rect
pdf.save.linkur 3395 { /Rect [ pdf.llx pdf.lly pdf.urx pdf.ury ] } def
pdf.llx 3396 /pdf.save.ll
pdf.lly 3397 {
pdf.urx 3398     currentpoint
pdf.ury 3399     /pdf.lly exch def
3400     /pdf.llx exch def
3401 }
3402 def
3403 /pdf.save.ur
3404 {
3405     currentpoint
3406     /pdf.ury exch def
3407     /pdf.urx exch def
3408 }
3409 def
3410 /pdf.save.linkll
3411 {
3412     currentpoint
3413     pdf.linkmargin add
3414     pdf.linkdp.pad add
3415     /pdf.lly exch def
3416     pdf.linkmargin sub
3417     /pdf.llx exch def
3418 }
3419 def
3420 /pdf.save.linkur
3421 {
3422     currentpoint
3423     pdf.linkmargin sub
3424     pdf.linkht.pad sub
3425     /pdf.ury exch def
3426     pdf.linkmargin add

```

```

3427 /pdf.urx exch def
3428 }
3429 def

```

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

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

```

pdf.dev.x 3430 /pdf.dest.anchor
pdf.dev.y 3431 {
pdf.tmpa 3432 currentpoint exch
pdf.tmpb 3433 pdf.dvi.pt 72 add
pdf.tmpc 3434 /pdf.dest.x exch def
pdf.tmpd 3435 pdf.dvi.pt
3436 vsize 72 sub exch sub
3437 /pdf.dest.y exch def
3438 }
3439 def
3440 /pdf.dest.point
3441 { pdf.dest.x pdf.dest.y } def
3442 /pdf.dest2device
3443 {
3444 /pdf.dest.y exch def
3445 /pdf.dest.x exch def
3446 matrix currentmatrix
3447 matrix defaultmatrix
3448 matrix invertmatrix
3449 matrix concatmatrix
3450 cvx exec
3451 /pdf.dev.y exch def
3452 /pdf.dev.x exch def
3453 /pdf.tmpd exch def
3454 /pdf.tmpc exch def
3455 /pdf.tmpb exch def
3456 /pdf.tmpa exch def
3457 pdf.dest.x pdf.tmpa mul
3458 pdf.dest.y pdf.tmpc mul add
3459 pdf.dev.x add
3460 pdf.dest.x pdf.tmpb mul
3461 pdf.dest.y pdf.tmpd mul add
3462 pdf.dev.y add
3463 }
3464 def

```

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

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

```

pdf.bordertracking.begin
pdf.bordertracking.end
pdf.leftboundary
pdf.rightboundary 3465 /pdf.bordertracking false def
pdf.brokenlink.rect
pdf.brokenlink.skip
pdf.brokenlink.dict
pdf.bordertracking.endpage
pdf.bordertracking.continue
pdf.originx
pdf.originy

```

```

3466 /pdf.bordertracking.begin
3467 {
3468   SDict /pdf.bordertracking true put
3469   SDict /pdf.leftboundary undef
3470   SDict /pdf.rightboundary undef
3471   /a where
3472     {
3473       /a
3474       {
3475         currentpoint pop
3476         SDict /pdf.rightboundary known dup
3477         {
3478           SDict /pdf.rightboundary get 2 index lt
3479           { not }
3480           if
3481         }
3482         if
3483           { pop }
3484           { SDict exch /pdf.rightboundary exch put }
3485         ifelse
3486         moveto
3487         currentpoint pop
3488         SDict /pdf.leftboundary known dup
3489         {
3490           SDict /pdf.leftboundary get 2 index gt
3491           { not }
3492           if
3493         }
3494         if
3495           { pop }
3496           { SDict exch /pdf.leftboundary exch put }
3497         ifelse
3498       }
3499     } put
3500   }
3501   if
3502 }
3503 def
3504 /pdf.bordertracking.end
3505 {
3506   /a where { /a { moveto } put } if
3507   /x where { /x { 0 exch rmoveto } put } if
3508   SDict /pdf.leftboundary known
3509     { pdf.outerbox 0 pdf.leftboundary put }
3510   if
3511   SDict /pdf.rightboundary known
3512     { pdf.outerbox 2 pdf.rightboundary put }
3513   if
3514   SDict /pdf.bordertracking false put
3515 }
3516 def
3517 /pdf.bordertracking.endpage
3518 {
3519   pdf.bordertracking

```

```

3520 {
3521 pdf.bordertracking.end
3522 true setglobal
3523 pdf.globaldict
3524 /pdf.brokenlink.rect [ pdf.outerbox aload pop ] put
3525 pdf.globaldict
3526 /pdf.brokenlink.skip pdf.baselineskip put
3527 pdf.globaldict
3528 /pdf.brokenlink.dict
3529 pdf.link.dict pdf.cvs put
3530 false setglobal
3531 mark pdf.link.dict cvx exec /Rect
3532 [
3533 pdf.llx
3534 pdf.lly
3535 pdf.outerbox 2 get pdf.linkmargin add
3536 currentpoint exch pop
3537 pdf.outerbox pdf.rect.ht sub pdf.linkmargin sub
3538 ]
3539 /ANN pdf.pdfmark
3540 }
3541 if
3542 }
3543 def
3544 /pdf.bordertracking.continue
3545 {
3546 /pdf.link.dict pdf.globaldict
3547 /pdf.brokenlink.dict get def
3548 /pdf.outerbox pdf.globaldict
3549 /pdf.brokenlink.rect get def
3550 /pdf.baselineskip pdf.globaldict
3551 /pdf.brokenlink.skip get def
3552 pdf.globaldict dup dup
3553 /pdf.brokenlink.dict undef
3554 /pdf.brokenlink.skip undef
3555 /pdf.brokenlink.rect undef
3556 currentpoint
3557 /pdf.originy exch def
3558 /pdf.originx exch def
3559 /a where
3560 {
3561 /a
3562 {
3563 moveto
3564 SDict
3565 begin
3566 currentpoint pdf.originy ne exch
3567 pdf.originx ne or
3568 {
3569 pdf.save.linkll
3570 /pdf.lly
3571 pdf.lly pdf.outerbox 1 get sub def
3572 pdf.bordertracking.begin
3573 }

```

```

3574         if
3575         end
3576     }
3577     put
3578 }
3579 if
3580 /x where
3581 {
3582     /x
3583     {
3584         0 exch rmoveto
3585         SDict
3586         begin
3587         currentpoint
3588         pdf.originy ne exch pdf.originx ne or
3589         {
3590             pdf.save.linkll
3591             /pdf.lly
3592             pdf.lly pdf.outerbox 1 get sub def
3593             pdf.bordertracking.begin
3594         }
3595         if
3596         end
3597     }
3598     put
3599 }
3600 if
3601 }
3602 def

```

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

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

```

3603 /pdf.breaklink
3604 {
3605     pop
3606     counttomark 2 mod 0 eq
3607     {
3608         counttomark /pdf.count exch def
3609         {
3610             pdf.count 0 eq { exit } if
3611             counttomark 2 roll
3612             1 index /Rect eq
3613             {
3614                 dup 4 array copy
3615                 dup dup
3616                 1 get
3617                 pdf.outerbox pdf.rect.ht
3618                 pdf.linkmargin 2 mul add sub
3619                 3 exch put

```

```

3620     dup
3621     pdf.outerbox 2 get
3622     pdf.linkmargin add
3623     2 exch put
3624     dup dup
3625     3 get
3626     pdf.outerbox pdf.rect.ht
3627     pdf.linkmargin 2 mul add add
3628     1 exch put
3629     /pdf.currentrect exch def
3630     pdf.breaklink.write
3631     {
3632     pdf.currentrect
3633     dup
3634     pdf.outerbox 0 get
3635     pdf.linkmargin sub
3636     0 exch put
3637     dup
3638     pdf.outerbox 2 get
3639     pdf.linkmargin add
3640     2 exch put
3641     dup dup
3642     1 get
3643     pdf.baselineskip add
3644     1 exch put
3645     dup dup
3646     3 get
3647     pdf.baselineskip add
3648     3 exch put
3649     /pdf.currentrect exch def
3650     pdf.breaklink.write
3651     }
3652     1 index 3 get
3653     pdf.linkmargin 2 mul add
3654     pdf.outerbox pdf.rect.ht add
3655     2 index 1 get sub
3656     pdf.baselineskip div round cvi 1 sub
3657     exch
3658     repeat
3659     pdf.currentrect
3660     dup
3661     pdf.outerbox 0 get
3662     pdf.linkmargin sub
3663     0 exch put
3664     dup dup
3665     1 get
3666     pdf.baselineskip add
3667     1 exch put
3668     dup dup
3669     3 get
3670     pdf.baselineskip add
3671     3 exch put
3672     dup 2 index 2 get 2 exch put
3673     /pdf.currentrect exch def

```

```

3674         pdf.breaklink.write
3675         SDict /pdf.pdfmark.good false put
3676         exit
3677     }
3678     { pdf.count 2 sub /pdf.count exch def }
3679     ifelse
3680 }
3681 loop
3682 }
3683 if
3684 /ANN
3685 }
3686 def
3687 /pdf.breaklink.write
3688 {
3689     counttomark 1 sub
3690     index /_objdef eq
3691     {
3692         counttomark -2 roll
3693         dup wcheck
3694         {
3695             readonly
3696             counttomark 2 roll
3697         }
3698         { pop pop }
3699     } ifelse
3700 }
3701 if
3702 counttomark 1 add copy
3703 pop pdf.currentrect
3704 /ANN pdfmark
3705 }
3706 def

```

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

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.

```

3707 /pdf.pdfmark
3708 {
3709     SDict /pdf.pdfmark.good true put
3710     dup /ANN eq
3711     {
3712         pdf.pdfmark.store
3713         pdf.pdfmark.dict
3714         begin
3715             Subtype /Link eq
3716             currentdict /Rect known and
3717             SDict /pdf.outerbox known and
3718             SDict /pdf.baselineskip known and
3719             {

```



```

3720         Rect 3 get
3721         pdf.linkmargin 2 mul add
3722         pdf.outerbox pdf.rect.ht add
3723         Rect 1 get sub
3724         pdf.baselineskip div round cvi 0 gt
3725         { pdf.breaklink }
3726         if
3727     }
3728     if
3729     end
3730     SDict /pdf.outerbox undef
3731     SDict /pdf.baselineskip undef
3732     currentdict /pdf.pdfmark.dict undef
3733 }
3734 if
3735 pdf.pdfmark.good
3736 { pdfmark }
3737 { cleartomark }
3738 ifelse
3739 }
3740 def
3741 /pdf.pdfmark.store
3742 {
3743     /pdf.pdfmark.dict 65534 dict def
3744     counttomark 1 add copy
3745     pop
3746     {
3747         dup mark eq
3748         {
3749             pop
3750             exit
3751         }
3752         {
3753             pdf.pdfmark.dict
3754             begin def end
3755         }
3756     } ifelse
3757 }
3758 loop
3759 }
3760 def

```

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

```
3761 </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.

A	
<code>\AtBeginDvi</code>	57
B	
bool commands:	
<code>\bool_gset_false:N</code>	1192, 1211, 1234, 1256, 1272, 1373, 1612, 1648, 2394, 2440
<code>\bool_gset_true:N</code>	1190, 1259, 1371, 1627, 2387, 2393
<code>\bool_if:NTF</code>	67, 569, 1202, 1206, 1222, 1225, 1229, 1240, 1247, 1251, 1263, 1267, 1384, 1389, 1394, 1586, 1631, 1770, 1820, 1960, 2002, 2382, 2397, 2402, 2407
<code>\bool_if:nTF</code>	2616, 2869, 3105
<code>\bool_lazy_and:nnTF</code>	782, 2119, 3234, 3268
<code>\bool_lazy_any:nTF</code>	1809
<code>\bool_lazy_or:nnTF</code>	1995
<code>\bool_new:N</code>	1193, 1260, 1374, 1628, 2367, 2368
<code>\bool_set_false:N</code>	1782, 1924, 2026, 2190
box commands:	
<code>\box_dp:N</code>	208, 210, 258, 260, 315, 317, 364, 366, 368, 370, 2419, 2452, 2453, 2478
<code>\box_ht:N</code>	210, 260, 317, 368, 370, 1833, 2067, 2424, 2463, 2464, 2480
<code>\box_if_empty:NTF</code>	2513
<code>\box_move_down:nn</code>	2341, 2419
<code>\box_move_up:nn</code>	2211, 2343, 2424
<code>\box_new:N</code>	2237, 2331, 2332
<code>\box_set_dp:Nn</code>	1711
<code>\box_set_ht:Nn</code>	1710
<code>\box_set_wd:Nn</code>	272, 1709
<code>\box_use:N</code>	215, 233, 247, 263, 290, 304, 320, 336, 348, 399, 413, 432, 1324, 1519, 1712, 2372
<code>\box_wd:N</code>	209, 217, 259, 265, 316, 322, 365, 367, 1832, 2066
box internal commands:	
<code>__box_backend_clip:N</code>	197, 197, 252, 252, 309, 309, 353, 353
<code>\l_box_backend_cos_fp</code>	267
<code>__box_backend_rotate:Nn</code>	219, 219, 267, 267, 324, 324, 403, 403
<code>__box_backend_rotate_aux:Nn</code>	219, 220, 221, 267, 268, 269, 324, 325, 326
<code>__box_backend_scale:Nnn</code>	236, 236, 295, 295, 339, 339, 416, 416
<code>\l_box_backend_sin_fp</code>	267
C	
clist commands:	
<code>\clist_map_function:nN</code>	1280, 1404, 1655
color internal commands:	
<code>__color_backend:nnn</code>	1018, 1025, 1040, 1048, 1054
<code>__color_backend_cmyk:w</code>	1019
<code>\g_color_backend_colorant_prop</code>	535, 554, 557, 577, 818
<code>__color_backend_devicen_colorants:n</code>	536, 536, 738, 876
<code>__color_backend_devicen_colorants:w</code>	536, 544, 551, 559
<code>__color_backend_devicen_init:nnn</code>	725, 725, 843, 843, 1075, 1075
<code>__color_backend_devicen_init:w</code>	843, 852, 881, 885
<code>__color_backend_fill:n</code>	922, 922, 924, 925, 926, 948, 949, 951, 953, 954, 973, 982, 983, 985, 987, 988, 999, 1008, 1009, 1011, 1013, 1014
<code>__color_backend_fill_cmyk:n</code>	922, 924, 948, 948, 982, 982, 1008, 1008
<code>__color_backend_fill_devicen:nn</code>	932, 942, 972, 976, 998, 1002, 1069, 1071
<code>__color_backend_fill_gray:n</code>	922, 925, 948, 950, 982, 984, 1008, 1010
<code>__color_backend_fill_reset</code>	944, 944, 978, 978, 1004, 1004, 1073, 1073
<code>__color_backend_fill_rgb:n</code>	922, 926, 948, 952, 982, 986, 1008, 1012
<code>__color_backend_fill_separation:nn</code>	932, 932, 942, 972, 972, 976, 998, 998, 1002, 1069, 1069, 1071
<code>\l_color_backend_fill_tl</code>	498, 510, 956, 970
<code>__color_backend_iccbackend_device:nnn</code>	905, 905

`_color_backend_iccbased_-`
`init:nnn` 744, 744, 887, 887, 1075, 1076
`_color_backend_init_resource:n`
..... 779, 779, 808, 879, 903, 918
`_color_backend_reset:`
..... 479, 494, 502, 514,
518, 523, 944, 945, 978, 979, 1004, 1073
`_color_backend_rgb:w` 1042
`_color_backend_select:n`
..... 479, 480, 482, 484,
486, 487, 518, 518, 520, 521, 522, 564
`_color_backend_select:nn`
..... 502, 503, 505, 507, 508, 775
`_color_backend_select_cmyk:n` ..
..... 479, 479, 502, 502, 518, 520
`_color_backend_select_devicen:nn`
..... 563, 565, 747, 748, 769, 777
`_color_backend_select_gray:n` ..
.... 479, 481, 502, 504, 518, 521, 528
`_color_backend_select_iccbased:nn`
..... 566, 566, 751, 751, 769, 778
`_color_backend_select_named:n` ..
..... 479, 483, 525, 525
`_color_backend_select_rgb:n` ...
..... 479, 485, 502, 506, 518, 522
`_color_backend_select_separation:nn`
..... 563, 563, 565,
747, 747, 748, 769, 770, 774, 777, 778
`_color_backend_separation_-`
`init:n` 567, 648, 661
`_color_backend_separation_-`
`init:nn` 796, 806, 810
`_color_backend_separation_-`
`init:nnn` 567, 602, 623
`_color_backend_separation_-`
`init:nnnn` 567, 625, 637
`_color_backend_separation_-`
`init:nnnnn` 567,
567, 588, 681, 749, 749, 796, 796, 836
`_color_backend_separation_-`
`init:nw` 567, 652, 663, 677
`_color_backend_separation_-`
`init:w` 567, 639, 654, 659
`_color_backend_separation_-`
`init_/DeviceCMYK:nnn` 567
`_color_backend_separation_-`
`init_/DeviceGray:nnn` 567
`_color_backend_separation_-`
`init_/DeviceRGB:nnn` 567
`_color_backend_separation_-`
`init_aux:nnnnn` 567, 573, 589
`_color_backend_separation_-`
`init_CIELAB:nnn`
..... 567, 679, 749, 796, 821
`_color_backend_separation_-`
`init_CIELAB:nnnnn` 750
`_color_backend_separation_-`
`init_count:n` 567, 626, 629
`_color_backend_separation_-`
`init_count:w` ... 567, 630, 631, 635
`_color_backend_separation_-`
`init_Device:Nn`
..... 567, 611, 613, 615, 616
`\l_color_backend_stack_int`
..... 440, 512, 515, 957, 969
`_color_backend_stroke:n`
..... 922, 927, 929,
930, 931, 948, 961, 963, 965, 966, 975
`_color_backend_stroke_cmyk:n` ..
..... 922, 929, 948, 960, 982, 992, 1018, 1018
`_color_backend_stroke_cmyk:w` ..
..... 1018, 1020
`_color_backend_stroke_devicen:nn`
..... 932,
943, 972, 977, 998, 1003, 1069, 1072
`_color_backend_stroke_gray:n` ..
..... 922, 930, 948, 962, 982, 994, 1018, 1031
`_color_backend_stroke_gray_-`
`aux:n` 1018, 1035, 1039
`_color_backend_stroke_reset:` ..
..... 944,
945, 978, 979, 1004, 1005, 1073, 1074
`_color_backend_stroke_rgb:n` ...
..... 922, 931, 948, 964, 982, 996, 1018, 1041
`_color_backend_stroke_rgb:w` ...
..... 1018, 1043
`_color_backend_stroke_separation:nn`
..... 932, 937, 943, 972, 974,
977, 998, 1000, 1003, 1069, 1070, 1072
`\l_color_backend_stroke_tl`
..... 498, 511, 958, 968
`\g_color_model_int` 574, 583, 731,
759, 808, 814, 815, 869, 870, 879, 903
`\c_color_model_range_CIELAB_tl` ..
..... 686, 721, 832, 839
`color.sc` 479, 3380
`cs commands:`
`\cs_generate_variant:Nn`
..... 49, 63, 66, 99, 138, 143, 154, 185,
191, 588, 1139, 1334, 1528, 1974,
2037, 2057, 2242, 2263, 2326, 2820,
2833, 2943, 2964, 2994, 3231, 3319
`\cs_gset:Npx` .. 2628, 2632, 3110, 3115
`\cs_gset_protected:Npn` 3272
`\cs_if_exist:NTF`
..... 27, 50, 1722, 2509, 2894, 2920
`\cs_if_exist_p:N` 783, 3235, 3269

<code>\cs_if_exist_use:NTF</code>	38, 601	1682, 1684, 1686, 1688, 1690, 1698,
<code>\cs_new:Npn</code>		1720, 1739, 1762, 1779, 1793, 1798,
. 551, 610, 612, 614, 616, 623, 629,		1806, 1836, 1849, 1867, 1877, 1893,
631, 637, 654, 661, 663, 881, 1285,		1912, 1921, 1929, 1941, 1947, 1950,
1409, 1659, 1835, 2070, 2228, 2255,		1965, 1975, 2014, 2023, 2029, 2035,
2327, 2329, 2362, 2534, 2634, 2635,		2038, 2045, 2058, 2063, 2071, 2078,
2787, 2802, 2821, 2822, 2925, 2957,		2095, 2129, 2160, 2161, 2163, 2165,
2995, 2997, 3013, 3037, 3118, 3119,		2167, 2173, 2179, 2187, 2193, 2196,
3127, 3139, 3144, 3145, 3150, 3151		2198, 2209, 2240, 2243, 2245, 2248,
<code>\cs_new:Npx</code>		2257, 2264, 2281, 2286, 2291, 2296,
536, 2655, 2690, 2834, 2845, 2912, 3132		2306, 2311, 2319, 2334, 2339, 2371,
<code>\cs_new_eq:NN</code>	46, 57, 59,	2373, 2378, 2380, 2385, 2400, 2405,
520, 521, 522, 565, 748, 777, 778,		2442, 2471, 2490, 2499, 2536, 2543,
924, 925, 926, 929, 930, 931, 942,		2569, 2574, 2602, 2614, 2626, 2630,
943, 944, 945, 976, 977, 978, 979,		2636, 2638, 2642, 2666, 2668, 2670,
1002, 1003, 1004, 1071, 1072, 1073,		2681, 2701, 2711, 2734, 2748, 2758,
1138, 1333, 1339, 1340, 1527, 1529,		2769, 2789, 2823, 2856, 2867, 2873,
1530, 1536, 1736, 1737, 1750, 1752,		2901, 2935, 2937, 2944, 2946, 2950,
1777, 1778, 1841, 1842, 1843, 1866,		2959, 2965, 2970, 2975, 2977, 2979,
1891, 1908, 1909, 1918, 1919, 1920,		2987, 3000, 3016, 3018, 3035, 3039,
1940, 1943, 1944, 1945, 2010, 2020,		3041, 3063, 3068, 3101, 3103, 3108,
2021, 2022, 2176, 2177, 2185, 2186,		3113, 3120, 3122, 3126, 3128, 3129,
2195, 2225, 2226, 2227, 2231, 2372		3130, 3131, 3133, 3134, 3135, 3136,
<code>\cs_new_protected:Npn</code>		3137, 3138, 3140, 3141, 3142, 3143,
. 47, 54, 61, 64, 72,		3146, 3147, 3148, 3149, 3152, 3153,
78, 83, 85, 89, 100, 110, 119, 128,		3156, 3175, 3182, 3188, 3193, 3198,
141, 144, 146, 148, 152, 157, 166,		3205, 3212, 3247, 3252, 3274, 3284,
176, 186, 197, 219, 221, 236, 252,		3290, 3296, 3322, 3324, 3326, 3328
267, 269, 295, 309, 324, 326, 339,		<code>\cs_new_protected:Npx</code>
353, 403, 416, 443, 457, 467, 479,	 567, 1054, 2884, 2941, 3020
481, 483, 485, 487, 494, 502, 504,		<code>\cs_set_eq:NN</code> 2530, 2531
506, 508, 514, 518, 523, 525, 563,		<code>\cs_set_protected:Npn</code> 2133
566, 589, 679, 725, 744, 747, 749,		
750, 751, 770, 774, 779, 796, 810,		
821, 843, 887, 905, 922, 927, 932,		
937, 948, 950, 952, 954, 960, 962,		
964, 966, 972, 974, 982, 984, 986,		
988, 992, 994, 996, 998, 1000, 1005,		
1008, 1010, 1012, 1014, 1018, 1020,		
1031, 1039, 1041, 1043, 1069, 1070,		
1074, 1075, 1076, 1140, 1145, 1150,		
1152, 1154, 1162, 1170, 1179, 1189,		
1191, 1194, 1196, 1213, 1218, 1236,		
1258, 1261, 1274, 1287, 1292, 1294,		
1296, 1298, 1300, 1302, 1304, 1306,		
1311, 1335, 1337, 1341, 1346, 1351,		
1361, 1370, 1372, 1375, 1377, 1379,		
1381, 1386, 1391, 1396, 1398, 1411,		
1416, 1418, 1420, 1422, 1424, 1426,		
1428, 1430, 1441, 1466, 1478, 1490,		
1502, 1509, 1531, 1537, 1542, 1547,		
1558, 1568, 1578, 1580, 1582, 1584,		
1615, 1617, 1622, 1624, 1626, 1629,		
1650, 1661, 1674, 1676, 1678, 1680,		

D	
dim commands:	
<code>\dim_compare:nNnTF</code>	2109, 2114
<code>\dim_compare_p:nNn</code>	2120, 2121
<code>\dim_eval:n</code>	
. . . 2337, 2572, 2650, 2651, 2652,	
2709, 2744, 2745, 2746, 3007, 3008,	
3009, 3040, 3066, 3164, 3165, 3168	
<code>\dim_gset:Nn</code>	3177, 3178
<code>\dim_max:nn</code>	2450, 2461
<code>\dim_set:Nn</code>	
. . 1832, 1833, 2066, 2067, 2105, 2106	
<code>\dim_set_eq:NN</code>	2171
<code>\dim_to_decimal:n</code> . . 364, 365, 366,	
367, 368, 370, 1540, 1545, 1551,	
1552, 1553, 1554, 1563, 1564, 1565,	
1656, 1675, 2218, 2219, 2448, 2459,	
2477, 2478, 2479, 2480, 2484, 2540	
<code>\dim_to_decimal_in_bp:n</code>	
. . . . 208, 209, 210, 258, 259, 260,	
315, 316, 317, 1158, 1159, 1166,	

1167, 1174, 1175, 1183, 1184, 1185,
 1282, 1286, 1290, 1344, 1349, 1355,
 1356, 1357, 1365, 1366, 1406, 1410,
 1414, 1660, 1744, 1745, 1746, 1747,
 1934, 1935, 1936, 1937, 1989, 1990,
 1991, 1992, 2203, 2204, 2205, 2206
 \dim_zero:N 2103, 2104
 \c_max_dim
 .. 2105, 2106, 2109, 2114, 2120, 2121
 draw internal commands:
 __draw_align_currentpoint... .. 35
 __draw_backend_add_to_path:n ...
 1537,
 1539, 1544, 1549, 1560, 1568, 1583
 __draw_backend_begin:
 .. 1140, 1140, 1335, 1335, 1531, 1531
 __draw_backend_box_use:Nnnnn ...
 31, 1311, 1311, 1509, 1509, 1698, 1698
 __draw_backend_cap_but:
 .. 1274, 1294, 1398, 1418, 1650, 1678
 __draw_backend_cap_rectangle: ..
 .. 1274, 1298, 1398, 1422, 1650, 1682
 __draw_backend_cap_round:
 .. 1274, 1296, 1398, 1420, 1650, 1680
 __draw_backend_clip:
 .. 1194, 1258, 1375, 1391, 1582, 1626
 __draw_backend_closepath:
 1194, 1194,
 1215, 1375, 1375, 1582, 1582, 1619
 __draw_backend_closestroke: ...
 .. 1194, 1213, 1375, 1379, 1582, 1617
 __draw_backend_cm:nnnn
 ... 1306, 1306, 1319, 1320, 1321,
 1430, 1430, 1513, 1690, 1690, 1701
 __draw_backend_cm_aux:nnnn
 1430, 1437, 1441
 __draw_backend_cm_decompose:nnnnN
 1436, 1465, 1466
 __draw_backend_cm_decompose_-
 auxi:nnnnN 1465, 1470, 1478
 __draw_backend_cm_decompose_-
 auxii:nnnnN 1465, 1482, 1490
 __draw_backend_cm_decompose_-
 auxiii:nnnnN 1465, 1494, 1502
 __draw_backend_curveto:nnnnnn ..
 .. 1154, 1179, 1341, 1351, 1537, 1558
 __draw_backend_dash:n
 1274, 1280, 1285,
 1398, 1404, 1409, 1650, 1655, 1659
 __draw_backend_dash_aux:nn
 1650, 1654, 1661
 __draw_backend_dash_pattern:nn .
 .. 1274, 1274, 1398, 1398, 1650, 1650
 __draw_backend_discardpath: ...
 .. 1194, 1261, 1375, 1396, 1582, 1629
 __draw_backend_end:
 .. 1140, 1145, 1335, 1337, 1531, 1536
 __draw_backend_evenodd_rule: ...
 .. 1189, 1189, 1370, 1370, 1578, 1578
 __draw_backend_fill:
 .. 1194, 1218, 1375, 1381, 1582, 1622
 __draw_backend_fillstroke:
 .. 1194, 1236, 1375, 1386, 1582, 1624
 __draw_backend_join_bevel:
 .. 1274, 1304, 1398, 1428, 1650, 1688
 __draw_backend_join_miter:
 .. 1274, 1300, 1398, 1424, 1650, 1684
 __draw_backend_join_round:
 .. 1274, 1302, 1398, 1426, 1650, 1686
 __draw_backend_lineto:mn
 .. 1154, 1162, 1341, 1346, 1537, 1542
 __draw_backend_linewidth:n
 .. 1274, 1287, 1398, 1411, 1650, 1674
 __draw_backend_literal:n
 1138, 1138, 1139, 1143,
 1147, 1151, 1153, 1156, 1164, 1172,
 1181, 1195, 1198, 1199, 1200, 1201,
 1204, 1210, 1220, 1227, 1233, 1238,
 1243, 1244, 1245, 1246, 1249, 1255,
 1265, 1271, 1276, 1289, 1293, 1295,
 1297, 1299, 1301, 1303, 1305, 1308,
 1313, 1314, 1315, 1316, 1317, 1318,
 1322, 1323, 1325, 1326, 1327, 1328,
 1329, 1333, 1333, 1334, 1343, 1348,
 1353, 1363, 1376, 1378, 1380, 1383,
 1388, 1393, 1397, 1400, 1413, 1417,
 1419, 1421, 1423, 1425, 1427, 1429,
 1527, 1527, 1528, 1589, 1608, 1634
 __draw_backend_miterlimit:n ...
 .. 1274, 1292, 1398, 1416, 1650, 1676
 __draw_backend_moveto:nn
 .. 1154, 1154, 1341, 1341, 1537, 1537
 __draw_backend_nonzero_rule: ...
 .. 1189, 1191, 1370, 1372, 1578, 1580
 __draw_backend_path:n
 1582, 1584, 1616, 1623, 1625
 \g__draw_backend_path_int 1597, 1614
 \g__draw_backend_path_tl
 1537, 1593, 1609, 1611, 1638
 __draw_backend_rectangle:nnnn ..
 .. 1154, 1170, 1341, 1361, 1537, 1547
 __draw_backend_scope_begin: 1150,
 1150, 1336, 1339, 1339, 1529, 1529
 __draw_backend_scope_end: 1150,
 1152, 1338, 1339, 1340, 1529, 1530
 __draw_backend_stroke: 1194, 1196,
 1216, 1375, 1377, 1582, 1615, 1620

<code>\g__draw_draw_clip_bool</code> ..	1194 , 1582	graphics internal commands:
<code>\g__draw_draw_eor_bool</code>		<code>\l__graphics_attr_tl</code>
...	1189 , 1206 , 1222 , 1229 , 1240 ,	1761 ,
	1251 , 1267 , 1370 , 1384 , 1389 , 1394	1766 , 1783 , 1795 , 1802 , 1804 , 1839
<code>\g__draw_draw_path_int</code>	1582	<code>__graphics_backend_dequote:w</code> ..
<code>\g__draw_path_tl</code>	1647
		1762 , 1801 , 1835
		<code>\l__graphics_backend_dir_str</code> .
		1844
		<code>\l__graphics_backend_ext_str</code> .
		1844
		<code>__graphics_backend_get_pagecount:n</code>
	
		1751 , 1752 , 1893 , 1893 ,
		2008 , 2010 , 2078 , 2078 , 2230 , 2231
		<code>__graphics_backend_getbb_auxi:n</code>
	
		1762 , 1775 , 1791 , 1793
		<code>__graphics_backend_getbb_-</code>
		auxi:nN
		2014 , 2018 , 2027 , 2029
		<code>__graphics_backend_getbb_-</code>
		auxii:n
		1762 , 1796 , 1798
		<code>__graphics_backend_getbb_-</code>
		auxiii:nnN ..
		2014 , 2032 , 2035 , 2037
		<code>__graphics_backend_getbb_-</code>
		auxiiii:n
		1762 , 1800 , 1806
		<code>__graphics_backend_getbb_-</code>
		auxiii:nNnn ..
		2014 , 2033 , 2036 , 2038
		<code>__graphics_backend_getbb_-</code>
		auxiv:nnNnn ..
		2014 , 2041 , 2045 , 2057
		<code>__graphics_backend_getbb_-</code>
		auxv:nNnn ..
		2014 , 2042 , 2049 , 2058
		<code>__graphics_backend_getbb_-</code>
		auxvi:nNnn
		2061 , 2063
		<code>__graphics_backend_getbb_bmp:n</code> .
	
		1906 , 1920 , 2014 , 2022
		<code>__graphics_backend_getbb_eps:n</code> .
	
		1734 , 1736 , 1844 ,
		1849 , 1866 , 1906 , 1908 , 2174 , 2176
		<code>__graphics_backend_getbb_eps:nm</code>
	
		1844
		<code>__graphics_backend_getbb_eps:n</code>
	
		1855 , 1867
		<code>__graphics_backend_getbb_jpeg:n</code>
	
		1762 , 1777 ,
		1906 , 1918 , 2014 , 2020 , 2179 , 2185
		<code>__graphics_backend_getbb_jpg:n</code> .
	
		1762 , 1762 , 1777 , 1778 , 1906 , 1912 ,
		1918 , 1919 , 1920 , 2014 , 2014 , 2020 ,
		2021 , 2022 , 2179 , 2179 , 2185 , 2186
		<code>__graphics_backend_getbb_-</code>
		pagebox:w ..
		2014 , 2053 , 2070 , 2076
		<code>__graphics_backend_getbb_pdf:n</code> .
	
		1762 , 1779 , 1875 ,
		1906 , 1921 , 2014 , 2023 , 2187 , 2187
		<code>__graphics_backend_getbb_png:n</code> .
	
		1762 , 1778 ,
		1906 , 1919 , 2014 , 2021 , 2179 , 2186
		<code>__graphics_backend_getbb_ps:n</code> ..
	
		1734 , 1737 ,
E		
<code>\errmessage</code>	38	
<code>\evensidemargin</code>	2417	
exp commands:		
<code>\exp_after:wN</code>	2076	
<code>\exp_args:Ne</code>		
...	625 , 1800 , 1857 , 1883 , 2571 , 3065	
<code>\exp_args:Nf</code>	1279 , 1403 , 2336	
<code>\exp_args:NNf</code>	220 , 268 , 325	
<code>\exp_args:Nnx</code>	2990	
<code>\exp_args:Nx</code>	571 , 806 , 1855 ,	
	1881 , 2293 , 2308 , 2413 , 3190 , 3249	
<code>\exp_not:N</code>	538 , 544 , 545 , 546 ,	
	573 , 574 , 577 , 578 , 583 , 2657 , 2659 ,	
	2662 , 2692 , 2694 , 2697 , 2836 , 2838 ,	
	2841 , 2847 , 2849 , 2852 , 2889 , 2890 ,	
	2896 , 2897 , 2916 , 2921 , 3022 , 3027	
<code>\exp_not:n</code>	48 , 97 , 108 , 136 ,	
	895 , 2284 , 2289 , 2565 , 2806 , 2807 ,	
	2821 , 2822 , 2968 , 2973 , 2984 , 3045	
<code>\ExplBackendFileDate</code>	1	
F		
file commands:		
<code>\file_compare_timestamp:nNnTF</code> .	1869	
<code>\file_parse_full_name:nNnn</code>	1851 , 1879	
<code>\fmtversion</code>	52	
fp commands:		
<code>\fp_compare:nNnTF</code>		
.	227 , 274 , 280 , 332 , 1446 , 1459 , 1504	
<code>\fp_eval:n</code> .	220 , 229 , 242 , 243 , 268 ,	
	285 , 300 , 302 , 325 , 334 , 345 , 346 ,	
	410 , 425 , 426 , 1026 , 1027 , 1028 ,	
	1036 , 1049 , 1050 , 1051 , 1448 , 1453 ,	
	1454 , 1461 , 1471 , 1472 , 1473 , 1474 ,	
	1483 , 1484 , 1485 , 1486 , 1495 , 1496 ,	
	1497 , 1498 , 2562 , 2731 , 3059 , 3191 ,	
	3201 , 3208 , 3250 , 3287 , 3294 , 3329	
<code>\fp_new:N</code>	293 , 294	
<code>\fp_set:Nn</code>	273 , 276	
<code>\fp_use:N</code>	279 , 283 , 288	
<code>\fp_zero:N</code>	275	
<code>\c_zero_fp</code>	227 , 274 , 280 , 332 , 1446 , 1459	
G		
graphics commands:		
<code>\l__graphics_search_ext_seq</code>		
.....	1732 , 1755 , 1901 , 2089	

[1844](#), [1866](#), [1906](#), [1909](#), [2174](#), [2177](#)
`__graphics_backend_getbb_svg:n` .
 [2095](#), [2095](#)
`__graphics_backend_getbb_svg_-`
`auxi:nNn` ... [2095](#), [2111](#), [2116](#), [2129](#)
`__graphics_backend_getbb_svg_-`
`auxii:w` [2095](#), [2133](#), [2155](#), [2160](#)
`__graphics_backend_getbb_svg_-`
`auxiii:Nw` [2095](#), [2143](#), [2161](#)
`__graphics_backend_getbb_svg_-`
`auxiv:Nw` [2095](#), [2146](#), [2163](#)
`__graphics_backend_getbb_svg_-`
`auxv:Nw` [2095](#), [2147](#), [2165](#)
`__graphics_backend_getbb_svg_-`
`auxvi:Nn` [2095](#), [2162](#), [2164](#), [2166](#), [2167](#)
`__graphics_backend_getbb_svg_-`
`auxvii:w` [2095](#), [2169](#), [2173](#)
`__graphics_backend_include:nn` ..
 [2193](#), [2194](#), [2197](#), [2198](#)
`__graphics_backend_include_-`
`auxi:nn` [1929](#), [1942](#), [1948](#), [1950](#)
`__graphics_backend_include_-`
`auxii:nnn` .. [1929](#), [1952](#), [1965](#), [1974](#)
`__graphics_backend_include_-`
`auxiii:nnn` [1929](#), [1972](#), [1975](#)
`__graphics_backend_include_-`
`bmp:n` [1929](#), [1945](#)
`__graphics_backend_include_-`
`dequote:w` [2209](#), [2220](#), [2228](#)
`__graphics_backend_include_-`
`eps:n` [1739](#),
[1739](#), [1750](#), [1844](#), [1877](#), [1891](#),
[1929](#), [1929](#), [1940](#), [2193](#), [2193](#), [2195](#)
`__graphics_backend_include_-`
`jpeg:n` . [1836](#), [1841](#), [1943](#), [2209](#), [2226](#)
`__graphics_backend_include_-`
`jpg:n` [1836](#),
[1836](#), [1841](#), [1842](#), [1843](#), [1929](#),
[1941](#), [1943](#), [1944](#), [1945](#), [2209](#), [2227](#)
`__graphics_backend_include_-`
`jpeg:n` [1929](#)
`__graphics_backend_include_-`
`pdf:n` [1836](#), [1842](#), [1881](#),
[1929](#), [1947](#), [2071](#), [2071](#), [2193](#), [2196](#)
`__graphics_backend_include_-`
`png:n`
 .. [1836](#), [1843](#), [1929](#), [1944](#), [2209](#), [2225](#)
`__graphics_backend_include_ps:n`
 [1739](#), [1750](#),
[1844](#), [1891](#), [1929](#), [1940](#), [2193](#), [2195](#)
`__graphics_backend_include_-`
`svg:n` .. [2209](#), [2209](#), [2225](#), [2226](#), [2227](#)
`__graphics_backend_loaded:n` ...
[1720](#), [1720](#), [1732](#), [1734](#), [1751](#), [1755](#),
[1901](#), [1906](#), [2009](#), [2089](#), [2174](#), [2230](#)
`\l__graphics_backend_name_str` . [1844](#)
`__graphics_bb_restore:nTF`
 [1795](#), [2060](#), [2097](#)
`__graphics_bb_save:n` [1804](#), [2068](#), [2124](#)
`\l__graphics_decodearray_str` ...
 [1768](#), [1769](#),
[1781](#), [1812](#), [1818](#), [1819](#), [1923](#), [1958](#),
[1959](#), [1997](#), [2000](#), [2001](#), [2025](#), [2189](#)
`__graphics_extract_bb:n`
 [1916](#), [1925](#), [2183](#), [2191](#)
`\l__graphics_final_name_str` .. [1874](#)
`__graphics_get_pagecount:n`
 [1752](#), [2010](#), [2231](#)
`\l__graphics_internal_box`
 .. [1830](#), [1832](#), [1833](#), [2065](#), [2066](#), [2067](#)
`\l__graphics_internal_dim` [2170](#), [2171](#)
`\l__graphics_internal_ior`
 [2099](#), [2100](#), [2107](#), [2126](#)
`\l__graphics_interpolate_bool` ...
 [1770](#), [1782](#), [1811](#), [1820](#),
[1924](#), [1960](#), [1996](#), [2002](#), [2026](#), [2190](#)
`\l__graphics_llx_dim`
 [1744](#), [1934](#), [1989](#), [2103](#), [2203](#)
`\l__graphics_lly_dim`
 [1745](#), [1935](#), [1990](#), [2104](#), [2204](#)
`\l__graphics_page_int`
 [1764](#), [1786](#), [1787](#), [1825](#),
[1826](#), [1914](#), [1956](#), [1957](#), [1983](#), [1984](#),
[2016](#), [2031](#), [2032](#), [2074](#), [2075](#), [2181](#)
`\l__graphics_pagebox_tl`
 [54](#), [1765](#), [1785](#),
[1827](#), [1828](#), [1915](#), [1954](#), [1955](#), [1985](#),
[1987](#), [2017](#), [2040](#), [2041](#), [2076](#), [2182](#)
`\l__graphics_pdf_str`
 .. [1772](#), [1773](#), [1788](#), [1789](#), [1813](#), [1822](#)
`__graphics_read_bb:n`
 .. [1736](#), [1737](#), [1908](#), [1909](#), [2176](#), [2177](#)
`\g__graphics_track_int`
 [1928](#), [1977](#), [1978](#)
`\l__graphics_urx_dim`
 ... [1746](#), [1832](#), [1936](#), [1991](#), [2066](#),
[2105](#), [2109](#), [2112](#), [2120](#), [2205](#), [2218](#)
`\l__graphics_ury_dim`
[1747](#), [1833](#), [1937](#), [1992](#), [2067](#), [2106](#),
[2114](#), [2117](#), [2121](#), [2206](#), [2211](#), [2219](#)
group commands:
`\group_begin:` [163](#), [182](#)
`\group_end:` [171](#)
`\group_insert_after:N` ... [3266](#), [3316](#)

H
hbox commands:
`\hbox:n` [2213](#), [2342](#), [2345](#),

[2420](#), [2426](#), [2579](#), [2586](#), [3073](#), [3084](#)
`\hbox_overlap_right:n` [215](#),
[247](#), [263](#), [304](#), [320](#), [348](#), [432](#), [1324](#), [1519](#)
`\hbox_set:Nn` [1830](#), [2065](#), [2412](#), [2444](#)
`\hbox_set:Nw` [2395](#)
`\hbox_set_end:` [2410](#)
`\hbox_unpack:N` [2531](#)
hook commands:
`\hook_gput_code:nnn` [55](#), [1722](#), [1724](#)

I

int commands:
`\int_compare:nNnTF`
. [1786](#), [1825](#), [1956](#), [1983](#),
[2031](#), [2074](#), [2503](#), [2604](#), [2887](#), [2915](#)
`\int_const:Nn` [445](#), [1802](#),
[1896](#), [1978](#), [2080](#), [2251](#), [2778](#), [2953](#)
`\int_eval:n` [465](#), [475](#), [621](#), [630](#), [643](#),
[645](#), [649](#), [662](#), [2628](#), [2632](#), [2865](#),
[2890](#), [2897](#), [2910](#), [3102](#), [3110](#), [3115](#)
`\int_gincr:N` [189](#),
[355](#), [1588](#), [1633](#), [1977](#), [2250](#), [2321](#),
[2352](#), [2429](#), [2952](#), [2989](#), [3002](#), [3022](#)
`\int_gset:Nn` [164](#), [183](#), [2492](#)
`\int_gset_eq:NN` [172](#), [2353](#), [2430](#), [3003](#)
`\int_if_exist:NTF` [1967](#)
`\int_if_odd:nTF` [2415](#)
`\int_max:nn` [2082](#)
`\int_new:N` [155](#),
[156](#), [402](#), [440](#), [1614](#), [1928](#), [2247](#),
[2333](#), [2364](#), [2366](#), [2948](#), [2999](#), [3015](#)
`\int_set_eq:NN` [160](#), [179](#), [2504](#)
`\int_step_function:nnnN` [647](#)
`\int_use:N`
. [357](#), [388](#), [574](#), [583](#), [731](#), [759](#), [808](#),
[814](#), [815](#), [869](#), [870](#), [879](#), [903](#), [1591](#),
[1597](#), [1604](#), [1636](#), [1644](#), [1787](#), [1826](#),
[1839](#), [1897](#), [1957](#), [1970](#), [1982](#), [1984](#),
[2075](#), [2083](#), [2256](#), [2323](#), [2328](#), [2356](#),
[2363](#), [2434](#), [2535](#), [2788](#), [2798](#), [2958](#),
[2991](#), [2996](#), [3006](#), [3014](#), [3027](#), [3038](#)
`\int_value:w`
. [2657](#), [2692](#), [2836](#), [2847](#), [2865](#)
`\int_zero:N` [1764](#), [1914](#), [2016](#), [2181](#)
ior commands:
`\ior_close:N` [2126](#)
`\ior_if_eof:NTF` [2100](#)
`\ior_map_break:` [2122](#)
`\ior_open:Nn` [2099](#)
`\ior_str_map_inline:Nn` [2107](#)

K

kernel internal commands:
`__kernel_backend_align_begin:`
. [72](#), [72](#), [200](#), [224](#), [239](#)
`__kernel_backend_align_end:`
. [72](#), [78](#), [214](#), [232](#), [246](#)
`__kernel_backend_first_shipout:n`
. [50](#), [54](#), [57](#), [59](#), [69](#), [571](#), [3158](#)
`\g_kernel_backend_header_bool`
. [67](#), [569](#)
`__kernel_backend_literal:n`
. [46](#), [46](#), [47](#), [48](#), [49](#),
[62](#), [65](#), [70](#), [74](#), [81](#), [84](#), [86](#), [142](#), [145](#),
[147](#), [149](#), [153](#), [329](#), [342](#), [489](#), [495](#),
[519](#), [524](#), [591](#), [727](#), [771](#), [923](#), [928](#),
[934](#), [939](#), [990](#), [1016](#), [1142](#), [1148](#),
[1443](#), [1450](#), [1456](#), [1516](#), [1521](#), [1741](#),
[1931](#), [1969](#), [1979](#), [2200](#), [2215](#), [2942](#),
[3040](#), [3102](#), [3106](#), [3111](#), [3116](#), [3160](#)
`__kernel_backend_literal_page:n`
. [100](#),
[100](#), [144](#), [144](#), [2936](#), [2938](#), [3121](#), [3123](#)
`__kernel_backend_literal_pdf:n`
. [89](#), [89](#), [99](#), [141](#), [141](#),
[143](#), [255](#), [312](#), [1333](#), [3260](#), [3277](#), [3310](#)
`__kernel_backend_literal_-
postscript:n` [61](#),
[61](#), [63](#), [75](#), [76](#), [80](#), [201](#), [202](#), [204](#),
[205](#), [213](#), [225](#), [240](#), [1138](#), [2606](#), [2618](#)
`__kernel_backend_literal_svg:n`
. [152](#), [152](#), [154](#), [159](#), [170](#), [178](#), [188](#),
[356](#), [358](#), [375](#), [753](#), [1527](#), [1702](#), [1713](#)
`__kernel_backend_matrix:n`
. [128](#), [128](#), [138](#), [277](#), [298](#), [1433](#)
`__kernel_backend_postscript:n`
. [64](#), [64](#),
[66](#), [491](#), [993](#), [995](#), [997](#), [1001](#), [2241](#),
[2298](#), [2313](#), [2342](#), [2348](#), [2388](#), [2420](#),
[2427](#), [2431](#), [2445](#), [2473](#), [2517](#), [2524](#),
[2530](#), [2538](#), [2545](#), [2579](#), [2586](#), [3214](#)
`__kernel_backend_scope:n`
. [157](#), [186](#), [191](#), [385](#),
[390](#), [1056](#), [1534](#), [1579](#), [1581](#), [1601](#),
[1641](#), [1663](#), [1675](#), [1677](#), [1679](#), [1681](#),
[1683](#), [1685](#), [1687](#), [1689](#), [1692](#), [3329](#)
`__kernel_backend_scope_begin:`
. [83](#), [83](#), [110](#), [110](#), [146](#), [146](#), [157](#), [157](#),
[199](#), [223](#), [238](#), [254](#), [271](#), [297](#), [311](#),
[328](#), [341](#), [1339](#), [1511](#), [1529](#), [1533](#), [1700](#)
`__kernel_backend_scope_begin:n`
. [157](#), [176](#), [185](#), [377](#), [405](#), [418](#)
`__kernel_backend_scope_end:`
. [83](#), [85](#), [110](#), [119](#),
[146](#), [148](#), [157](#), [166](#), [216](#), [234](#), [248](#),
[264](#), [291](#), [305](#), [321](#), [337](#), [349](#), [400](#),
[414](#), [433](#), [1340](#), [1523](#), [1530](#), [1536](#), [1714](#)
`\g_kernel_backend_scope_int`
. [155](#), [162](#), [164](#), [169](#), [173](#), [181](#), [183](#), [189](#)

`\l__kernel_backend_scope_int` . . .
 [155](#), [161](#), [174](#), [180](#)
`\g__kernel_clip_path_int`
 [353](#), [1588](#), [1591](#), [1604](#), [1633](#), [1636](#), [1644](#)
`__kernel_color_backend_stack_-`
`init:Nnn` [443](#), [443](#), [3239](#)
`__kernel_color_backend_stack_-`
`pop:n` [457](#), [467](#), [515](#), [3281](#)
`__kernel_color_backend_stack_-`
`push:nn`
 [457](#), [457](#), [512](#), [957](#), [969](#), [3263](#), [3313](#)
`__kernel_dependency_version_-`
`check:Nn` [1](#)
`__kernel_dependency_version_-`
`check:nn` [27](#), [29](#)
`__kernel_file_name_quote:n`
 [1857](#), [1883](#)
`__kernel_kern:n`
 [2347](#), [2349](#), [2578](#), [2582](#),
[2585](#), [2589](#), [3072](#), [3080](#), [3083](#), [3099](#)

L

lua commands:
`\lua_load_module:n` [1132](#)

M

`\MessageBreak` [40](#)
 mode commands:
`\mode_if_horizontal:TF` [2494](#), [2501](#)
`\mode_if_math:TF` [2392](#)
 msg commands:
`\msg_error:nnn` [529](#), [2101](#)
`\msg_new:nnn` [531](#)

O

`\oddsidemargin` [2416](#)
 opacity internal commands:
`__opacity_backend:nn`
 [3322](#), [3323](#), [3325](#), [3327](#), [3328](#)
`__opacity_backend:nnn` [3188](#),
[3195](#), [3196](#), [3200](#), [3207](#), [3212](#), [3231](#)
`__opacity_backend_fill:n`
 [3188](#), [3198](#), [3284](#), [3284](#), [3322](#), [3324](#)
`__opacity_backend_fill_stroke:nn`
 [3286](#), [3292](#), [3296](#), [3319](#)
`\l__opacity_backend_fill_tl`
 [3245](#), [3254](#), [3293](#), [3301](#)
`__opacity_backend_fillstroke:nn`
 [3284](#)
`__opacity_backend_reset:`
 [3247](#), [3266](#), [3274](#), [3316](#)
`__opacity_backend_select:n`
 [3188](#), [3188](#), [3247](#), [3247](#), [3322](#), [3322](#)

`__opacity_backend_select_aux:n`
 [3188](#), [3190](#),
[3193](#), [3247](#), [3249](#), [3252](#), [3272](#), [3299](#)
`\c__opacity_backend_stack_int`
 [3234](#), [3263](#), [3281](#), [3313](#)
`__opacity_backend_stroke:n`
 [3188](#), [3205](#), [3284](#), [3290](#), [3322](#), [3326](#)
`\l__opacity_backend_stroke_tl`
 [3245](#), [3255](#), [3288](#), [3302](#)

P

pdf commands:
`\pdf_object_if_exist:nTF` [823](#), [889](#), [907](#)
`\pdf_object_new:n`
 [814](#), [825](#), [869](#), [891](#), [909](#)
`\pdf_object_ref:n`
 [771](#), [838](#), [902](#), [917](#), [935](#), [940](#)
`\pdf_object_ref_last:`
 [791](#), [816](#), [819](#), [875](#)
`\pdf_object_unnamed_write:nn`
 [798](#), [845](#), [901](#), [916](#)
`\pdf_object_write:nnn`
 [815](#), [826](#), [870](#), [892](#), [910](#)

pdf internal commands:

`__pdf_backend:n` [2941](#), [2941](#), [2943](#),
[2945](#), [2947](#), [2967](#), [2972](#), [2981](#), [3004](#),
[3023](#), [3036](#), [3043](#), [3075](#), [3076](#), [3086](#)
`__pdf_backend_annotation:nnnn`
 [2334](#), [2334](#),
[2642](#), [2642](#), [3000](#), [3000](#), [3126](#), [3126](#)
`__pdf_backend_annotation_-`
`aux:nnnn` [2336](#), [2339](#)
`\g__pdf_backend_annotation_int`
 [2333](#), [2353](#), [2363](#), [2999](#), [3003](#), [3014](#)
`__pdf_backend_annotation_last:`
 [2362](#), [2362](#),
[2655](#), [2655](#), [3013](#), [3013](#), [3127](#), [3127](#)
`__pdf_backend_bdc:nn` [2636](#), [2636](#),
[2935](#), [2935](#), [3120](#), [3120](#), [3152](#), [3152](#)
`__pdf_backend_catalog_gput:nn`
 [2243](#), [2243](#),
[2748](#), [2748](#), [2944](#), [2944](#), [3136](#), [3136](#)
`__pdf_backend_compress_objects:n`
 [2602](#), [2614](#),
[2856](#), [2867](#), [3101](#), [3103](#), [3146](#), [3147](#)
`__pdf_backend_compresslevel:n`
 [2602](#), [2602](#),
[2856](#), [2856](#), [3101](#), [3101](#), [3146](#), [3146](#)
`\l__pdf_backend_content_box` [2331](#),
[2395](#), [2419](#), [2422](#), [2424](#), [2453](#), [2464](#)
`__pdf_backend_destination:nn`
 [2543](#), [2543](#),
[2711](#), [2711](#), [3041](#), [3041](#), [3134](#), [3134](#)

_pdf_backend_destination:nnnn .	_pdf_backend_link_sf_restore: .
..... 2543, 2569, 2373, 2396, 2439, 2499
2711, 2734, 3041, 3063, 3134, 3135	_pdf_backend_link_sf_save: ...
_pdf_backend_destination_- 2373, 2391, 2409, 2490
aux:nnnn	\l_pdf_backend_model_box . 2332,
.. 2543, 2571, 2574, 3041, 3065, 3068	2412, 2444, 2452, 2463, 2478, 2480
_pdf_backend_emc: .. 2636, 2638,	_pdf_backend_objcompresslevel:n
2935, 2937, 3120, 3122, 3152, 3153 2856, 2870, 2871, 2873
_pdf_backend_info_gput:nn	\g_pdf_backend_object_int
..... 2243, 2245, 2247, 2250,
2748, 2758, 2944, 2946, 3136, 3137	2253, 2321, 2323, 2328, 2352, 2353,
_pdf_backend_link:nw	2356, 2429, 2430, 2948, 2952, 2955,
..... 2373	2989, 2991, 2996, 3002, 3003, 3006
_pdf_backend_link_aux:nw ... 2373	_pdf_backend_object_last:
_pdf_backend_link_begin:n 2327, 2327,
..... 3016, 3017, 3019, 3020	2834, 2834, 2995, 2995, 3138, 3144
_pdf_backend_link_begin:nnnw ..	_pdf_backend_object_new:n 2248,
.. 2666, 2667, 2669, 2670, 3128, 3130	2248, 2769, 2769, 2950, 2950, 3138
_pdf_backend_link_begin:nw ...	_pdf_backend_object_new:nn . 3138
..... 2375, 2379, 2380	_pdf_backend_object_now:nn ...
_pdf_backend_link_begin_aux:nw 2319, 2319, 2326, 2823, 2823, 2833,
..... 2383, 2385	2987, 2987, 2994, 3138, 3142, 3143
_pdf_backend_link_begin_-	\g_pdf_backend_object_prop
goto:nnw 2768, 2948
..... 2373, 2373,	_pdf_backend_object_ref:n
2666, 2666, 3016, 3016, 3128, 3128 2248, 2255, 2260, 2769,
_pdf_backend_link_begin_-	2787, 2950, 2957, 2962, 3138, 3139
user:nnw	_pdf_backend_object_write:nn ..
..... 2373, 2378, 2789, 2800, 2802, 2831, 3138
2666, 2668, 3016, 3018, 3128, 3129	_pdf_backend_object_write:nnn .
\g_pdf_backend_link_bool 2257, 2257, 2263, 2789, 2789, 2820,
..... 2368, 2382, 2387, 2402, 2440	2959, 2959, 2964, 3138, 3140, 3141
\g_pdf_backend_link_dict_tl ...	_pdf_backend_object_write_-
..... 2365, 2390, 2435	array:nn ... 2257, 2281, 2959, 2965
_pdf_backend_link_end:	_pdf_backend_object_write_-
..... 2373, 2400,	aux:nnn ... 2257, 2259, 2264, 2322
2666, 2681, 3016, 3035, 3128, 3131	_pdf_backend_object_write_-
_pdf_backend_link_end_aux: ...	dict:nn
..... 2373, 2403, 2405 2257, 2286, 2959, 2970
\g_pdf_backend_link_int	_pdf_backend_object_write_-
..... 2364, 2430,	fstream:nn . 2257, 2291, 2959, 2975
2434, 2535, 3015, 3022, 3027, 3038	_pdf_backend_object_write_-
_pdf_backend_link_last:	fstream:nnn
..... 2534, 2534, 2294, 2296
2690, 2690, 3037, 3037, 3132, 3132	_pdf_backend_object_write_-
_pdf_backend_link_margin:n ...	stream:nn .. 2257, 2306, 2959, 2977
..... 2536, 2536,	_pdf_backend_object_write_-
2701, 2701, 3039, 3039, 3133, 3133	stream:nnn
\g_pdf_backend_link_math_bool 2257, 2309, 2311
..... 2367, 2393, 2394, 2397, 2407	_pdf_backend_object_write_-
_pdf_backend_link_minima:	stream:nnnn . 2959, 2976, 2978, 2979
..... 2373, 2411, 2442	_pdf_backend_pageobject_ref:n .
_pdf_backend_link_outerbox:n 2329, 2329,
..... 2373, 2413, 2471	2845, 2845, 2997, 2997, 3138, 3145
\g_pdf_backend_link_sf_int	_pdf_backend_pagesize_gset:nn .
..... 2366, 2492, 2503, 2504	.. 3156, 3156, 3175, 3175, 3182, 3182

_pdf_backend_pdfmark:n ..	2240 , 2240 , 2242 , 2244 , 2246 , 2266 , 2283 , 2288 , 2354 , 2546 , 2590 , 2637 , 2639	pdf.llx	2373 , 3394
_pdf_backend_version_major: ...	2628 , 2634 , 2634 , 2912 , 2912 , 3110 , 3111 , 3118 , 3118 , 3150 , 3150	pdf.lly	2373 , 3394
_pdf_backend_version_major_-gset:n	2626 , 2626 , 2884 , 2884 , 3108 , 3108 , 3148 , 3148	pdf.originx	3465
_pdf_backend_version_minor: ...	2632 , 2634 , 2635 , 2912 , 2925 , 3115 , 3116 , 3118 , 3119 , 3150 , 3151	pdf.originy	3465
_pdf_backend_version_minor_-gset:n	2626 , 2630 , 2884 , 2901 , 3108 , 3113 , 3148 , 3149	pdf.outerbox	2373 , 3707
\l_pdf_breaklink_pdfmark_tl	2369 , 2437 , 2529	pdf.pdfmark	3707
_pdf_breaklink_postscript:n	2371 , 2371 , 2421 , 2423 , 2530	pdf.pdfmark.dict	3707
_pdf_breaklink_usebox:N	2372 , 2372 , 2422 , 2531	pdf.pdfmark.good	3707
_pdf_exp_not_i:nn	2789 , 2810 , 2815 , 2821	pdf.pt.dvi	3387
_pdf_exp_not_ii:nn	2789 , 2811 , 2816 , 2822	pdf.rect	3394
\l_pdf_internal_box	2237	pdf.rect.ht	3387
pdf.baselineskip	2373 , 3707	pdf.rightboundary	3465
pdf.bordertracking	3465	pdf.save.linkll	3394
pdf.bordertracking.begin	3465	pdf.save.linkur	3394
pdf.bordertracking.continue	3465	pdf.save.ll	3394
pdf.bordertracking.end	3465	pdf.save.ur	3394
pdf.bordertracking.endpage	3465	pdf.tmpa	3430
pdf.breaklink	3603	pdf.tmpb	3430
pdf.breaklink.write	3603	pdf.tmpc	3430
pdf.brokenlink.dict	3465	pdf.tmpd	3430
pdf.brokenlink.rect	3465	pdf.urx	3394
pdf.brokenlink.skip	3465	pdf.ury	2373 , 3394
pdf.count	3603	pdfmanagement commands:	
pdf.currentrect	3603	\pdfmanagement_add:nnn	788 , 3242 , 3256 , 3303 , 3306
pdf.cvs	3387	\pdfmanagement_if_active_p:	783 , 784 , 3235 , 3236 , 3269 , 3270
pdf.dest.anchor	3430	peek commands:	
pdf.dest.point	3430	\peek_meaning:NtF	2142 , 2145
pdf.dest.x	3430	\peek_remove_spaces:n	2140
pdf.dest.y	3430	prg commands:	
pdf.dest2device	3430	\prg_replicate:nn	168 , 619 , 640 , 650 , 851
pdf.dev.x	3430	prop commands:	
pdf.dev.y	3430	\prop_gput:Nnn	577 , 818
pdf.dvi.pt	3387	\prop_if_in:NnTF	554
pdf.globaldict	3384	\prop_item:Nn	557
pdf.leftboundary	3465	\prop_new:N	535 , 2768 , 2949
pdf.link.dict	2373	\ProvidesExplFile	2
pdf.linkdp.pad	2373 , 3391		
pdf.linkht.pad	2373 , 3391		
pdf.linkmargin	3391		

Q	
quark commands:	
\quark_if_recursion_tail_stop:n	553
\q_recursion_stop	546
\q_recursion_tail	545

S	
scan commands:	
\scan_stop:	113 , 122 , 475 , 2170 , 2173 , 2684 , 2709 , 2732 , 2746 , 2865 , 2882 , 2890 , 2897 , 2910
scan internal commands:	
\s_color_stop	630 , 631 , 635 , 639 , 652 , 655 , 659 , 663 , 677 , 852 , 881 , 885 , 1019 , 1021 , 1042 , 1044

\s__graphics_stop	\tex_pdfinfo:D	2764
..... 1801, 1835, 2135, 2150,	\tex_pdflastannot:D	2662
2157, 2161, 2163, 2165, 2220, 2228	\tex_pdflastlink:D	2697
separation	\tex_pdflastobj:D	2784, 2841
seq commands:	\tex_pdflastximage:D	1803, 1831
\seq_set_from_clist:Nn	\tex_pdflastximagepages:D	1897
..... 1733, 1757, 1903, 2091	\tex_pdflinkmargin:D	2707
skip commands:	\tex_pdfliteral:D	95, 106
\skip_horizontal:n	\tex_pdfmajorversion:D	
..... 217, 265, 322 2894, 2896, 2920, 2921	
str commands:	\tex_pdfminorversion:D ...	2908, 2932
\c_hash_str	\tex_pdfobj:D	2775, 2795, 2829
\c_percent_str	\tex_pdfobjcompresslevel:D ...	2880
\str_case:nn	\tex_pdfpageref:D	2852
\str_case:nnTF	\tex_pdfrefximage:D	1831, 1838
\str_convert_pdfname:n .	\tex_pdfrestore:D	125
\str_if_empty:NTF	\tex_pdfsave:D	116
\str_if_empty_p:N	\tex_pdfsetmatrix:D	134
\str_if_eq:nnTF	\tex_pdfstartlink:D	2676
\str_new:N	\tex_pdfvariable:D	2704,
\str_tail:N	2860, 2877, 2889, 2905, 2916, 2929	
sys commands:	\tex_pdfximage:D	1808, 1895
\sys_if_shell:TF	\tex_spacefactor:D	2495, 2504
\sys_shell_now:n	\tex_special:D	46
	\tex_the:D	1803, 2916, 2921, 2927
	\tex_vss:D	2580, 2587, 3078, 3097
	\tex_XeTeXpdffile:D	2027, 2073
	\tex_XeTeXpdfpagecount:D	2083
	\tex_XeTeXpicfile:D	2018
	TeXcolorseparation	<u>3381</u>
	\textwidth	2479
	tl commands:	
	\c_space_tl	
	. 279, 284, 287, 540, 545, 583, 686,	
	760, 970, 1573, 1743, 1744, 1745,	
	1746, 1933, 1934, 1935, 1936, 1984,	
	1987, 1989, 1990, 1991, 1992, 2053,	
	2075, 2202, 2203, 2204, 2205, 2435,	
	2664, 2699, 2843, 2854, 3006, 3028	
	\tl_clear:N	1765, 1781,
	1915, 1923, 2017, 2025, 2182, 2189	
	\tl_gclear:N	1611, 1647
	\tl_gset:Nn	1570, 2390
	\tl_if_blank:nTF	453, 538,
	634, 651, 658, 676, 802, 884, 2052, 2138	
	\tl_if_empty:NTF .	1573, 1768, 1818,
	1827, 1954, 1958, 1985, 2000, 2040	
	\tl_if_empty:nTF	896, 1667
	\tl_if_empty_p:N	1812, 1997
	\tl_new:N	498,
	499, 1577, 1761, 2365, 2369, 3245, 3246	
	\tl_put_right:Nn	2511

T

TeX and L ^A T _ε commands:	
\@cclv	2513, 2515, 2523
\@ifl@t@r	50, 52
\@makecol@hook	<u>2507</u>
\special	2
tex commands:	
\tex_afterassignment:D	2169
\tex_baselineskip:D	2484
\tex_endinput:D	44
\tex_global:D	
..... 2858, 2875, 2889, 2896, 2903	
\tex_immediate:D	
..... 1808, 2792, 2795, 2826, 2829	
\tex_luatexversion:D	2887, 2915
\tex_pageheight:D	3178
\tex_pagewidth:D	3177
\tex_pdfannot:D	2648
\tex_pdfcatalog:D	2754
\tex_pdfcolorstack:D	463, 473
\tex_pdfcolorstackinit:D	451
\tex_pdfcompresslevel:D	2863
\tex_pdfdest:D	2717, 2740
\tex_pdfendlink:D	2687
\tex_pdfextension:D	
..... 92, 103, 113, 122, 131,	
460, 470, 2645, 2673, 2684, 2714,	
2737, 2751, 2761, 2772, 2792, 2826	
\tex_pdffeedback:D	
... 448, 2659, 2694, 2781, 2838, 2849	

<code>\tl_set:Nn</code>	500, 501, 510, 511, 956, 968, 1766, 1783, 1874, 2370, 2529, 3254, 3255, 3301, 3302	<code>\use:n</code>	59, 786, 812, 867, 1023, 1033, 1046, 1279, 1403, 1468, 1480, 1492, 1652, 2047, 2131, 2153
<code>\tl_to_str:n</code>	2134, 2156, 2252, 2256, 2779, 2788, 2799, 2954, 2958	<code>\use_none:n</code>	1669, 2507
<code>\tl_use:N</code>	718, 831	V	
token commands:		<code>\value</code>	2415
<code>\c_math_toggle_token</code>	2398, 2408	vbox commands:	
U		<code>\vbox_set:Nn</code>	2515
use commands:		<code>\vbox_to_zero:n</code>	2576, 2583, 3070, 3081
<code>\use:N</code>	43, 2279, 2961, 2990	<code>\vbox_unpack_drop:N</code>	2523